AREA EFFICIENT IMPLEMENTATION OF THE ADVANCED ENCRYPTION STANDARD S-BOX LOGIC FUNCTIONS

by

Jia Bao Uang
B.A.Sc., Engineering Science
Simon Fraser University

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

In the School
of
Engineering Science

© Jia Bao Uang 2009

SIMON FRASER UNIVERSITY

Summer 2009

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without permission of the author.

APPROVAL

Name: **Jia Bao Uang**

Degree: **Master of Applied Science**

Title of Thesis: **Area Efficient Implementation of the Advanced Encryption Standard S-BOX Logic Functions**

Examining Committee:

Chair: **Dr. Albert Leung**
Professor of the School of Engineering Science

Dr. Rick Hobson
Senior Supervisor
Professor of the School of Engineering Science

Dr. Lesley Shannon
Supervisor
Assistant Professor of the School of Engineering Science

Dr. Marek Syrzycki
Internal Examiner
Professor of the School of Engineering Science

Date Defended: May 6, 2009

SFU SIMON FRASER UNIVERSITY
LIBRARY

Declaration of
Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website <www.lib.sfu.ca> at: <http://ir.lib.sfu.ca/handle/1892/112>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada

ABSTRACT

The Advanced Encryption Standard (AES) was approved in 2001 and has been used since then. It is more computationally robust compared with previous algorithms, providing a higher level of security. However, it needs more time to execute due to the long calculation and several iterations. The goal of this thesis is to study AES and develop a new hardware algorithm for the most time consuming section of AES to improve the performance.

Four hardware algorithms for data scrambling were implemented through a combination of synthesis and custom layout. One reference circuit was synthesized from truth tables with no custom hardware. The second one makes use of an efficient custom decoder technique. The third and fourth use both the improved decoder and two different full custom Domino-like and_or arrays. Physical layouts were produced through a digital design flow involving Synopsys and Cadence CAD software. Simulations were performed with VHDL and Hspice. Layout area reductions up to 68% were achieved via the new circuits with similar circuit delays to the reference circuit. The power also reduced to roughly 20% of the reference algorithm. The reduced area brought by the custom designs allows the T-BOX architecture, which has a higher throughput rate than the S-BOX architecture to compensate the drawback of occupying more area than S-BOX does.

DEDICATION

To my grandfather and grandmother, your expectation is always my inspiration. I miss you two.

To my Mom and Dad, I cannot pursue this degree and accomplish this achievement without your support and encouragement.

To my girl friend, Ami, thank you for always believing in me.

ACKNOWLEDGEMENTS

I would like to give special thanks to Dr. Rick Hobson. His experience and advice was a huge support to me. Hobson's direction was indispensable for me to finish the thesis. I look forward to discussing new ideas with you in the future.

I would like to thank my colleague, Hooman. Thank you for helping me. It has been a pleasure to work with you.

Finally, I would like to thank all my friends in SFU. Thank you for sharing time with me. Thank you for your help and encouragement. I wish the best for all of you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Acronym	Definition
3-DES	Triple DES Encryption
Abox	A length unit for 0.42 µm
AES	Advanced Encryption Standard, defined in Federal Information Processing Standards Publication 197
ASIC	Application Specific Integrated Circuit
BDD	Binary Decision Diagram
CMOS	Complementary Metal Oxide Semiconductor
DEF	Design Exchange Format
DES	Data Encryption Standard, defined in Federal Information Processing Standard Publication 46-2
DRC	Design Rule Checking
FPGA	Field Programmable Gate Array
GF	Galois field
LEF	Library Exchange Format
LIB	Library file containing cell information
LVS	Layout Versus Schematic
MOSFET	Metal–Oxide–Semiconductor Field-Effect Transistor
PLA	Programmable Logic Array

RCON	Round Constant
SDC	Synthesized Design Constraint
TLF	Timing Library Format
VS18SC	Virtual Silicon Cell Library for 0.18 μm Technology
XOR	Exclusive-Or
XSP	X Layout Grid Spacing
YSP	Y Layout Grid Spacing

1: INTRODUCTION

Communication technology has been popular since the Internet was brought to the world. Over the past two decades, the Internet has become an essential tool in people's lives for communication, entertainment, business, web surfing…etc. It brings convenience to people. Today, hundreds of millions of people over the world are accessing the Internet. The Internet has changed people's lives into a new generation.

Remarkably, the Internet is used as a medium for conducting business such as online banking, emails and online conference. However, Internet crime is also rising fast and severely. People's personal information may be "stolen" while they are having transactions over Internet. Thus, a robust and secure way of sharing information is necessary especially for the government and companies' confidential information.

The Internet Protocol suffers from a number of shortcomings such as being forged and snooped since its header could be hacked easily. Several encryption algorithms such as the Data Encryption Standard (DES) [1], 3-DES [2], and the AES [3] were developed to address this problem.

The DES algorithm, which uses a 56-bit key, was developed in the 1970s and was used for more than 20 years. In 1997, the National Institute of Standard and Technology of United Stated initiated the development of an AES to replace DES [4]. The Rijndael algorithm was chosen for the AES in 2001 due to its higher capability of security [5]. Nevertheless, the higher secure algorithm of AES requires a longer calculation time. The goal of this thesis is to study the AES algorithm and to come up with a solution to improve the performance and throughput rate of the AES algorithm. The algorithm was implemented in hardware rather than software for higher data rate applications. The synthesis tool utilized both standard and custom cells. The standard cells were already built in the cell library and were ready to be used. However, custom cells had to be designed, drawn, and verified by Cadence before importing them for synthesis. The performance of the new algorithms was compared based on the area, power and speed by using Synopsys, Encounter and Hspice.

The thesis is organized as follows: AES algorithm is introduced in Chapter 2. Solutions for faster AES algorithms are discussed in Chapter 3. The flow of the design process is in chapter 4. Chapter 5 lists the results and the comparison with different algorithms. Possible future improvement is in Chapter 6 and the conclusion is in Chapter 7.

2: ENCRYPTION ALGORITHM

The AES provides three levels of security: 128, 192 and 256 bits. The number represents the size of the encryption key used. In theory, AES-128 provides a very high level of security [6] and is good enough for any type of commercial application.

There are two general classes of encryption algorithms [7]:

- Symmetric key
- Asymmetric key (public-key)

A symmetric key encryption algorithm is one that both sender and receiver within the transmission channel share the same key. The key is used in both encryption and decryption. In other words, both ends must know the key. Thus, the key, in practice, represents a shared secret between two or more parties that can be used to maintain a private information link. However, a critical drawback of this algorithm is how the key is distributed. If the key is attached with the encryption data, then the risk of being hacked by other people who can access the transmission channel increases. This drawback will damage the security of the encrypted data and make encryption meaningless.

The solution to this problem is to utilize the asymmetric key algorithm. The asymmetric key algorithm, also known as the public-key algorithm, uses two different keys for encryption and decryption[8]:

- Public key: can only be used in encryption
- Private key: can only be used in decryption

Typically, the private key is kept secret while the public key is widely distributed. The public key can be stored in a public database such as a Certificate Authority. The keys are related mathematically, but the private key cannot be practically derived from the public key. A message encrypted with the public key can be decrypted only with the corresponding private key.

The keys are generated by passing a big random number into a key generation function. The public key is distributed widely so that other people or parties can retrieve it. For encryption, the sender first retrieves the receiver's public key from the database. Then, the message is encrypted with the public key and the encrypted message is transmitted to the recipient site. The recipient can decrypt the message by using the private key. The flow of process is summarized in Figure 1 and Figure 2.

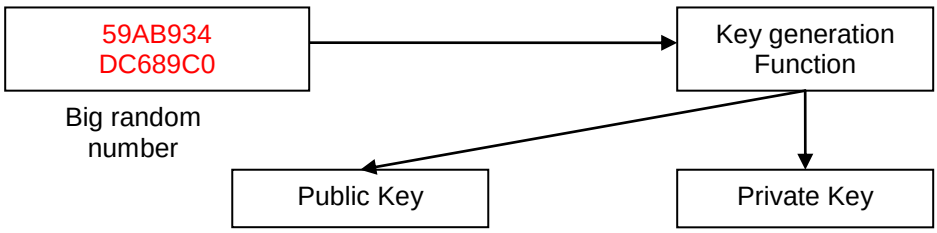


Figure 1: Key Generation

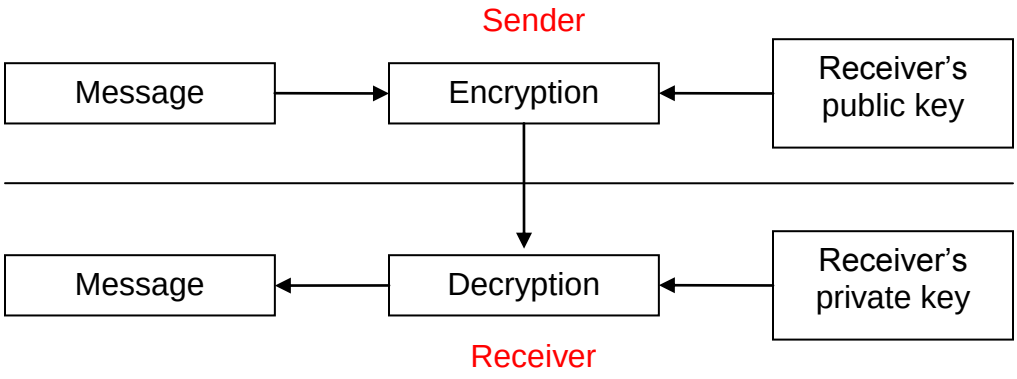


Figure 2: Flow of transmission

Even though the asymmetric key algorithm offers a much higher security than the symmetric key algorithm does, it has the drawback of being computationally intensive and slow. The reason is that the asymmetric encryption algorithm relies heavily on modular exponentiation using large integers [9]. In fact, the asymmetric key algorithm can be as slow as three times or more than the symmetric key algorithm.

A better encryption algorithm can be achieved by combing the advantages of the two encryption algorithms [7]. The message is encrypted by the symmetric-key algorithm while the encryption key used in the symmetric key algorithm is encrypted by the asymmetric key algorithm. In other words, the symmetric key algorithm protects the message that can contain a fair amount of information and would consume a lot of time if the asymmetric key algorithm were applied. Meanwhile the asymmetric key algorithm protects the key used in the symmetric key algorithm since the size of the encryption key is relatively small so that the

time taken to encrypt the key stays short. Thus, the key remains a high level of security. Moreover, this modified algorithm is more efficient and possesses the advantages of both algorithms. The flow of the algorithm is shown in Figure 3.

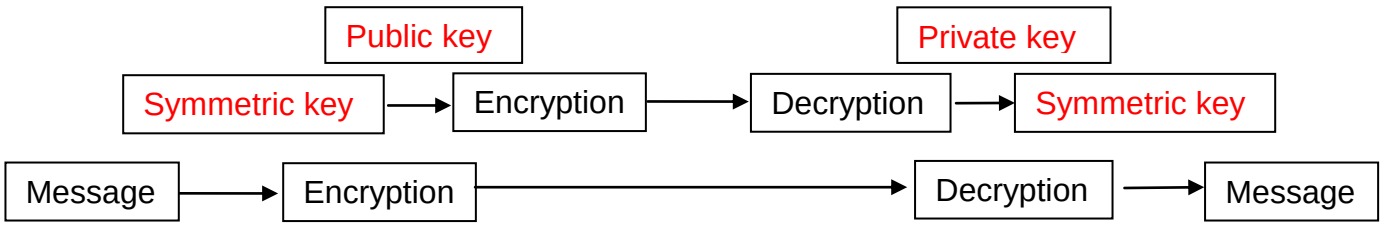


Figure 3: Flow of Encryption/Decryption

There are two types of commonly used ciphers in the symmetric key algorithm [8]: block cipher and stream cipher. A stream cipher operates on individual digits one at a time and the transformation varies during the encryption. The DES and its replacement, AES, belong to the block cipher category. This thesis focuses on the AES algorithm so that the stream cipher algorithm will not be further discussed.

In cryptography, a block cipher is a symmetric key cipher, which operates on fixed-length groups of bits, termed blocks, with an unvarying transformation [10]. For encryption, a block cipher takes a certain size of block of plaintext as input, and outputs a corresponding block of cipher-text. The exact transformation is performed using a second input — the secret key. Decryption is just the

opposite of encryption. The decryption algorithm takes, the block of cipher-text together with the secret key, and yields the original block of plaintext.

2.1 Galois Field

Galois field arithmetic, also known as the finite field, is fundamental to cryptography [3]. A finite field has a set of symbols (usually a sequence of positive binary numbers) together with addition, multiplication, an additive inverse and a multiplicative inverse. The “extended” binary field GF (2^8) has the special property that its 256 symbols can be represented by a positive 8 bit Byte. Addition (and therefore subtraction) is modulo 2 (i.e. without carry), requiring only exclusive-or gates (XOR). A Byte value ($a_7 \ldots a_0$) can also be interpreted as a set of polynomial coefficients, $a_7 2^7 + a_6 2^6 + a_5 2^5 ... + a_0$. In order for a unique multiplicative inverse to exist, multiplication is performed modulo an irreducible polynomial of degree 8. The AES specifies this polynomial to be $m(x) = x^8 + x^4 + x^3 + x + 1$ (1 0001 1011) or 0x11B (hex).

2.2 AES Cipher

The input is operated on 128 bits at a time and is organized into a 4 x 4 matrix called the State. Each element of the state is one byte. As shown in Figure 4, In 0 corresponds to bit 0 (least significant bit) to bit 7 of the input data block.

In 0	In 4	In 8	In 12
In 1	In 5	In 9	In 13
In 2	In 6	In 10	In 14
In 3	In 7	In 11	In 15

Figure 4: Input data table

The AES algorithm involves iterative computation consisting of sequence of operations called “rounds” [11]. Depending on the key size, the number of rounds could be 11, 13 or 15. In addition, each round requires a different key value, which may be pre-computed, or generated on-the –fly from a key expander. The flow of the AES cipher is shown in Figure 5.

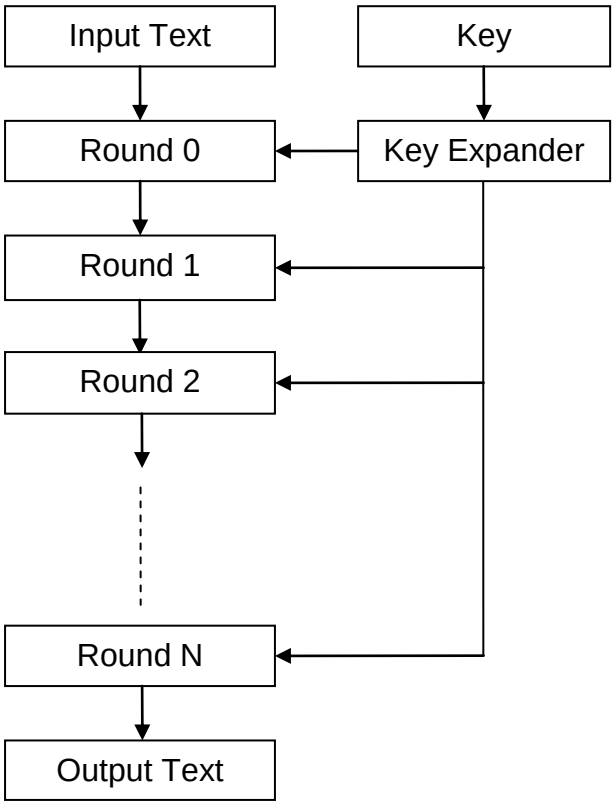


Figure 5: Flow of AES algorithm

(N can be 10, 12 or 14 depending on the key size)

There are several operations within each round. Round 0 only contains the AddRoundKey operation. Round 1 to round N-1 contain 4 operations in sequence as shown in Figure 6 [12]. The last round, round N, contains all operations in Figure 6 except the “MixColumns” operation.

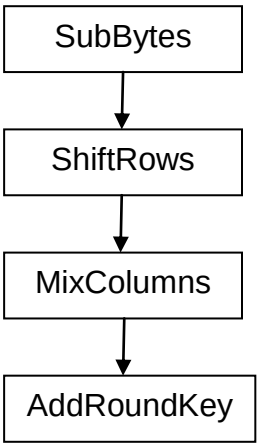


Figure 6: Operations in Rounds

2.2.1 SubBytes

The SubBytes operation simply substitutes each byte of the state with a new value by using a substitution box, the Rijndael S-BOX [13]. This operation provides non-linearity in the cipher. The S-BOX is derived from the multiplicative inverse over GF(2^8) that possesses good non-linearity properties. There are two ways to implement SubBytes operation.

1. Perform matrix multiplication and addition
2. Use look-up table

Figure 7 demonstrates how inputs are derived by simple arithmetic calculations with the matrix where b_n is the input vector and b'_n is the output vector. More detail about S-BOX is discussed in Chapter 3.The look up table can be obtained by substituting all possible 256 values into the input matrix [7].

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 7: SubBytes matrix calculation

2.2.2 ShiftRows

ShiftRows performs circular left shifts on each row of the state. The number of shifts performed is determined by the index of the row. For example, elements on row 0 are not shifted and elements on row 3 are shifted 3 positions left. The process is shown in Figure 8.

S00	S01	S02	S03
S10	S11	S12	S13
S20	S21	S22	S23
S30	S31	S32	S33

→

S00	S01	S02	S03
S11	S12	S13	S10
S22	S23	S20	S21
S33	S30	S31	S32

Figure 8: ShiftRows Process

2.2.3 MixColumns

The MixColumns operation performs integer multiplication and addition on each column to produce a new column. Each column is considered as a four-term polynomial with coefficients over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial *a(x)* , where $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ with inverse, $\{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$ [3]. The transformation can also be expressed by a matrix multiplication or equations, which are displayed in Figure 9.

$$\begin{bmatrix} S'_{0,C} \\ S'_{1,C} \\ S'_{2,C} \\ S'_{3,C} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} S_{0,C} \\ S_{1,C} \\ S_{2,C} \\ S_{3,C} \end{bmatrix}$$

$S'_{0,C} = 2*S_{0,C} + 3*S_{1,C} + 1*S_{2,C} + 1*S_{3,C}$

$S'_{1,C} = 1*S_{0,C} + 2*S_{1,C} + 3*S_{2,C} + 1*S_{3,C}$

$S'_{2,C} = 1*S_{0,C} + 1*S_{1,C} + 2*S_{2,C} + 3*S_{3,C}$

$S'_{3,C} = 3*S_{0,C} + 1*S_{1,C} + 1*S_{2,C} + 2*S_{3,C}$

Figure 9: MixColumns process (C is the index of the columns).

The 1x multiplication (e.g. $1*S_{0,C}$ in Figure 9) is very simple since the output is the input itself. The 2x multiplication (e.g. $2*S_{1,C}$ in Figure 9) is executed differently depending on the initial value of the input. The calculation is done by appending a zero at the right-most bit and shifting the other bits left. If the initial

value of the input is equal to or larger than 128, the product subtracts 0x1B to generate the final result [7] (i.e. modulo 0x11B). On the other hand, if the initial value is less than 128, no further calculation is required. The 3x multiplication is simply the summation of the 1x and 2x results.

2.2.4 AddRoundKey

In each round, the AddRoundKey process XORs the state with a new key value generated from the key expander.

2.2.4.1 Key Expander

As mentioned previously, the AES cipher requires a new key value in each round. The process of generating round keys from a 128-bit input key is depicted in Figure 10 [14]. The key is split into four 32-bit words indexed from 0 to 3. The RotWord process takes a 32-bit word, b0, b1, b2 and b3, and performs a byte permutation [15] to yield b1, b2, b3 and b0. Each byte of the word is then replaced by the same S-BOX as described in the SubBytes process. The word is then XOR'ed with a constant value, RCon, depending on the round number. Table 1 lists the constant value used in each round.

The result generated by the RCon operation forms word0 of the next key, namely NWord0. This value is then XOR'ed with word1 of the input key to

produce NWord1. Word2 is XOR'ed with NWord1 to produce NWord2 and Word3 is XOR'ed with NWord2 to produce NWord3. The Key Expander creates 10 new key values from the input key, for a total of 11 round keys. These round keys can be pre-computed, stored in memory and fetched when they are needed. They may also be computed on-the-fly.

Round	RCon Value
1	0x01000000
2	0x02000000
3	0x04000000
4	0x08000000
5	0x10000000
6	0x20000000
7	0x40000000
8	0x80000000
9	0x1B000000
10	0x36000000

Table 1: Round Constant (RCon) Values for Key Expansion

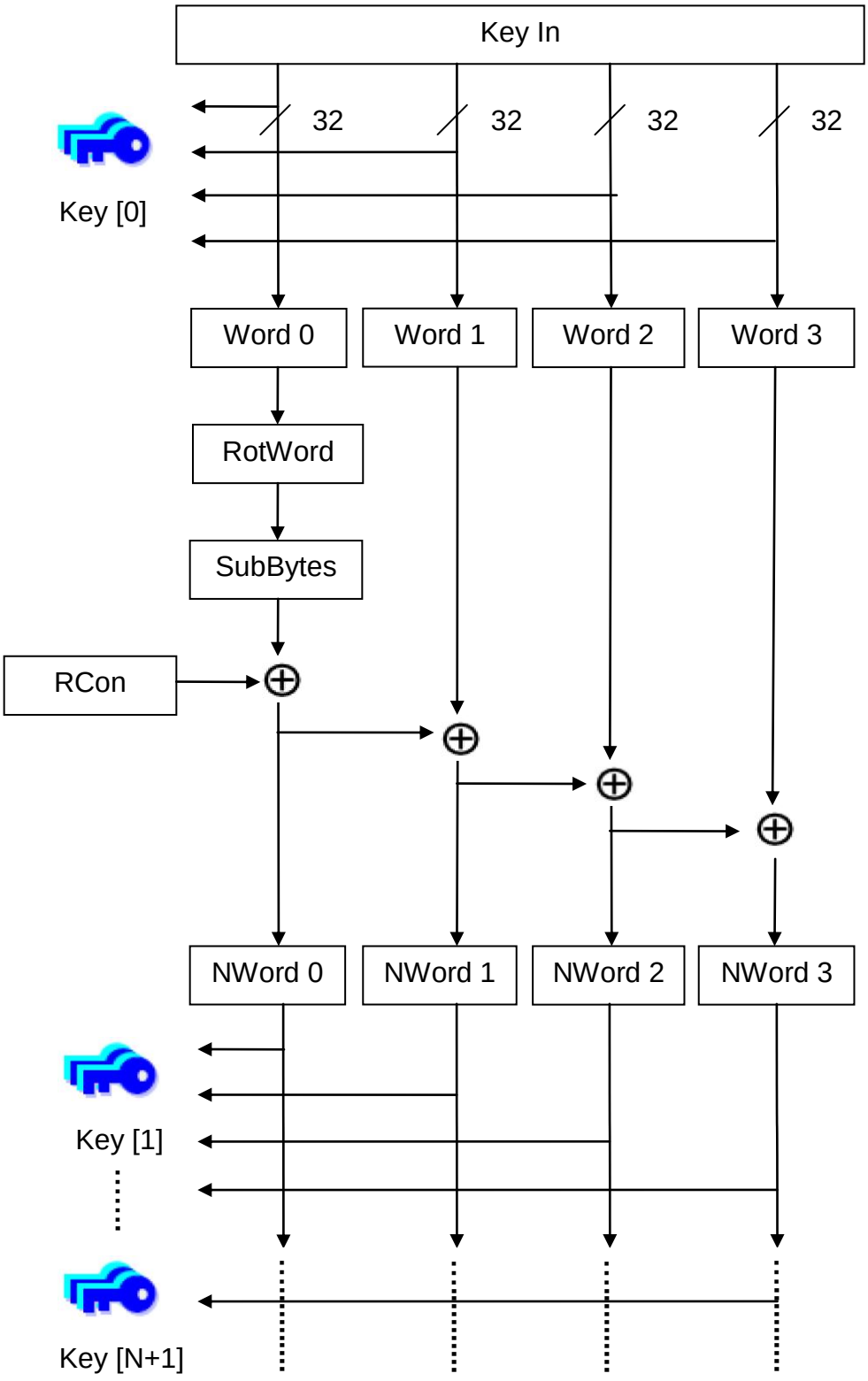


Figure 10: Key expander

3: AREA OPTIMIZATION

A variety of ASIC and FPGA implementations have been employed to improve the speed of AES. Currently, AES implemented as an ASIC can reach 40 Gbps by using 0.13 µm CMOS technology while AES implemented in an FPGA can reach 25 Gbps [16]. Despite the reusability feature brought by an FPGA, ASICs provide a higher throughput rate, use less lower power, and are cheaper in large volumes. Therefore, an ASIC implementation is more preferred when speed and area optimizations are desired. The AES cipher involves repetitive round operations. Within each round operation, SubBytes has been shown to be the performance-limiting step, being several times slower than the other three steps. In general, SubBytes operation occupies half of the total delay time and 20% of the circuit area [14]. The objective of this thesis was to design custom application specific circuits and to focus on area-efficiency for the SubBytes operation, S-BOX, to improve the overall encryption algorithm performance. Custom designs have been simulated. Area, speed and power have been measured to verify the improvement brought by the custom designs.

3.1 S-BOX

An S-BOX is a multiplicative inversion on a Galois field GF (2^8) followed by an affine transformation as mentioned in Section 2.2.1 [17]. Directly applying GF

arithmetic logic operations yields the lowest gate count S-BOX implementation [14]. However, it has a long critical path of XOR gates, which makes it slower. Several ASIC and FPGA methods have been employed to improve the speed of the S-BOX logic functions. Examples of some algorithms with very high throughput rates are stated in the following sections.

3.1.1 Twisted-BDD Architecture

BDD S-BOX architecture was shown to have a similar performance to a table look-up architecture [17]. A Twisted-BDD architecture was designed to solve the large fan-out and long propagation delay at the output through serially connected selectors inside the BDD architecture. It utilized a modified S-BOX, namely T-BOX, to minimize the delay of the MixColumns. Moreover, it contained eight BDDs arranged in parallel where each BDD corresponds to one primary output so that no node was shared between each BDD. These modifications significantly reduced the large fan-out and allowed the new architecture to reach a high throughput. A twisted-BDD S-BOX architecture was capable of running at 780 MHz based on a 0.13 µm technology and achieving 10 Gbps throughput in all encryption modes. However, this architecture simply read the truth table for each output bit without simplifying the tables first. Thus, an instant improvement could be achieved if the tables were simplified in the first stage.

3.1.2 FPGA

A fully pipelined AES Algorithm using an FPGA was another way to achieve high throughput rate. It was shown to provide a throughput rate of 21.54 Gbps [18]. In this algorithm, the ShiftRows step was done by interconnection and the key addition was done by XORing the round data and the round key. The MixColumns step consisted of a chain of XORs to permute the elements of data in each column. The arithmetic of these three stages could be combined in one pipeline stage for each round. During the byte substitution phase, the input was mapped into the GF (2^4) elements and the GF (2^4) operations were used. Moreover, this design resulted in an area efficient implementation of S-BOXes.

3.2 T-BOX

The T-Box algorithm is an improved version of S-Box [19] where SubBytes, ShiftRows, and MixColumns are combined together. The combined matrix is shown in Figure 11. Like the S-BOX, look-up tables for T-BOX can also be generated by substituting all possible inputs. Instead of storing only the value of SubBytes ($S_{i,j}$) in the S-BOX approach, the T-BOX approach stores values of SubBytes ($S_{i,j}$), 2*SubBytes ($S_{i,j}$), and 3*SubBytes ($S_{i,j}$). Each T-BOX has three 8-bit outputs and can be expressed as Figure 12. T_1-BOX is the same as the S-BOX .The Matrix in Figure 11 can now be re-written as the one in Figure 13.

$$\begin{bmatrix} S'_{0,C} \\ S'_{1,C} \\ S'_{2,C} \\ S'_{3,C} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} SubBytes(S_{0,C}) \\ SubBytes(S_{1,C+1}) \\ SubBytes(S_{2,C+2}) \\ SubBytes(S_{3,C+3}) \end{bmatrix}$$

Figure 11: Combined matrix

$$T(S_{i,j}) = \begin{bmatrix} T_1(S_{i,j}) \\ T_2(S_{i,j}) \\ T_3(S_{i,j}) \end{bmatrix} = \begin{bmatrix} SubBytes(S_{i,j}) \\ 2 * SubBytes(S_{i,j}) \\ 3 * SubBytes(S_{i,j}) \end{bmatrix}$$

Figure 12: T-BOX matrix

$$\begin{bmatrix} S'_{0,C} \\ S'_{1,C} \\ S'_{2,C} \\ S'_{3,C} \end{bmatrix} = \begin{bmatrix} T_2(S_{0,C}) \oplus T_3(S_{1,C+1}) \oplus T_1(S_{2,C+2}) \oplus T_1(S_{3,C+3}) \\ T_1(S_{0,C}) \oplus T_2(S_{1,C+1}) \oplus T_3(S_{2,C+2}) \oplus T_1(S_{3,C+3}) \\ T_1(S_{0,C}) \oplus T_1(S_{1,C+1}) \oplus T_2(S_{2,C+2}) \oplus T_3(S_{3,C+3}) \\ T_3(S_{0,C}) \oplus T_1(S_{1,C+1}) \oplus T_1(S_{2,C+2}) \oplus T_2(S_{3,C+3}) \end{bmatrix}$$

Figure 13: Re-written form of T-BOX

The T-BOX approach is roughly 1.7 times faster than the S-BOX approach [19]. However, each S-BOX corresponds to 3 T-BOXes so three times more area is required for the T-BOX approach. One-stage pipeline algorithms for the S-BOX and T-BOX are shown in Figure 14.

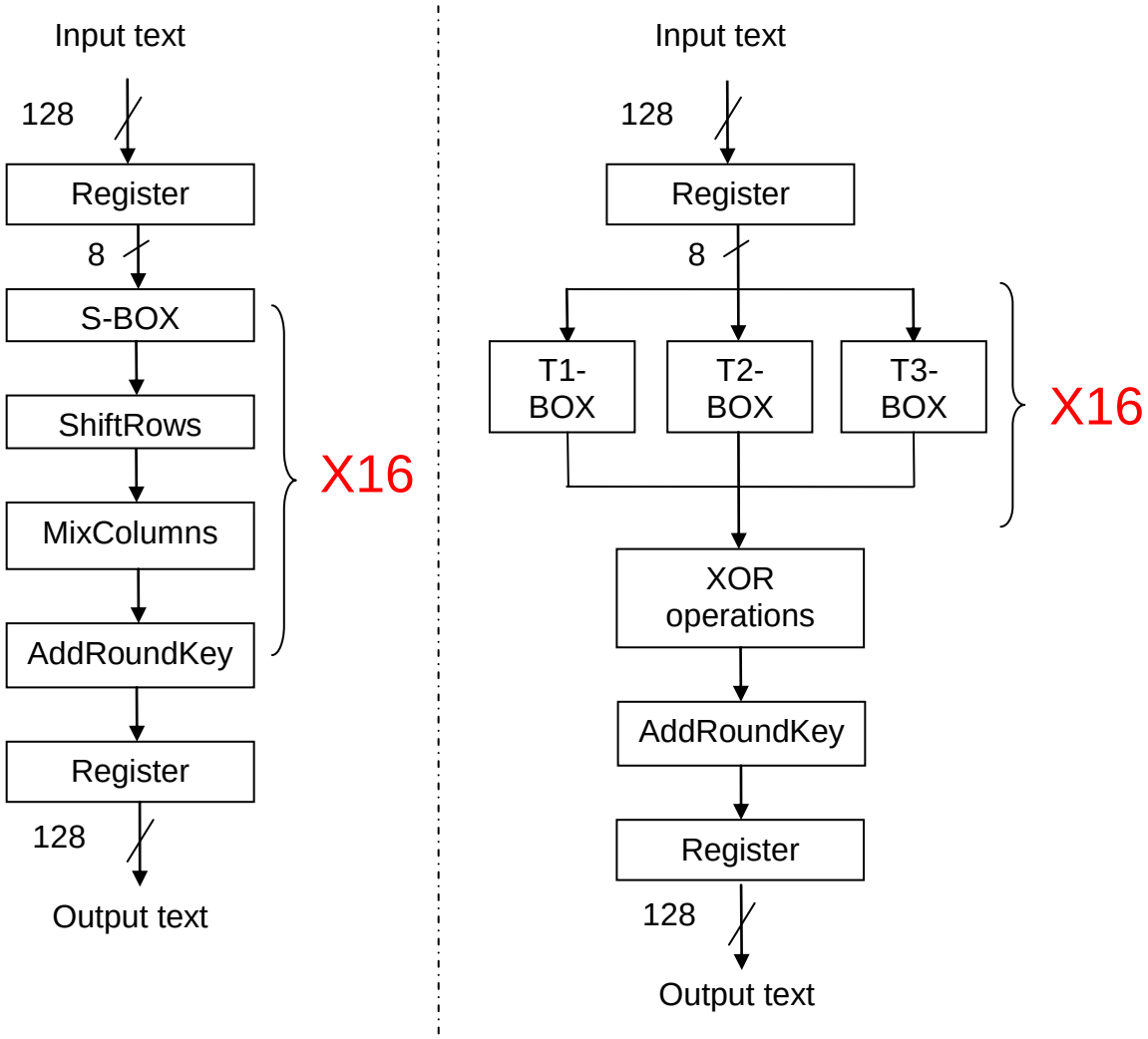


Figure 14: S-BOX and T-BOX

3.3 G-BOX

S-BOX and T-BOXes have similar properties since T-BOXes are derived from the S-BOX [20]. Thus, a generic box (G-BOX) that can be any box is formed. Custom designs are implemented based on the G-BOX. In this thesis, S-BOX (T_1-BOX) is chosen as the G-BOX.

3.4 Area-Efficient G-BOX Architecture

Even though the twisted-BDD architecture resulted in a very high throughput, it required 2818 NAND-equivalent gates, twice the number of the NAND-equivalent gates used in BDD or table look-up algorithms. Hobson et al. came up with a new algorithm that significantly reduced the number of NAND-equivalent gates used to only 738 NAND-equivalent gates [20]. The algorithm was then simulated by Hspice and the result was remarkable: the delay of the newly proposed algorithm and the twisted-BDD algorithm was approximately the same but the area of the new algorithm was significantly reduced. In the new algorithm, two methods were implemented: logic minimization and decoding, and wire-ORing.

3.4.1 Logic Minimization and Custom decoder

Based on the G-BOX look-up tables, truth tables for each output bit can be produced. Since there are XOR operations after T-BOXes, complement truth tables may be useful to generate both polarities of the output, and enable differential CMOS logic. However, the produced truth tables are long. For each output bit, there are 256 terms since the input has 8 bits. A desirable goal is to reduce the fan-out and fan-in to optimize the efficiency of the algorithm. The Berkeley *espresso* logic minimization program was used to further simplify the truth tables as much as possible. The typical reduced number of terms for each bit with its complement is listed in Table 2. The simplified and un-simplified truth table for bit0 is included in Appendix1. The remaining numbers of terms for each

bit are in a close range, with a maximum of 52. The average number of terms is 49.

Bit	Terms for True	Terms for complement
0	44	48
1	50	47
2	45	48
3	49	52
4	45	48
5	52	50
6	48	49
7	49	48

Table 2: G-BOX Minimization table

In order to reduce the fan-in of each term, the 8-bit input logic terms are split into two 4-bit input halves, namely, X and Y. The logic AND function of X and Y is performed as a wired-or of all terms, which is discussed in Section 3.4.2. There are several possible outcomes for X and Y. X and Y can be “All care” meaning xxxx, “one don’t care” meaning xxx-, xx-x, x-xx, -xxx (where “-“ means don’t care”), “two don’t care” meaning xx--, x—x, --xx, x-x-, -x-x, -xx- and finally “three don’t care” meaning ---x, --x-, -x--, x---. A C program was written to gather statistics about the various types of logic terms left after minimization [20]. Table

3 shows the summary of the statistics. The majority of X or Y is the “one don’t care” type and a symmetry between the true and complement table is observed.

Term	True	Comp	Type
xxxx	99	98	4-TERM
xxx-	202	217	3-TERM
xx--	80	73	2-TERM
x---	0	2	1-TERM
yyyy	130	121	4-TERM
yyy-	192	205	3-TERM
yy--	54	62	2-TERM
y---	6	2	1-TERM

Table 3: Aggregate Summary of term counts

The X, Y Terms could be implemented with 2, 3, or 4-input gates. However, a custom decoder based on the use of 2-input NAND, NOR gates and inverters should be the most efficient. The idea is to fully utilize the NAND2 gate. For example, inputs “10--“ and “101-“ have the same value at the first two bits so that the NAND2 gate for “10” at the first two bits can be shared. Thus, it will be more efficient to utilize NAND2 gates for 3-terms and 4–terms when they share the same inputs. The structures for different number of terms are depicted in

Figure 15. The output of the 2-term NAND gate, A, can be utilized to generate a 3-term or a 4-term value. However, the output, A, before outputting has to be inverted so that the final output is an AND operation.

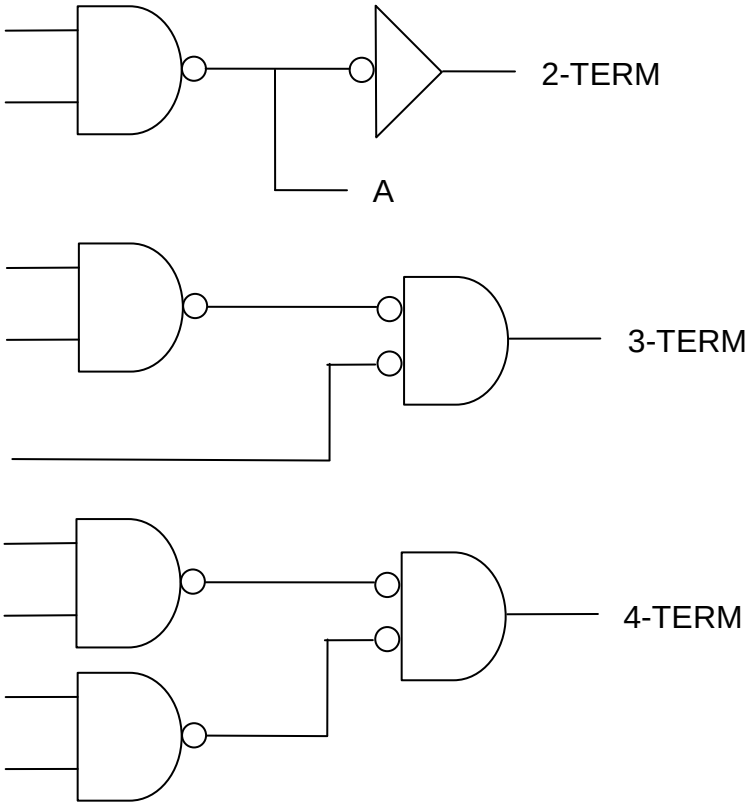


Figure 15: X/Y decoder logic structure

A hardware synthesis implementation requires the use of VHDL codes. Thus, truth tables have to be converted into VHDL codes. Due to the vast number of interconnections within the algorithm, a specified C++ code saves a lot of work and time. The Visual C++ code first creates internal signals for all possible input combinations, 81 for X and 81 for Y. Then, the code scanns

through each input line from each input file and chooses the corresponding internal signal for the particular input. These internal signals are used repeatedly so that fewer number of internal signals are needed to be declared. The result generated by the C++ code is the custom decoder that assigns correct values for X and Y terms.

3.4.2 Wire-ORing and Custom and_or circuit

A traditional way to implement the and-or logic functions is by the use of a PLA where inputs are first ANDed then ORed to produce the outputs. A PLA contains many AND and OR gates. A custom and_or circuit, which only contains N-MOSFETs is created so that P-MOSFETs within AND and OR gates can be taken away to reduce the area. The N-MOSFETs act as the pull-down circuit and there is a pull-up circuit cascading after the pull-down circuit to pull-up the signal.

The custom and_or circuit is based on the wire-oring architecture. The outputs of the custom decoder are either X or Y. X and Y are combined by the AND operation and all XY pairs are combined by the OR operation to form the and-or logic functions. A brief structure is displayed in Figure 16. A pair of XY devices shared the drain connection to help reduce output load. Based on the truth table, XY pairs can be on to connect the signal to ground for pull-down. The ground is connected to the source terminals of the 2-input NAND gates. More detail about the custom and_or circuit is discussed in Section 4.2.1.

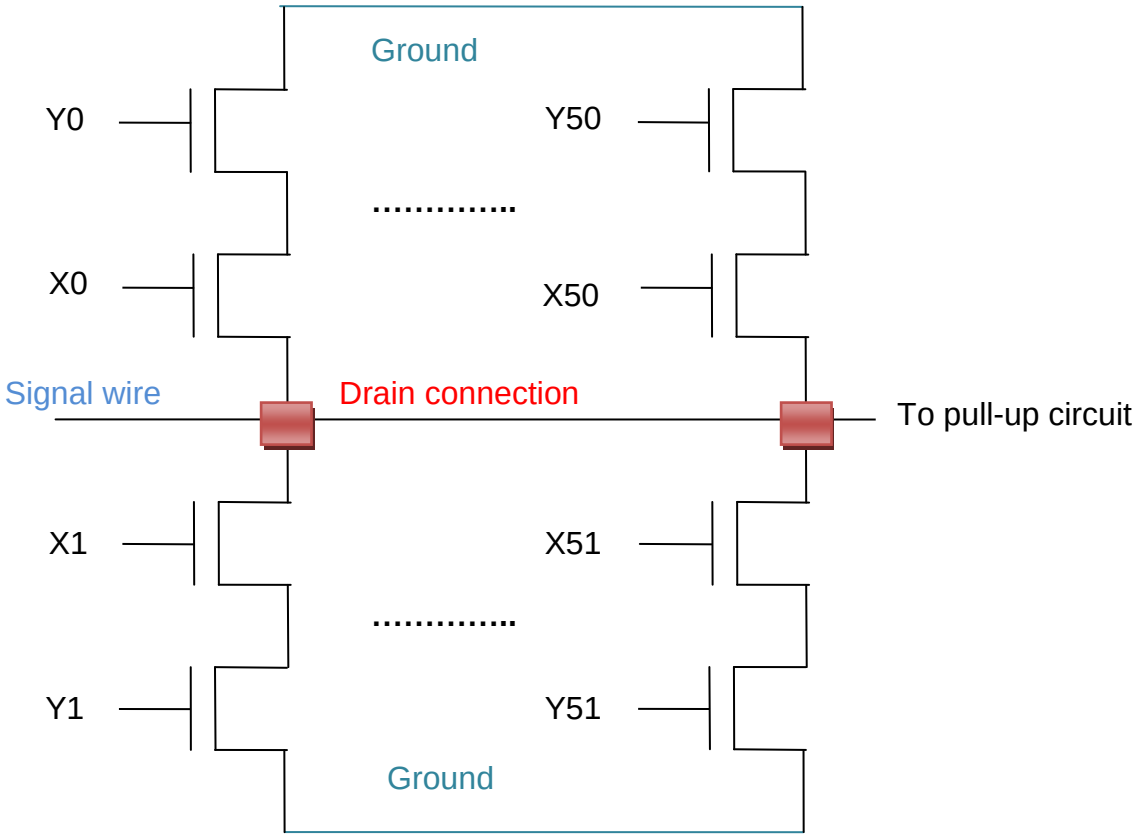


Figure 16: Wired-OR structure for an array of 2-input NAND gates (without PMOS pull-up)

4: DESIGN IMPLEMENTATION

In this thesis, actual automated layouts of eight custom and_or circuits corresponding to output bit0 to bit7 were created and verified. The technology was based on 0.18 µm CMOS since 0.13 µm CMOS was not available due to the lack of a cell library. In order to observe the improvement brought by the new custom algorithms, four different algorithms were implemented: *encryption_raw*, *encryption_de*, *encryption_ao_diff* and *encryption_ao_domino*. *Encryption_raw* was implemented by directly converting the simplified truth tables mentioned in Section 3.4.1 into VHDL codes. The entire *Encryption_raw* was then synthesized by a Synthesis Tool, Synopsys. *Encryption_raw* is set as the reference circuit.

Encryption_de was derived by adding the custom decoder into the *encryption_raw* algorithm. The remaining and_or section was then synthesized by Synopsys. These two algorithms only utilized standard cells from the Virtual Silicon cell library.

On the other hand, the two *encryption_ao* algorithms contain both standard cells and custom cells. The *encryption_ao* algorithms were implemented by inserting the custom and_or circuits into the *encryption_de* algorithm. Diff and Domino refer to the pull-up circuit used in the two algorithms.

Diff has a differential pull-up circuit while Domino has a clocked precharge for pull-up. Figure 17 summarizes the four algorithms.

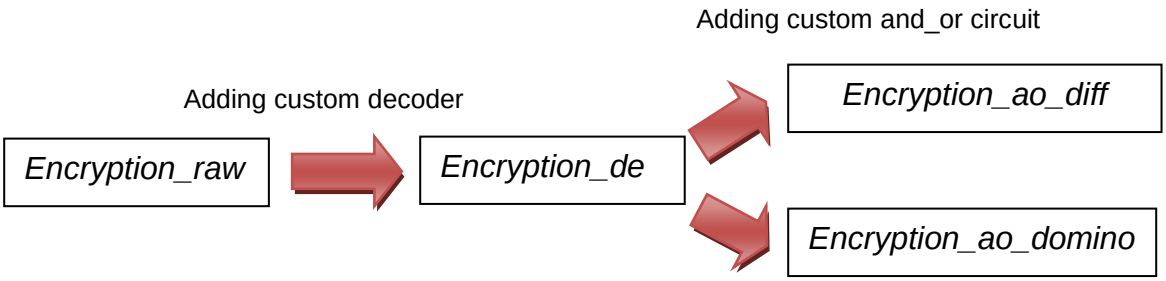


Figure 17: Summary of the three algorithms

In Figure 14, two registers were inserted to form a pipeline structure. All four algorithms share a similarity of architecture and are shown in Figure 18. The difference was the custom circuits. The custom decoder is embedded in the VHDL codes and is made of standard cells. The process of generating the custom and_or circuits is stated in Section 4.2.1. Other layouts such as registers, NAND gates, NOR gates and inverters already exist in the VS18SC library as standard cells.

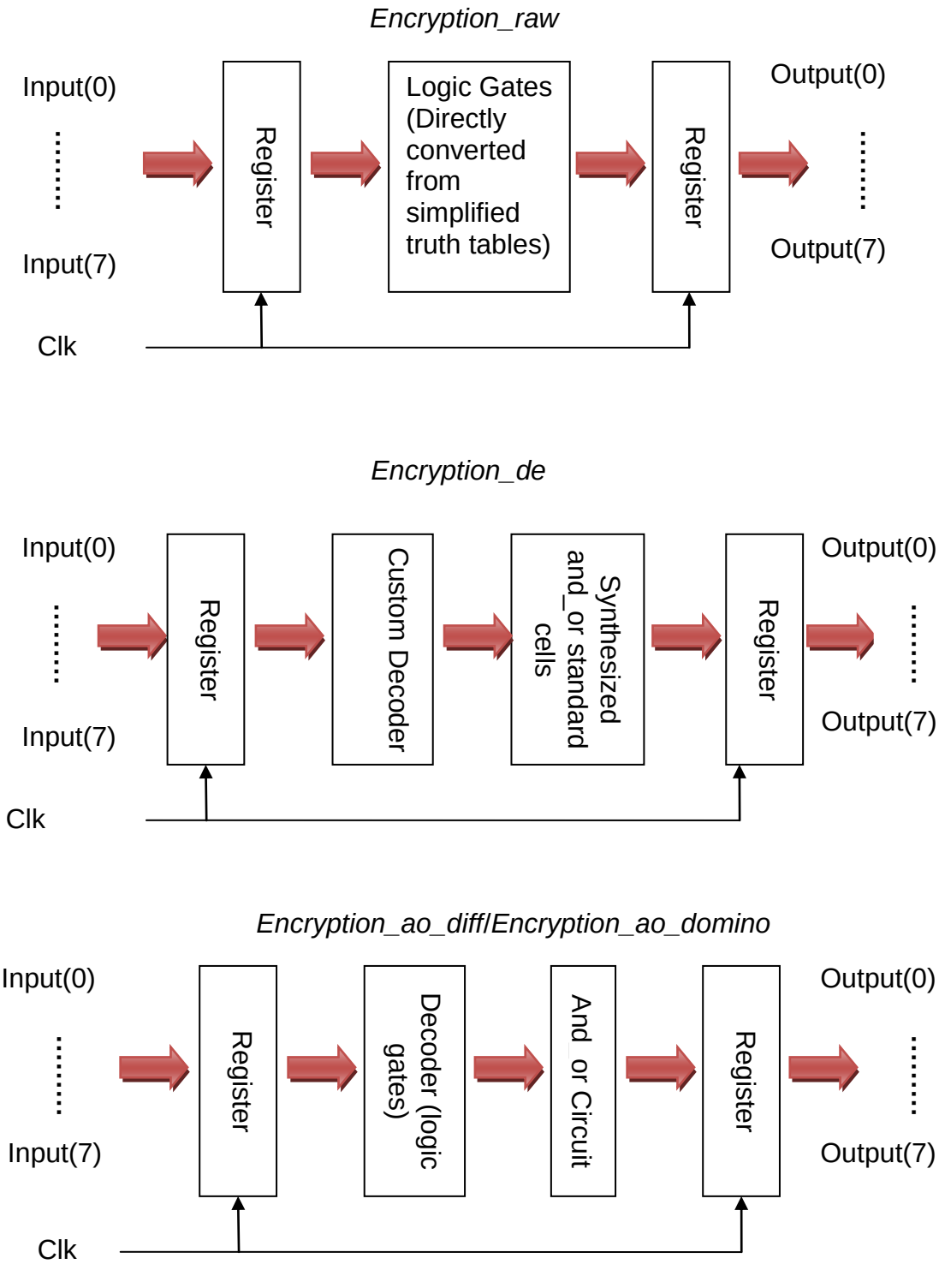


Figure 18: Comparison of architectures

Figure 19 demonstrates the design flow of all four algorithms. Source files containing the truth tables for each output bit were converted into VHDL code. This process was done using Visual C++. The C++ code is generic, meaning the code can be reused as long as the input files follow a particular format. The reusability helped save time on programming, modification and error fixing. The C++ and VHDL codes are attached in Appendix 2.

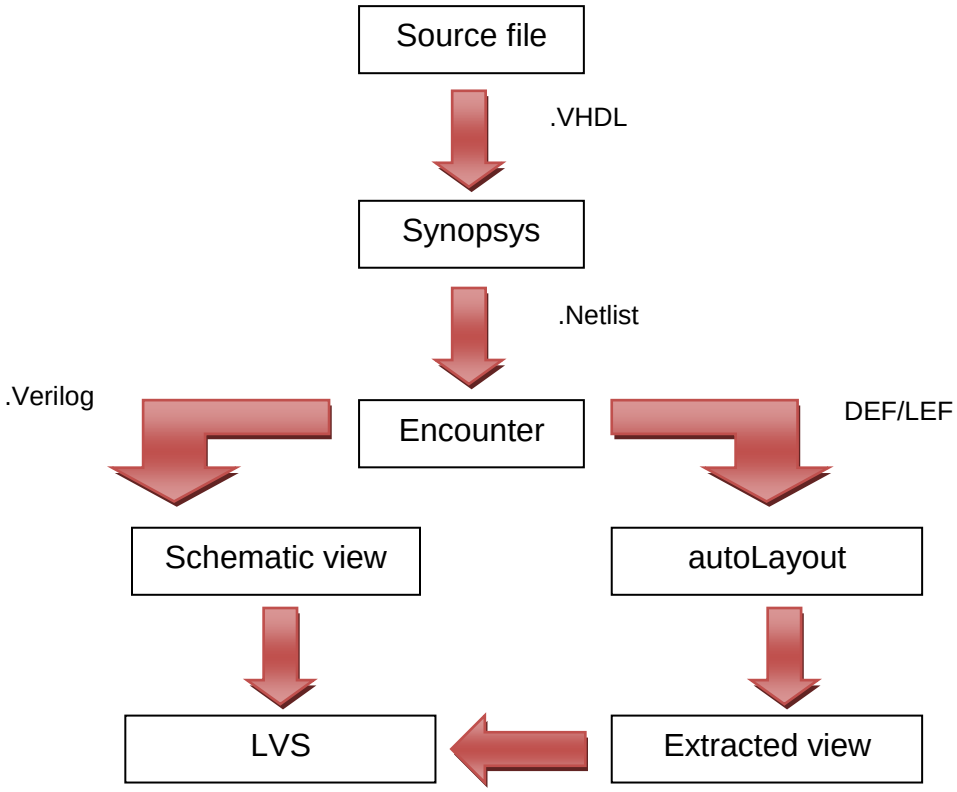


Figure 19: Flow of Design

Once VHDL codes were generated, they were then imported into the Synthesis tool, Synopsys. Synopsys synthesizes the entire algorithm and chooses standard cell sizes based on fan-out loading to achieve an optimization

of performance. A .LIB file containing timing information of each cell is required for a maximum optimization. Synopsys generates a Verilog netlist file containing the chosen cells and interconnection nets. Synopsys is also capable of checking if the time period specified in the SDC file can meet the design specification. A SDC file contains the information of the clock input. Synopsys also displays the critical path of the algorithm. However, Synopsys calculates the delay by adding up the delay of each standard cell and output RC delay without considering wire loads so that the result contains inaccuracy.

Another Synthesis tool called Encounter is used to verify if the physical layout of the entire algorithm is feasible. The user has to specify parameters of the chip such as the aspect ratio and expected layout density. Moreover, a LEF file has to be included so that Encounter can interconnect each pin properly. A LEF file is a simplified abstract gate representation and contains the size of a gate and coordinates of input and output pins for each component. In order to achieve an even more accurate and better performance, a TLF file is usually included since a TLF file contains timing models for each standard cell. Encounter reads the Verilog and LEF files and iteratively places and routes each cell in order to achieve the best performance according to given constraints, which can be added into the SDC file. If Encounter finishes successfully, then there is some guarantee that the design of the algorithm is feasible. Like Synopsys, Encounter can also check the time period specified in SDC file and displays the critical paths. Encounter outputs three files: a new Verilog file, which

can be imported into Cadence to generate the schematic view of the algorithm and DEF and LEF files, which can be imported into Cadence to produce an actual physical layout, namely, the autoLayout view.

The autoLayout view is further extracted to generate an extracted view of the algorithm. The extracted view can be chosen so that the wire capacitance is included. The last step, LVS, is to compare the schematic and extracted view to see if the netlists are matched.

4.1 *Encryption_raw*/*Encryption_de*

The architecture of *encryption_raw* and *encryption_de* is very similar since they only utilize standard cells from the VS18SC library. The main difference between the two is how C++ code interprets the source files. *Encryption_raw* was derived from C++ directly converting truth table into VHDL code. For example, “001--100 1” for output bit 0 is converted to:

If (input(7) = ‘0’ and input(6) = ‘0’ and input(5) = ‘1’ and input(2) = ‘1’ and input(1) = ‘0’ and input(0) = ‘0’) then

Output(0) <= ‘1’;

elseif….

On the other hand, *encryption_de* was generated by adding the custom decoder mentioned previously into the C++ code for *encryption_raw*. The rest of the flow is the same as Figure 19. Since *encryption_raw* and *encryption_de* are similar, only the results of *encryption_de* are displayed. Figure 20 contains the result of LVS. The schematic view is in Figure 21. An Encounter view with aspect ratio of 0.6 and row density of 0.6 is in Figure 22. Another Encounter view with different aspect ratio and row density is displayed later for comparison. Since the appearance of the Encounter view is the same as the autoLayout view, the autoLayout view is not shown.

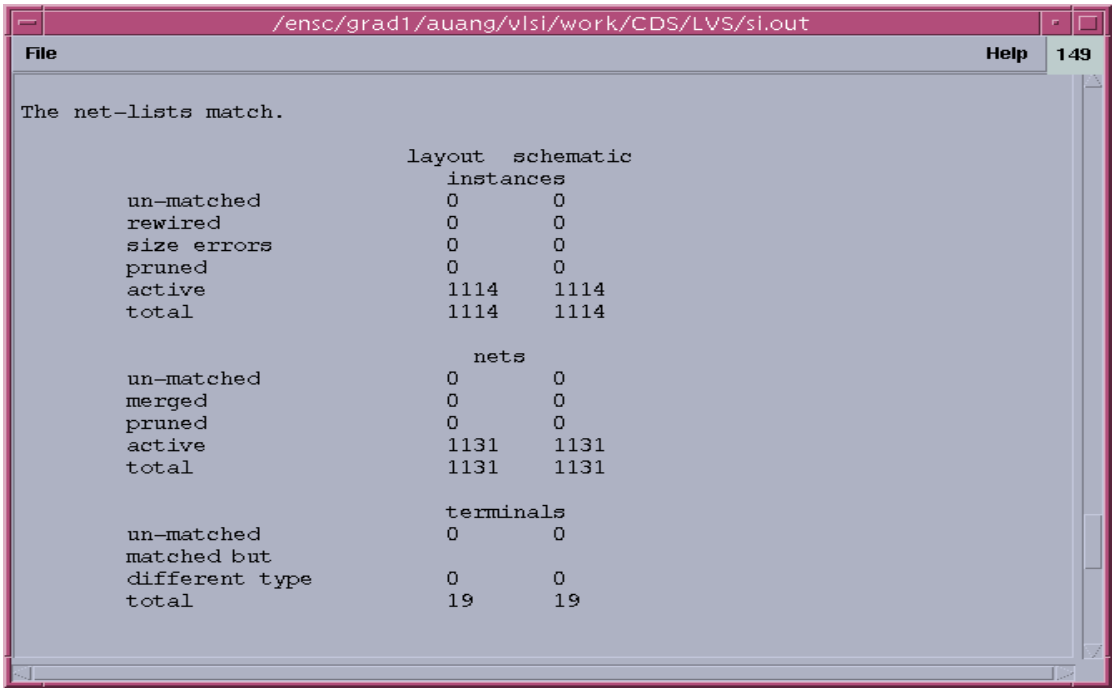


Figure 20: LVS of *Encryption_de*

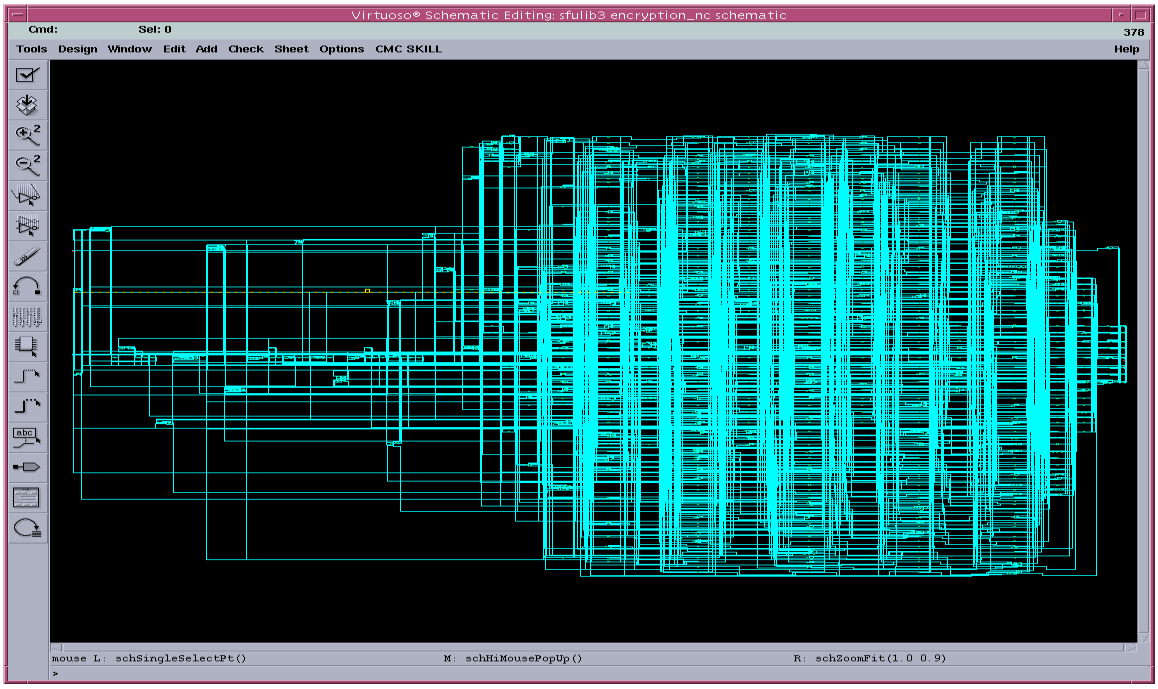


Figure 21: Schematic view of *Encryption_de*

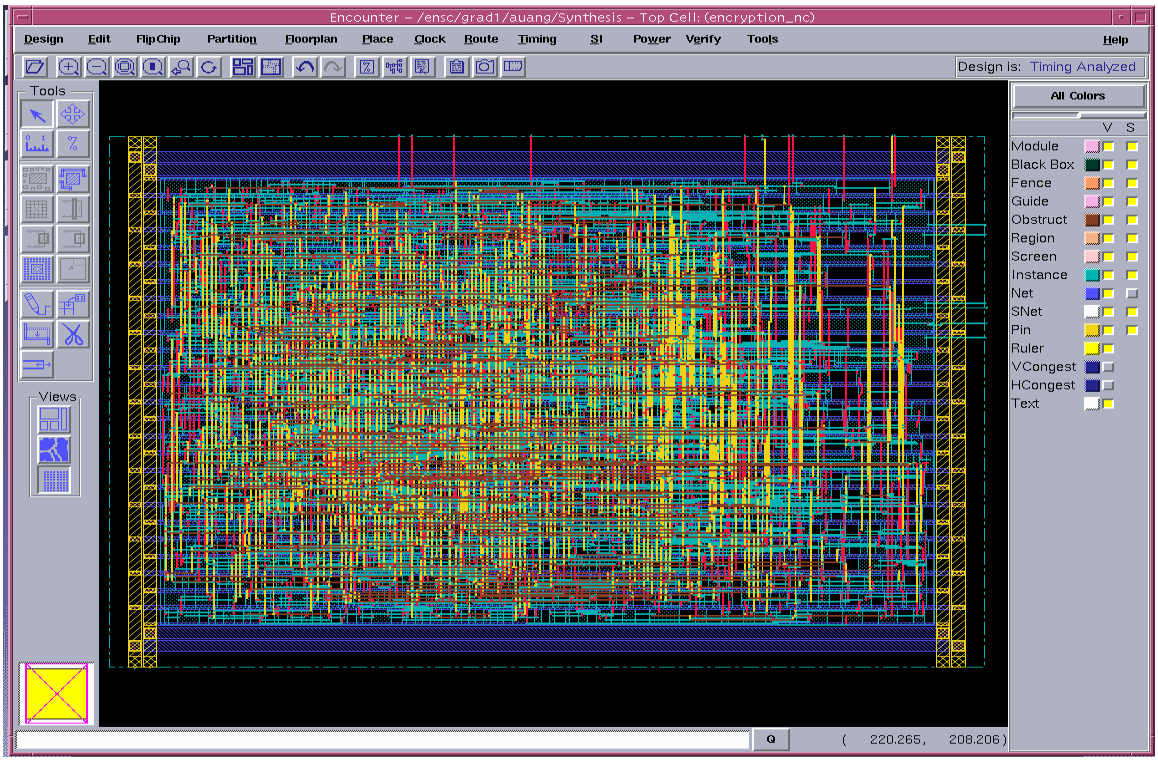


Figure 22: Result of Encounter

4.2 *Encryption_ao*

The *encryption_ao* algorithms contain the custom decoder and eight custom and_or circuits for each output bit. Eight VHDL codes are written to declare the eight and_or circuits. Each VHDL code only contains the entity part and an empty architecture. The eight and_or circuits and the custom decoder are connected by using port maps in the main VHDL code, “encryption.vhd”. VHDL codes are attached in Appendix 2.

4.2.1 And_or Circuit

Since custom and_or circuit layouts don’t exist in any cell library, layouts have to be designed and implemented for synthesis process. An important design goal of the custom and_or circuit is to let the Encounter execute “place and route” in a much easier and more efficient way. Thus, each and_or circuit is designed following the rule of the standard cell. A standard cell has a height of 11 YSP, the vertical grid pitch that is 6.16 µm. The width has to be a multiple of XSP, the horizontal grid pitch, which is 0.66 µm, and all pins within the cell have to be on the grid. Each grid point has to be a multiple of XSP and YSP. Following the same standard, cells can be cascaded between power and ground lines within a chip easily.

Each and_or circuit contains many 2-input NAND sub-circuits. Sub-circuits expand horizontally and since sub-circuits were designed following the rule of the

standard cell, the horizontally expanded layout also has a height of 11 YSP. A brief and_or circuit layout is shown in Figure 23.

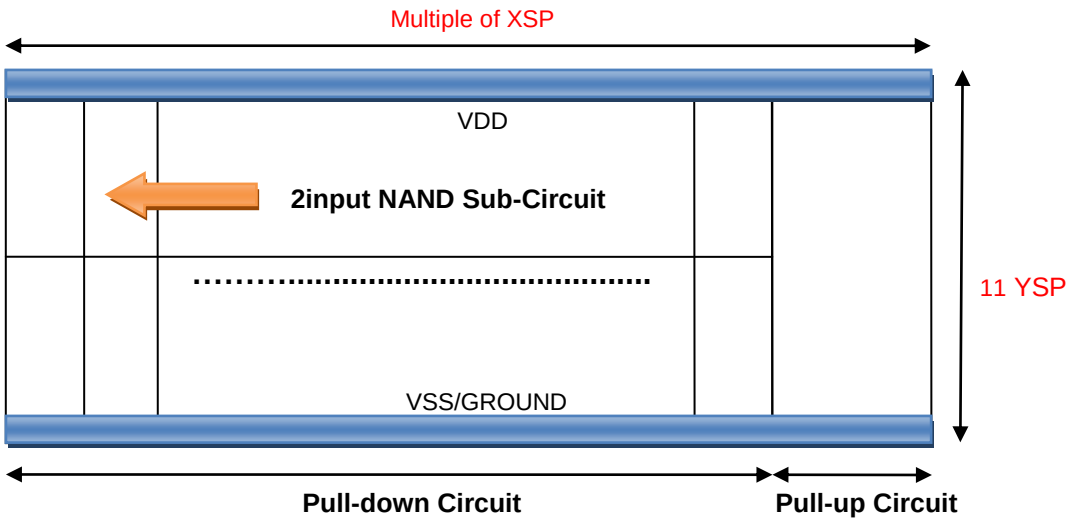


Figure 23: Overview of And_or circuit

The layout can be divided into 2 regions: one for the pull-up and the other one for the pull-down circuit as shown in Figure 23. The layout of the pull-down circuit is composed of layouts of 2-input NAND sub-circuits. The pull-up circuit can be either differential or domino circuit. The differential circuit possesses the advantage of generating both the true and complement output at the same time so that the XOR circuits after it can use the complement output directly. Thus, the differential circuit increases the flexibility of the whole encryption algorithm. Another advantage of the differential circuit is the voltage equalization, which is further discussed in Section 4.2.1.3. The domino circuit is expected to have a

smaller area since it only generates the true output. More detail of it is discussed in Section 4.2.1.4.

As mentioned previously, each 8-bit input is divided into a 4-bit X and a 4-bit Y. There are three possible input values, ‘1’, ‘0’ and ‘–‘ (“don’t care”), which makes the 4-bit X, 3^4, 81 possible combinations. An index number is given to each X combination. In each and_or circuit, input pins are placed from left to right with an increasing index. This design utilizes the property of the “or” function: the order of the inputs is not important as long as X is connected to the correct Y at the end. For example, the logic function of “(A*B) + (C*D) + (E*F)” is equal to “(C*D) + (E*F) + (A*B)”. The advantage of this design is that X inputs within different and_or circuits can be routed vertically straight. An efficient routing is made possible by designs, and it was observed that Encounter took advantage of it. Figure 24 shows how X indexes are distributed among and_or circuits.

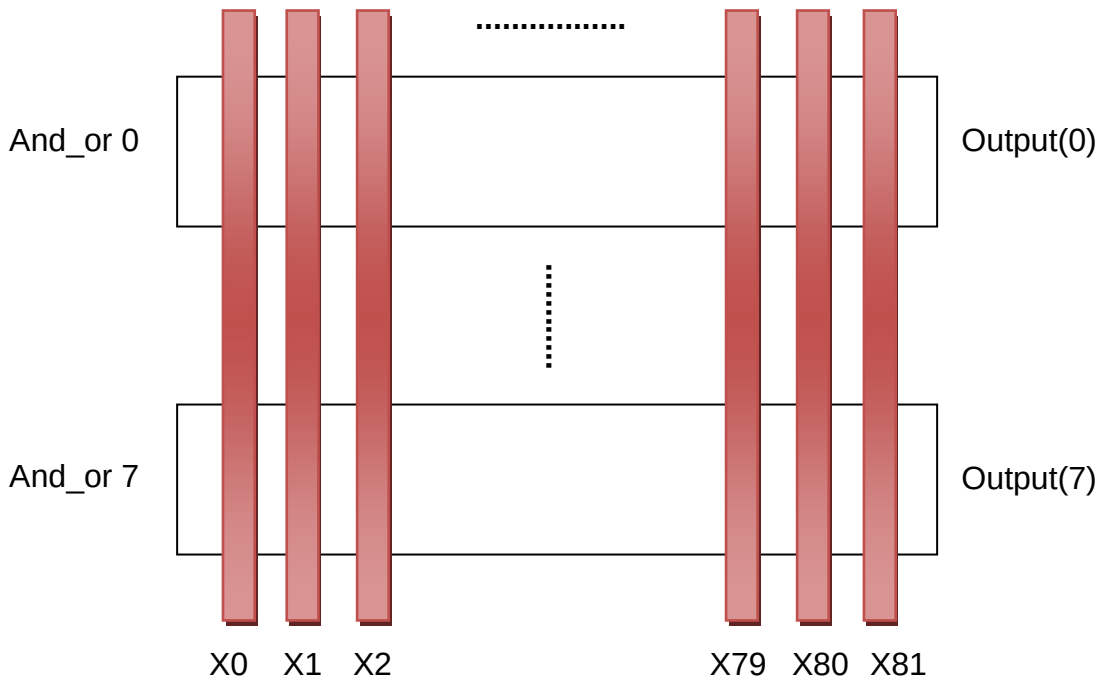


Figure 24: X indexes

There are six kinds of sub-circuits. They can be described as Normal, Special and Empty type. Each type has two different forms; one is for the true output that is located at the top and the other one is for the complement output that is located at the bottom. The complement ones are only utilized when the pull-up circuit is differential circuit. Sub-circuits were all made following the rule of standard cells. More detail of sub-circuits is discussed in Section 4.2.1.2.

A generic C++ code was written to produce the layout for each and_or circuit. It reads truth tables from input files and selects the corresponding sub-circuit to form Skill codes. The Skill code could be imported into Cadence to generate the actual layout. The layouts have to be checked against design rules

by DRC and verified to produce abstract views for Encounter synthesis. The flow of and_or circuit layout is shown in Figure 25.

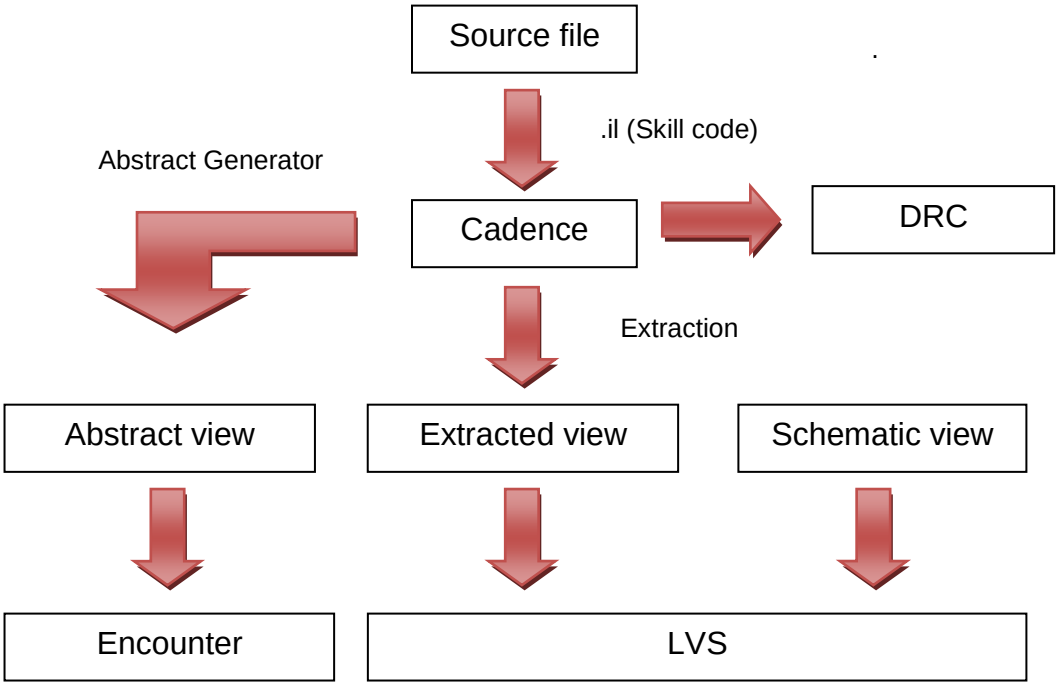


Figure 25: Design flow of and_or circuits

4.2.1.1 Reordering

Since the generic C++ code reads truth tables in input files from top to bottom, reordering inputs based on the value of the first four bits, the X part, allows the C++ code to read inputs and to locate the X input pins at an increasing order in a more efficient way. The code uses a base 3 algorithm and treats '-' as 2, '1' as 1 and '0' as 0. For example, "-010" means 57 ($2*3^3 + 0*3^2 + 1*3^1 + 0*3^0 = 57$). The code then outputs an ordered version of the input file, which can then

be executed to generate the IL files for layout. IL is the file type name for Skill code. The reordering C++ code is attached in Appendix 2.

4.2.1.2 Sub-circuit

In order to minimize the input capacitance, the X input of the Top and Bottom sub-circuit share the same input pin. The value of the X input determines the name of the input pin and its location. The naming convention uses a base 3 algorithm as described in Section 4.2.1.1. On the other hand, the locations of Y input pins do not depend on the value of Y. The naming of Y input pins for true output starts with Y0 and the naming of Y input pins for complement output starts with Y52 since the maximum number of inputs among all true and complement output is 52. The naming of Y increases from the left-most sub-circuit to the right-most sub-circuit. Figure 26 contains the layouts of a Normal Top and Bottom sub-circuits. The combined layout has a height of 11 YSP and a width of 2 XSP.

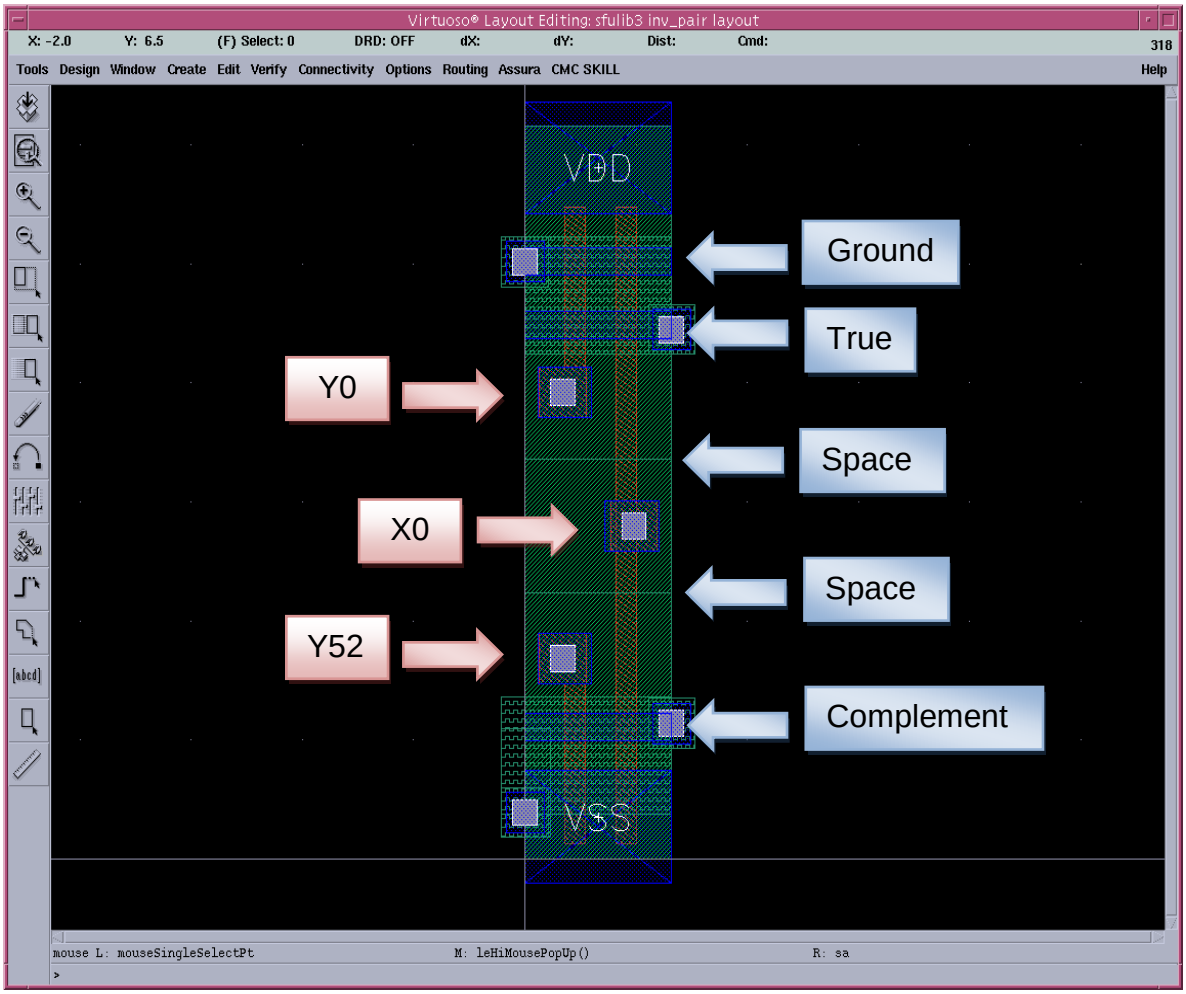


Figure 26: Layouts for Normal Top/Bottom sub-circuits

The X and Y input pins are designed to have enough vertical space for a horizontal metal wire to pass through so that Encounter has more freedom for interconnection routings.

However, there exists a dilemma in this design. Inputs with the same X value could occur more than once. Since each X input has its own location, those repeated X value inputs cannot simply be located right beside the first occurring

one. These repeated inputs have to be placed somewhere else in the and_or circuit. Fortunately, thanks to property of random distribution of the S-BOX, many locations within the and_or circuit remain empty. In the generic C++ code, there is a function that searches for the closest empty space around the first occurring sub-circuit for other repeated inputs.

These repeated sub-circuits are named sTop and sBot where s refers to the Special type. Depending on the surrounding of the first occurring sub-circuit, two methods for connecting the repeated input pin to the first occurring input pin are created. In both methods, the vertical space between the X and Y input pins is utilized.

If there is an empty space right beside the first occurring sub-circuit, the repeated sub-circuit is placed right at the location. A poly wire through the prepared space connects the X inputs of the sub-circuits. An example is shown in Figure 27.

If there is an obstacle between the first occurring sub-circuit and the repeated one, a horizontal metal1 wire crossing through the obstacle sub-circuit is used to connect the input pins. An instance is shown in Figure 28.

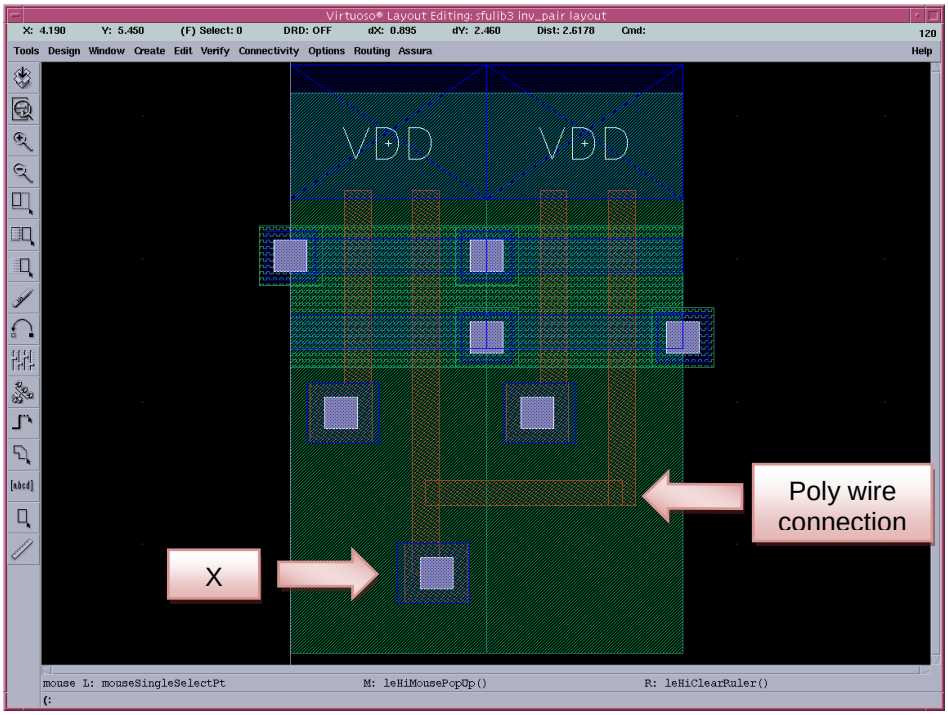


Figure 27: Special type I

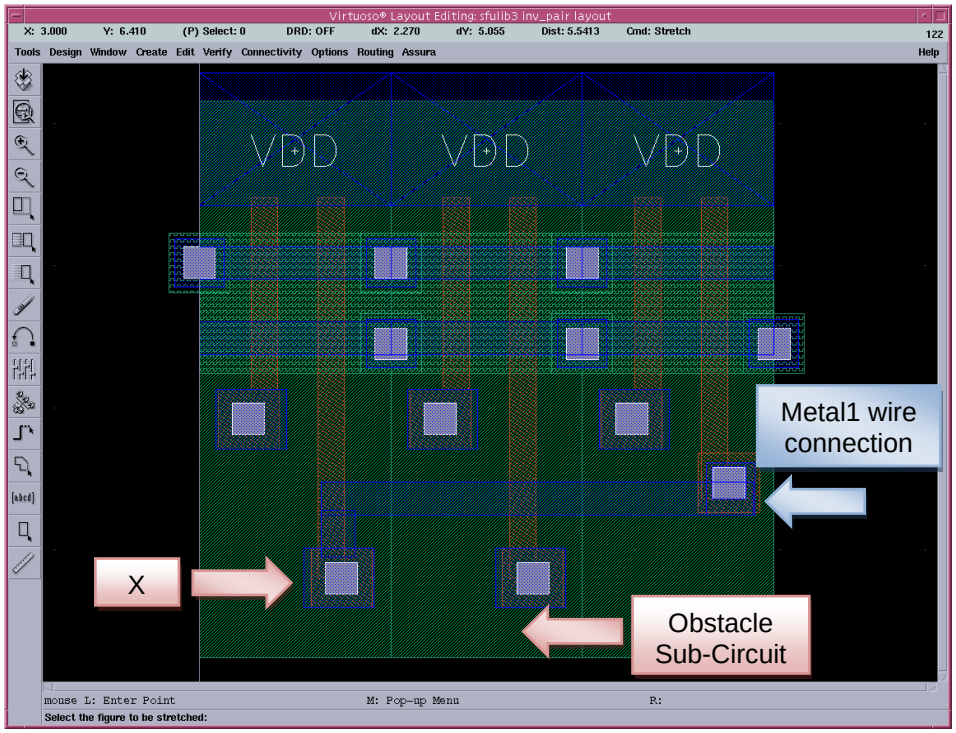


Figure 28: Special type II

An empty Top or Bottom sub-circuit is chosen when there is no Normal or Special type of sub-circuits at the location. The empty sub-circuit is very simple and only contains horizontal metal wire for continuous signal transmission. Empty Top and Bottom layouts are displayed in Figure 29.

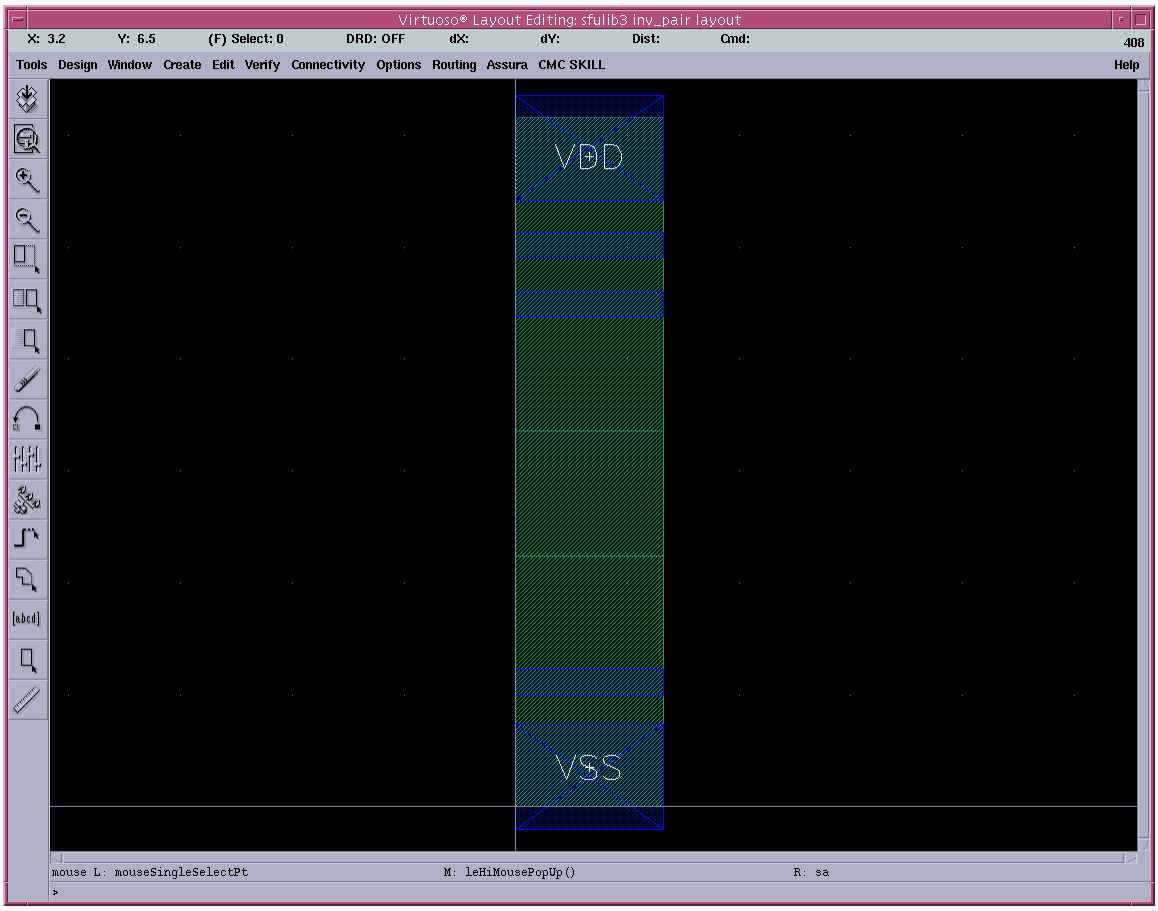


Figure 29: Layouts for Empty Top/Bottom sub-circuits

One of the DRC checking rule applied in the and_or circuit is “20micron_sub_check”, which means that a p-substrate contact has to be around NMOS transistors within 20 µm. Figure 30 depicts how p-substrates are

distributed through the and_or circuit and the layout of the p-substrate is displayed in Figure 31.

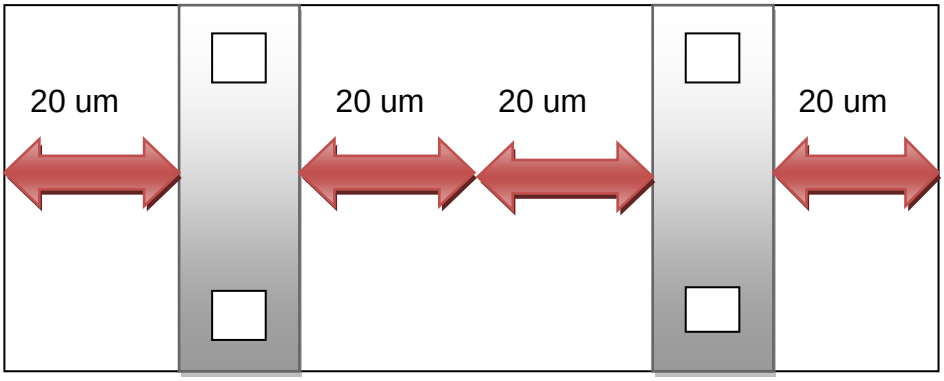


Figure 30: Cover region of P-substrate

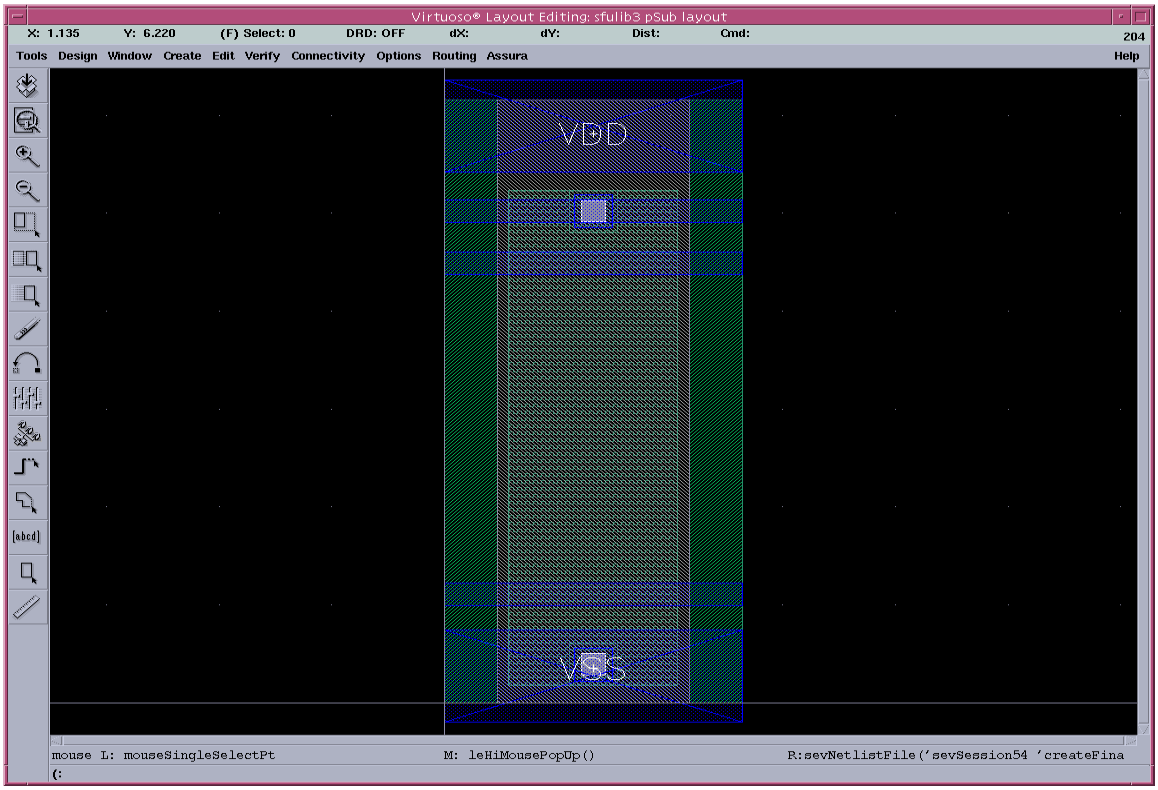


Figure 31: Layout of P-substrate

4.2.1.3 Differential circuit

The pull-up differential circuit is placed at the end of each and_or circuit. A differential circuit has the potential to be faster than the standard static CMOS logic gates and requires fewer transistors [21]. A schematic view is depicted in Figure 32.

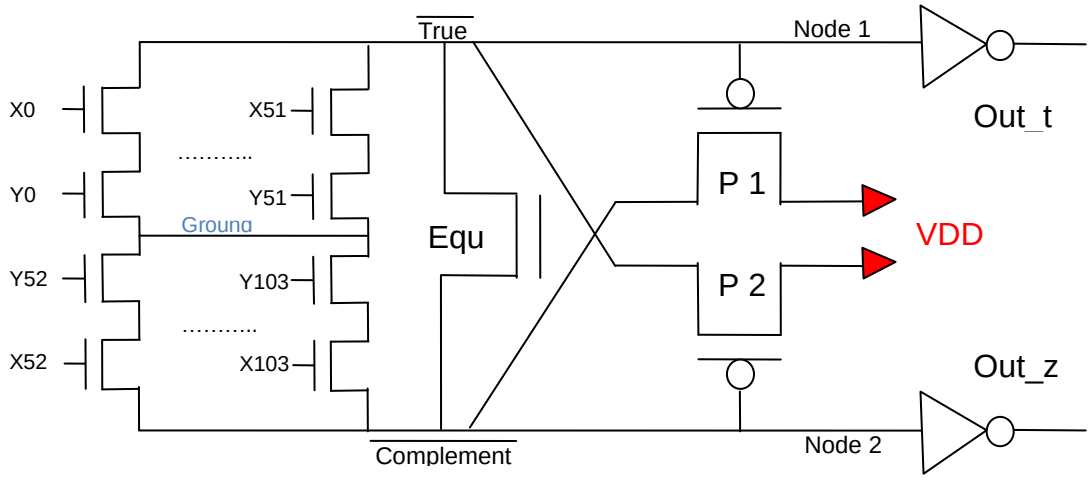


Figure 32: Schematic View of *encryption_ao_diff*

The differential circuit reads in the true and complement signals and produces a true output, out_t and a complement output, out_z. There is another input called “Equ” in this circuit. Its feature is to level up the true and complement signals when “Equ” is turned on. “Equ” is fed into the gate terminal of an NMOS while the drain and source terminals connect to the true and complement signals. “Equ” can simply connect to a clock-like signal. During the equalization phase, a channel between the true and complement signals occurs and the two outputs are undefined but their values are expected to approach half of VDD. Once the

“Equ” returns to its low state, the two outputs become valid again. Therefore, a full voltage swing from ‘0’ to ‘1’ or ‘1’ to ‘0’ at the outputs could be avoided since new outputs will swing from half of VDD. This feature speeds up the whole process by avoiding any full swing voltage change. The true and complement signals are guaranteed to be pulled toward each other since the two signals are always complement to each other. The feature of “Equ” is similar to the precharge phase of a Domino circuit.

A layout overview of and_or_diff and Cadence layout for pull-up differential circuit are shown in Figure 33 and Figure 34 respectively. Two signal wires are crossing horizontally through the layout in Figure 33. The top one is for the “True” signal and the bottom one is for the “Complement” signal. Each signal connects the outputs of the sub-circuits horizontally and passes them into the pull-up circuits. The transistor size in Figure 34 is 1 Abox that is 0.42 µm. Optimized sizes for transistors are discussed in Section 5.2.

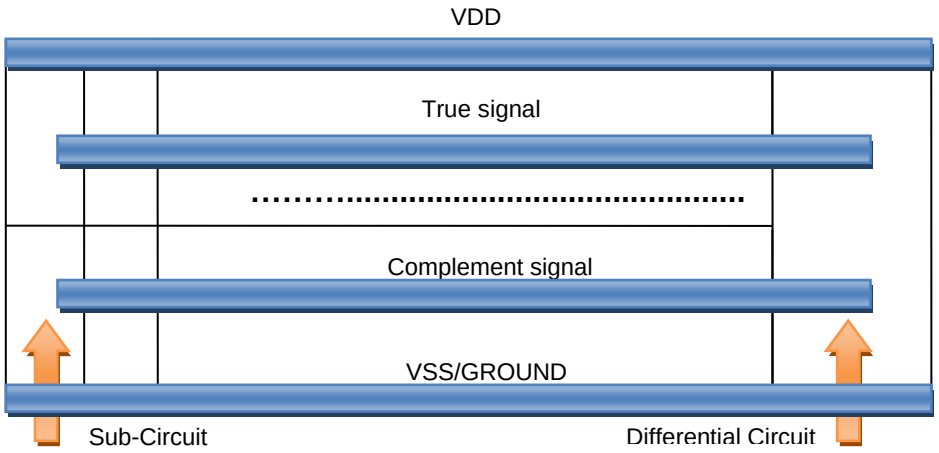


Figure 33: Layout overview of *encryption_ao_diff*

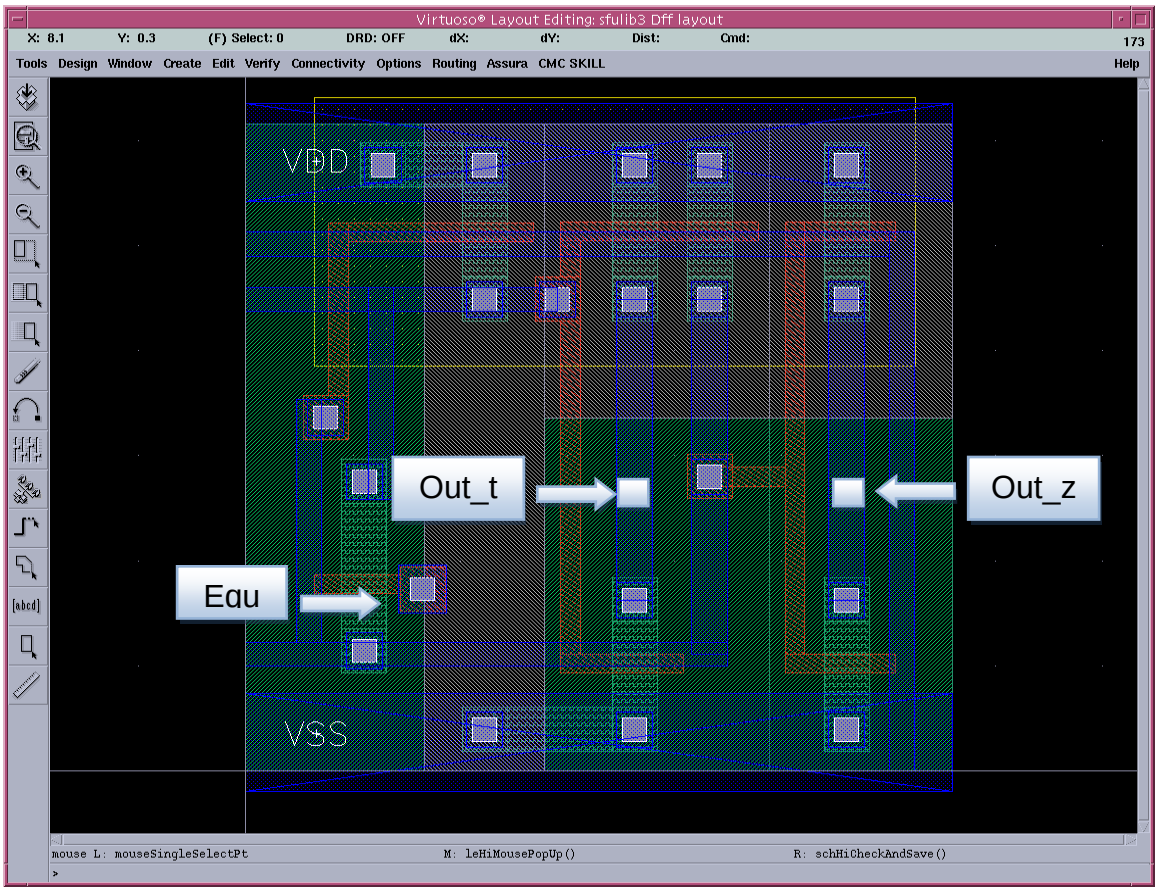


Figure 34: Layout of the differential pull-up circuit

4.2.1.4 Domino Circuit

The schematic view of the domino circuit is shown in Figure 35. The 2-input NAND gates work as the pull-down logic while “Equ” controls the period of the precharge and evaluation phases.

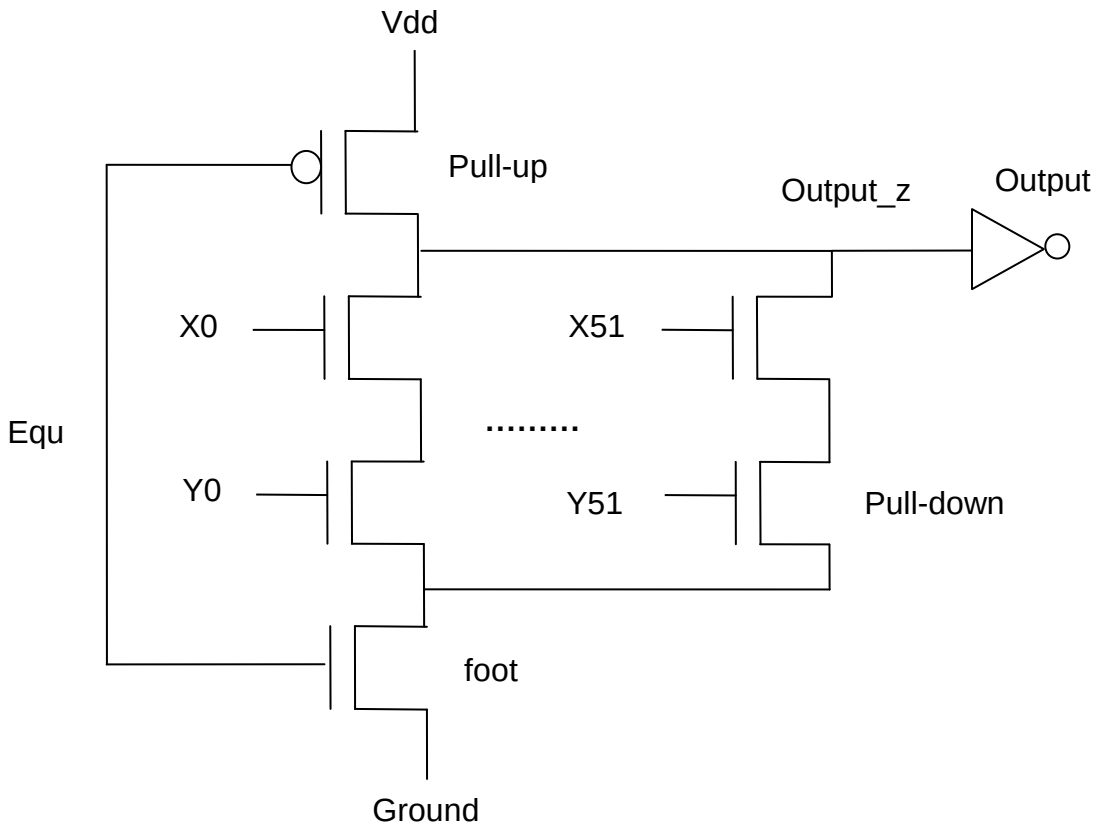


Figure 35: Schematic View of *encryption_ao_domino*

When “Equ” is ‘0’, the pull-up PMOS is turned on and output_z is charged up to VDD. Meanwhile, the foot NMOS is turned off so that output_z stays high no matter what inputs are fed into the pull-down circuits. The output is ‘0’ during the precharge phase. On the other hand, when “Equ” is ‘1’, the pull-up PMOS is off and the foot NMOS is on. The value of output_z now depends on the pull-down circuit. If two inputs in series are both on at the same time, then, there is a connection between output_z and ground and output_z is pull-down to ground. In this case, output becomes to ‘1’. The value of the output depends on the pull-down circuit during the evaluation phase.

When the domino circuit is used as the pull-up circuit at the end, sub-circuits for complement outputs are not included. Thus, the architecture of the and_or circuit is different from the one, which uses the differential circuit. An overview of the and_or_domino circuit is depicted in Figure 36 and a Cadence layout of the pull-up domino circuit is included in Figure 37. Like the differential circuit, the transistor size is also 1 Abox and optimized sizes for transistors are further discussed in Section 5.3.

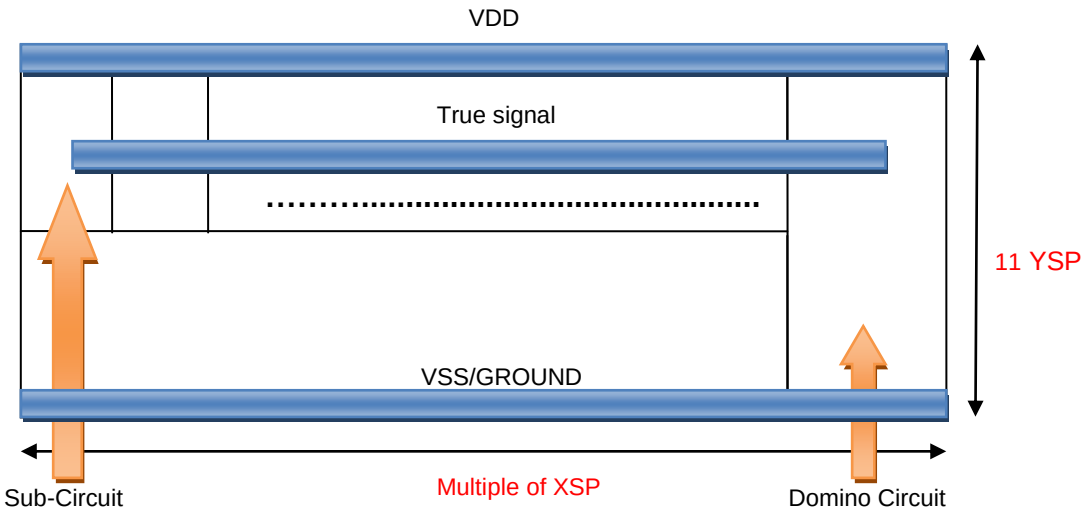


Figure 36: Layout overview of *encryption_ao_domino*

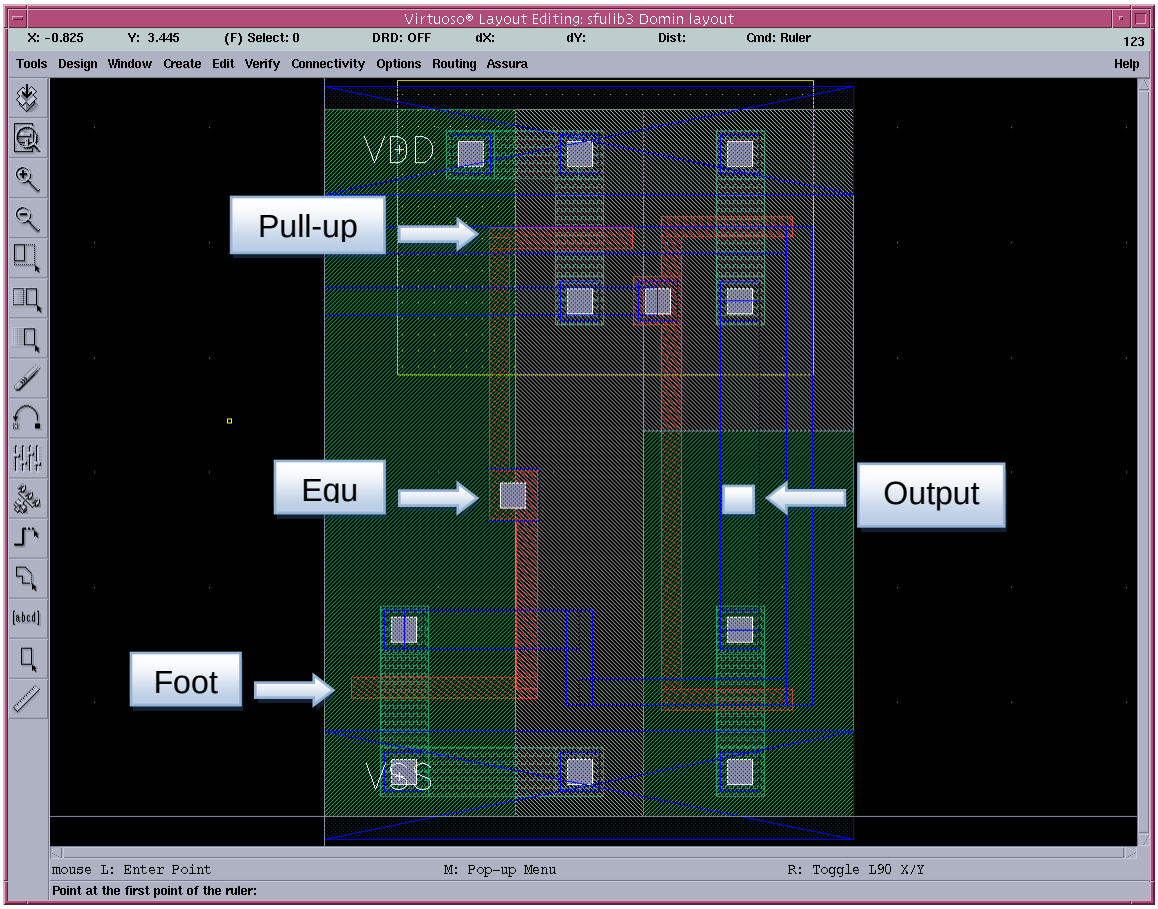


Figure 37: Layout of a domino pull-up circuit

4.2.1.5 DRC/LVS check

An and_or circuit is formed by putting all sub-circuits and pins at the right locations. All 8 and_or circuits follow the generic structure shown in Figure 23. The only difference is how pins and sub-circuits are distributed. Before the layout of each and_or circuit is ready for abstraction, it has to pass both the DRC and LVS checks. The schematic view of each and_or circuit is drawn by using Cadence Virtuoso. Another Cadence tool, abstract generator, is used to generate the abstract view of each and_or layout. A full version of the and_or_0 layout,

schematic view and abstract view are included in Appendix 3 where the number at the end of the name represents the index of the output bits. Figure 38 and Figure 39 shows the results of DRC and LVS for And_or_0 that contains the differential circuit.

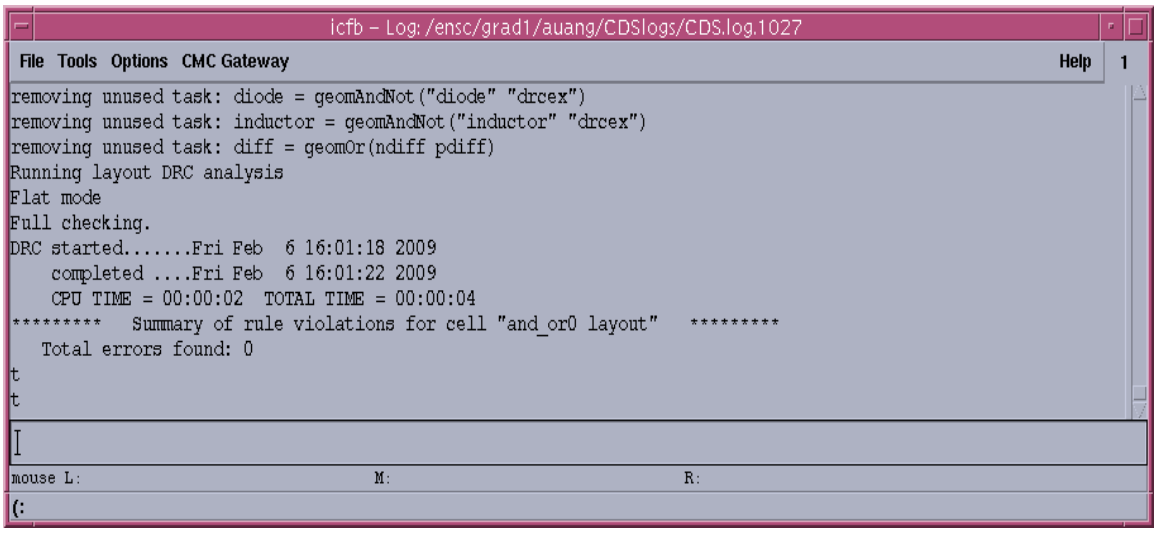


Figure 38: Results of DRC for and_or_0

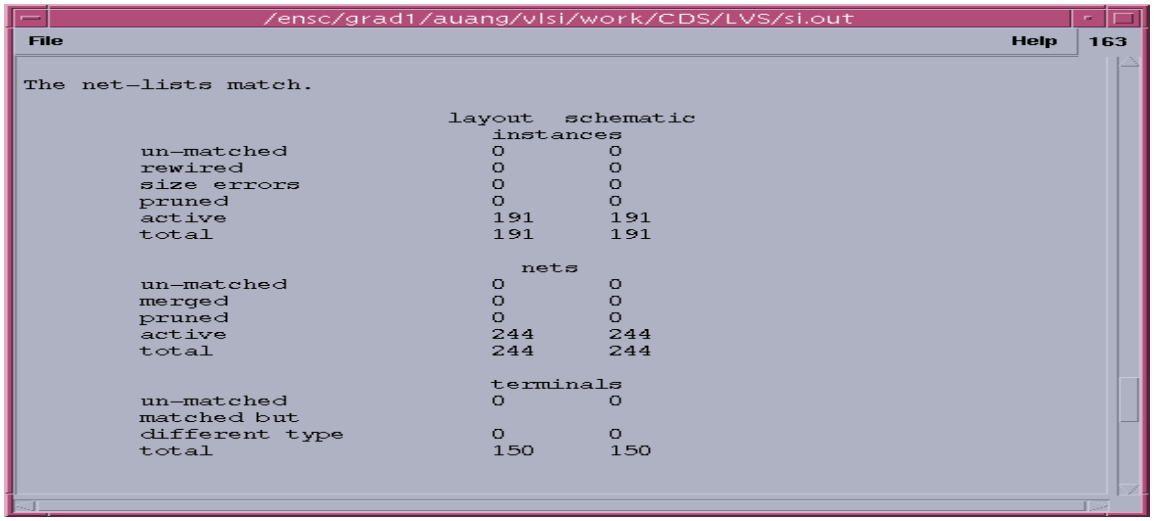


Figure 39: Results of LVS for and_or_0

Like *encryption_raw* and *encryption_de*, *encryption_ao* was also implemented by Synopsys and Encounter to generate its Verilog, LEF, DEF, schematics view, autoLayout and extracted view. The schematic view is shown in Figure 40 and the LVS result is shown in Figure 41.

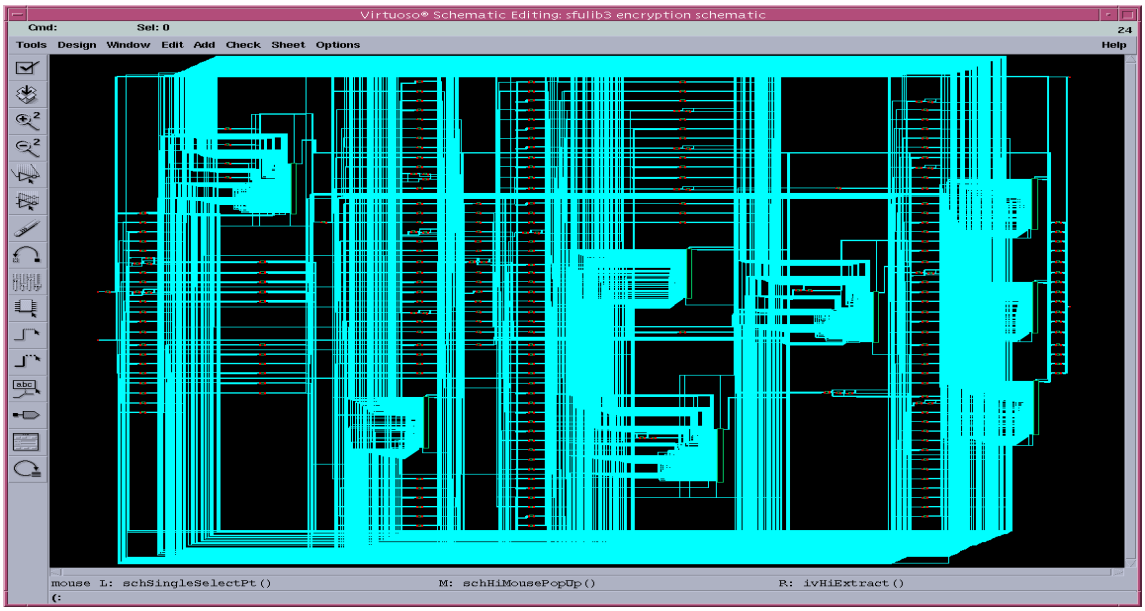


Figure 40: Schematic view of *encryption_ao*

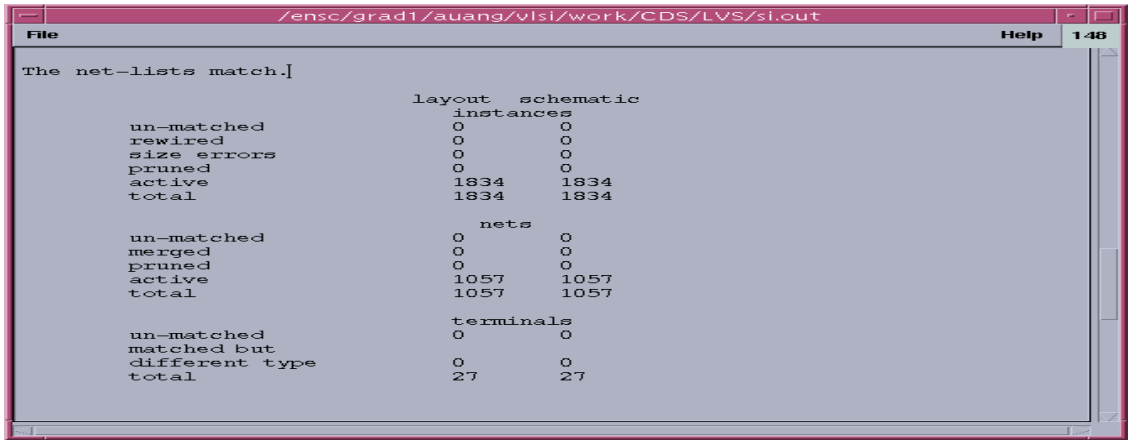


Figure 41: LVS for *encryption_ao*

Figure 42 and Figure 43 illustrate the Encounter views of *encryption_ao_diff*. It was derived with an aspect ratio of 0.9 and row density of 0.94. Figure 43 clearly shows that the eight and_or circuits are packed tightly within the chip and the algorithm meets the expectation described in Figure 24. Several aspect ratios were tried to achieve the highest row density and 0.9 is the highest value for the aspect ratio to achieve a maximum density of 0.94.

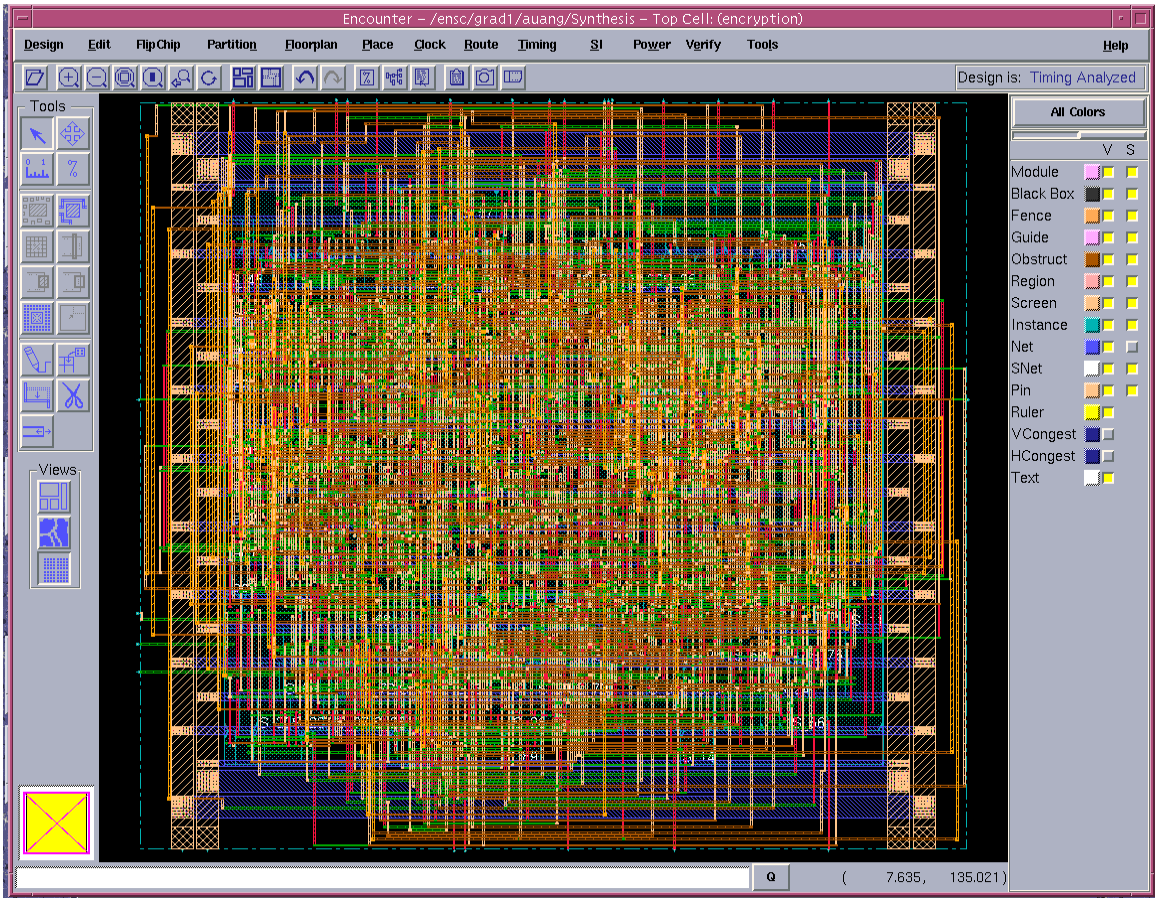


Figure 42: Encounter view of *Encryption_ao*

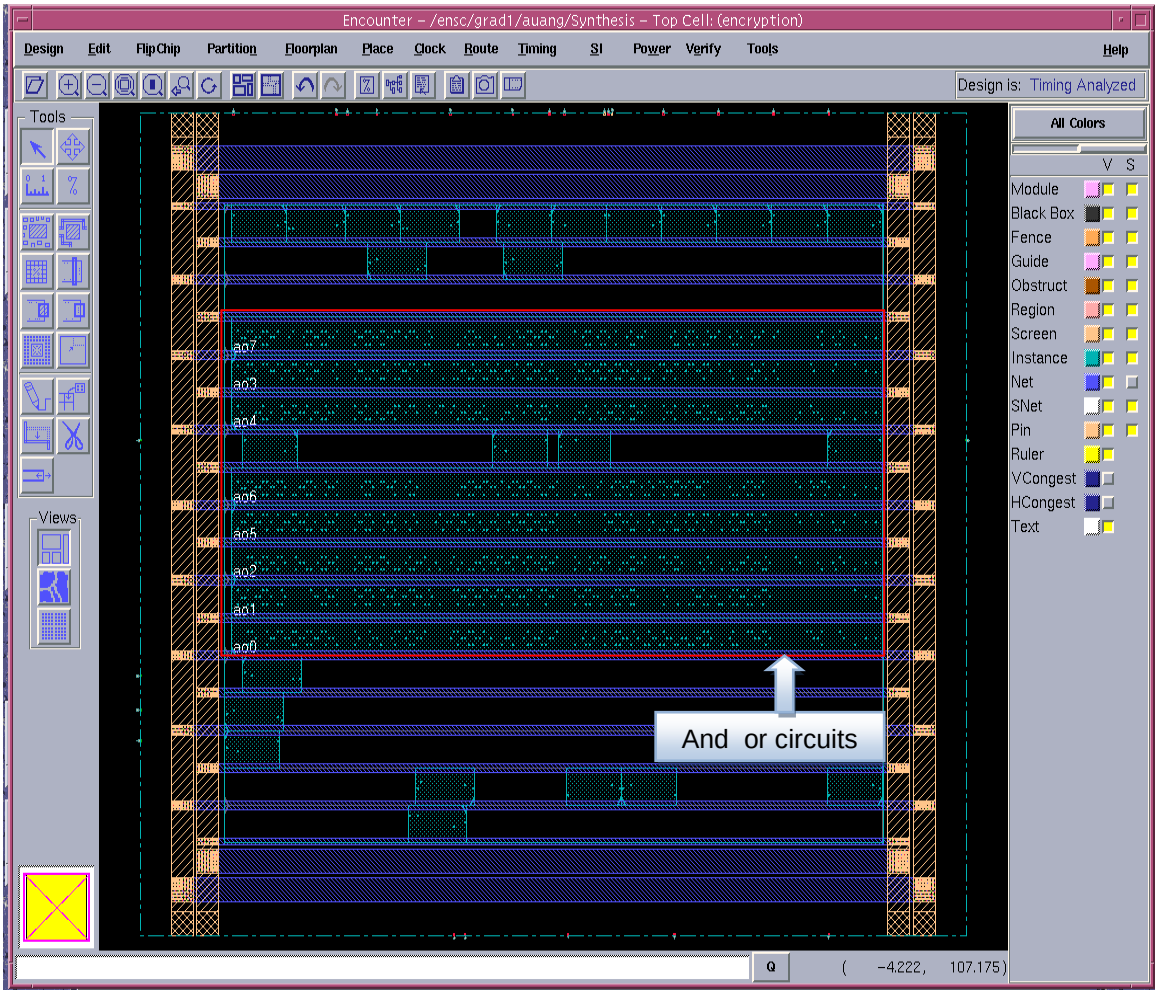


Figure 43: Encounter view based on standard cells

5: SIMULATION RESULTS

In order to observe the improvement brought by the custom decoder and the custom and_or circuits, area, speed and power of *encryption_raw*, *encryption_de*, *encryption_ao_diff* and *encryption_ao_domino* were measured and analyzed by Synopsys, Encounter and Hspice.

5.1 Output Verification

Output verification is an important process before fabricating real circuits. Two verification methods are implemented: VHDL test bench and Hspice.

The designer writes the input values into the test bench files and compares the VHDL simulation results with the correct output values to see if they are matched. Several randomly selected input values were verified.

Hspice is a powerful tool for simulating outputs for circuits. Hspice code can be extracted from the layout using the Cadence Analog Artist tool. In order to retrieve an accurate simulation result, wire capacitance within the algorithm has to be included. This can be done by turning on the “parasitic_caps” switch when

generating the extracted view. Then, by using the Cadence tool, analog environment, Hspice codes of each algorithm can be generated.

The waveform for each input bit was specified by the function “PULSE”. Like the VHDL test bench, several randomly selected input waveforms were verified. Verification by VHDL test bench and Hspice with several randomly selected inputs should provide a high reliance on the circuit. However, in real fabrication situation, all 256 combinations should be verified. The output waveforms for the case that inputs toggle from 0x00 to 0xFF at 10 ns are displayed in Figure 44.

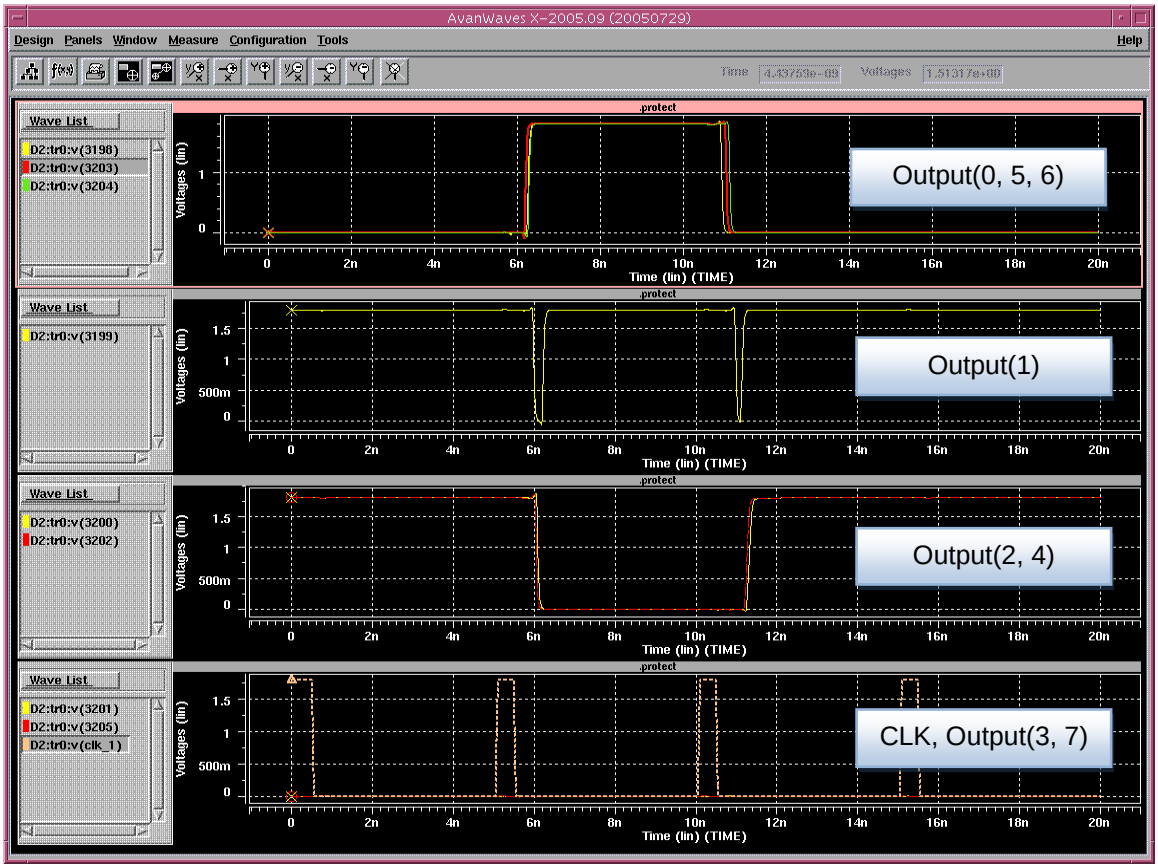


Figure 44: Output Simulation for inputs toggling from 0x00 to 0xFF at 10 ns for *Encryption_raw*

5.2 Differential circuit verification

One important feature of the differential circuit within the and_or circuit is the voltage equalizer. The affect of this voltage equalizer is clearly shown in Figure 45 as the output with ‘0’ is pulling up and the output with ‘1’ is pulling down during equalization phase. The two graphs in Figure 46 used different size of NMOS transistors in and_or sub-circuits and different size of PMOS transistors, P1 and P2 in Figure 32 and produced different performances. Thus, by adjusting the size of NMOS and PMOS transistors defined in and_or circuits and by measuring true and complement signals before feeding into the inverters in the differential circuit, node 1 and node 2 in Figure 32, an optimized size can be obtained. The results are list into Table 4.

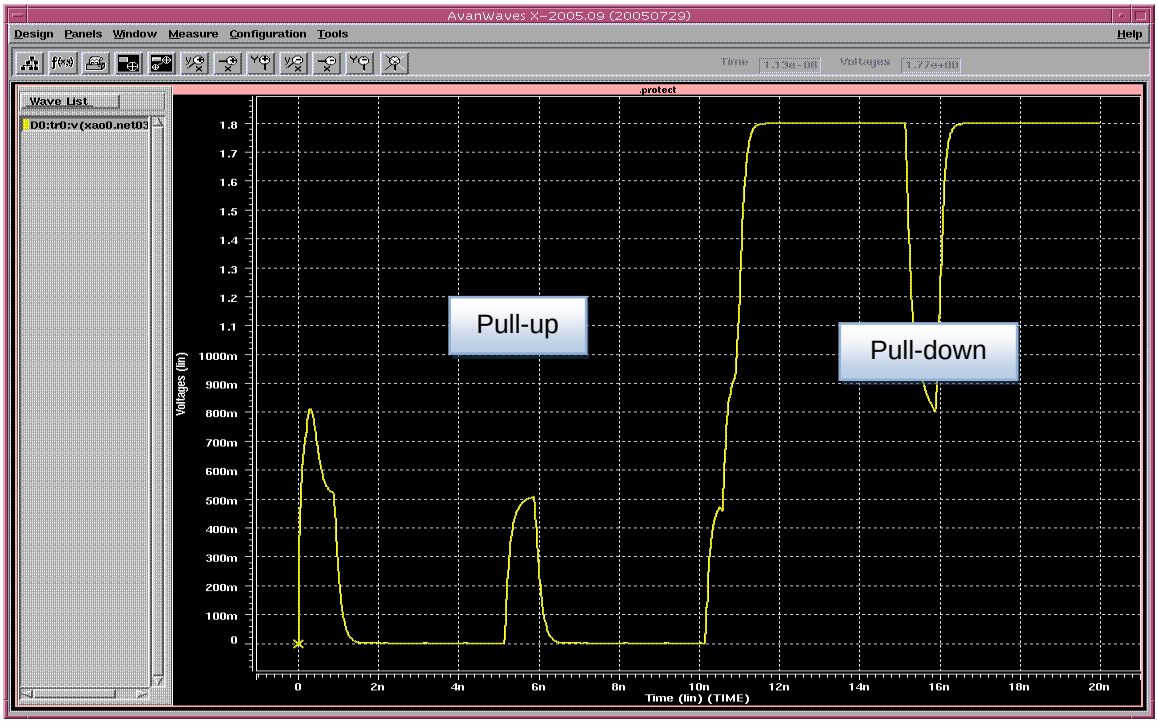


Figure 45: Effect of voltage equalizer

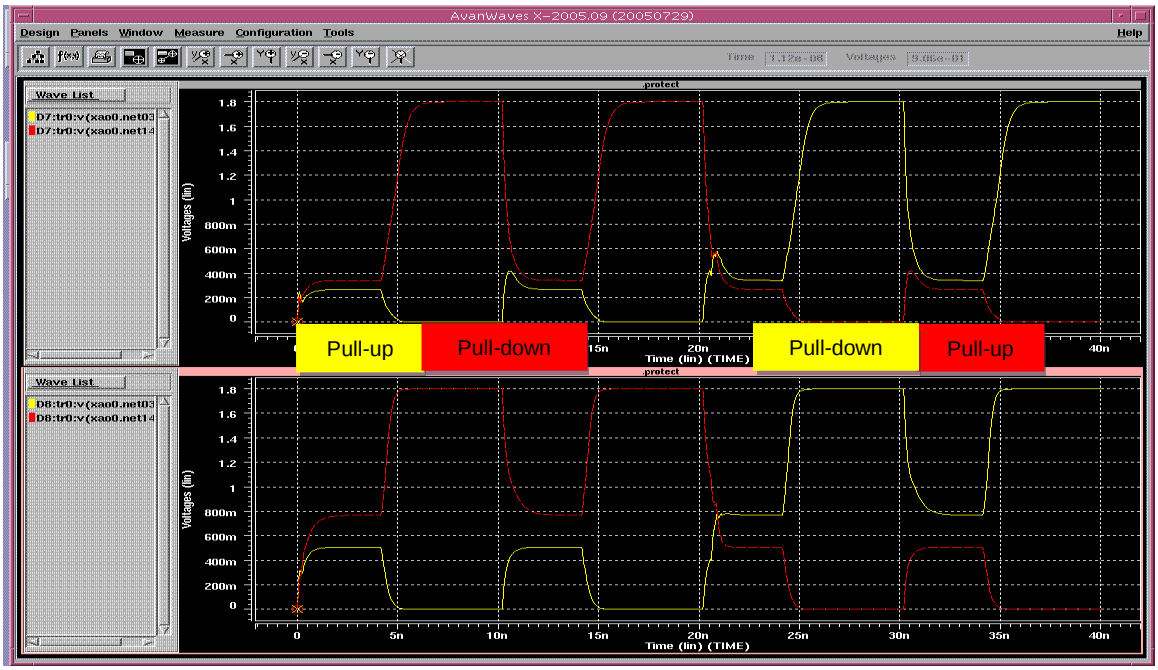


Figure 46: Pull-up/down effect with different size of transistors

In Table 4, voltage swing refers to the voltage change due to the effect of the equalizer. The desirable voltage swing is half of VDD, which is 0.9 V in this case. The swing time is the time taken for the signal to reach the swing voltage during voltage equalization phase. On the other hand, the recovery time is the time taken for the equalized voltage to swing back to a valid value after equalization phase. Both times are measured from 10% to 90% of voltage swings. The first one that NMOS has a width of 0.99 μm and PMOS has a width of 0.42 μm is the original size defined in the Cadence layout.

NMOS (µm)	PMOS (µm)	Voltage swing (V)	Swing time (ns)	Recovery time (ns)
0.99	0.42	0 → 0.254	0.3	0.8
		1.8 → 0.304	0.9	1.4
0.63	0.42	0 → 0.414	0.2	0.7
		1.8 → 0.340	0.7	1.2
0.63	0.84	0 → 0.427	0.3	0.5
		1.8 → 0.579	0.6	0.6
0.42	0.42	0 → 0.444	0.3	0.6
		1.8 → 0.465	0.8	1.1
0.42	0.84	0 → 0.506	0.6	0.5
		1.8 → 0.769	0.6	0.9
0.42	1.26	0 → 0.475	0.5	0.5
		1.8 → 1.14	0.6	0.3

Table 4: Voltage swing with respect to transistor sizes

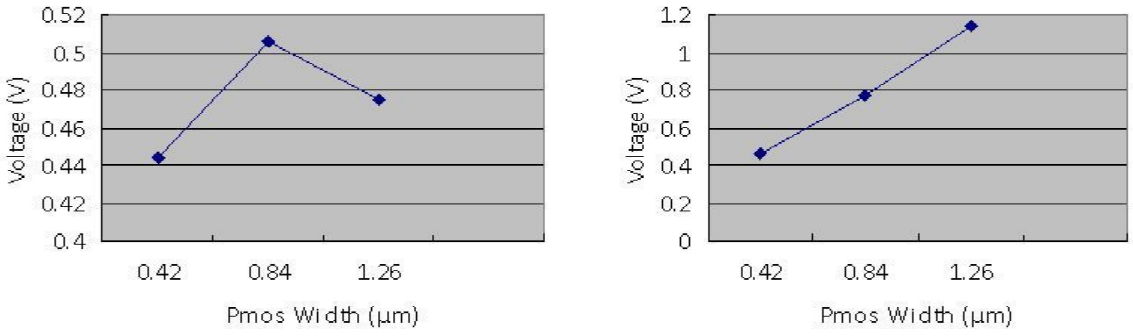


Figure 47: Voltage Swing for constant NMOS transistor of 0.42 µm in and_or sub-circuits (Left: Pull-up case, Right: Pull-down case)

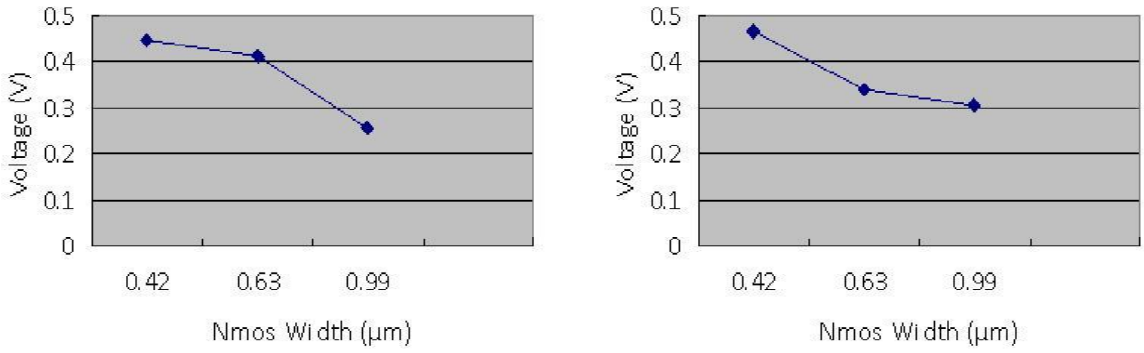


Figure 48: Voltage Swing for constant PMOS transistor of 0.42 µm, P1 and P2 in Figure 32 (Left: Pull-up case, Right: Pull-down case)

The recovery time exceeds 1 ns when the size of PMOS is 0.42 µm. The optimized sizes for PMOS and NMOS were chosen by observing Table 4, Figure 47 and Figure 48. PMOS transistor of 0.84 µm and NMOS transistor of 0.42 µm reach the closest voltage swing value to 0.9 V. However, this combination has a relatively long pull-down recovery time of 0.9 ns. Another combination generating close voltage swing value to 0.9 V is NMOS of 0.42 µm and PMOS of 1.26 µm. Both the swing and recovery time stay relative low under this combination. Thus, NMOS size of 0.42 µm and PMOS of 1.26 µm were used for Hspice optimized-speed simulation.

Another important factor is the length of the equalization phase since the new output values cannot be defined during this phase. If the phase is too long, then the new outputs are delayed. However, if the phase is too short, voltages cannot fully swing to the expected values. Thus, several equalization periods were tested and the optimized period was found to be 0.4 ns. The critical time for

the differential circuit to swing to the correct output after equalization is 0.29 ns. Therefore, the total critical time is 0.69 ns.

However, there is a drawback for *encryption_ao_diff*. Finding out the optimized period of equalization for one and_or circuit is not hard but finding out one for all 8 and_or circuits is difficult since different and_or circuits have different fan-ins. The advantage of voltage equalizer is lost if the delay and width of equalization phase is not optimized. Thus, an equalization pulse with suitable delay and width for all 8 and_or circuits is required. For comparison, another and_or circuit that takes out the voltage equalizer was also simulated and is named and_or_diff (WT). To simplify the work, the area of and_or_diff with and without “Equ” was designed to be the same.

5.3 Domino Circuit verification

As shown in Figure 35, output_z is pull-up to VDD during the precharge phase. Thus, the precharge phase has to be long enough for output_z to be fully pulled up. Like the differential circuit, different size of transistors within the and_or_domino would result in different performances. The most optimized precharge phase was found to be 0.8 ns with NMOS of 0.42 μm and PMOS of 1.26 μm. The critical evaluation phase was found to be 0.37 ns so that the total critical time is 1.17 ns. Thus, the critical time of and_or_diff was proven faster

than the one of and_or_domino. Figure 49 demonstrates the precharge and evaluation phases.

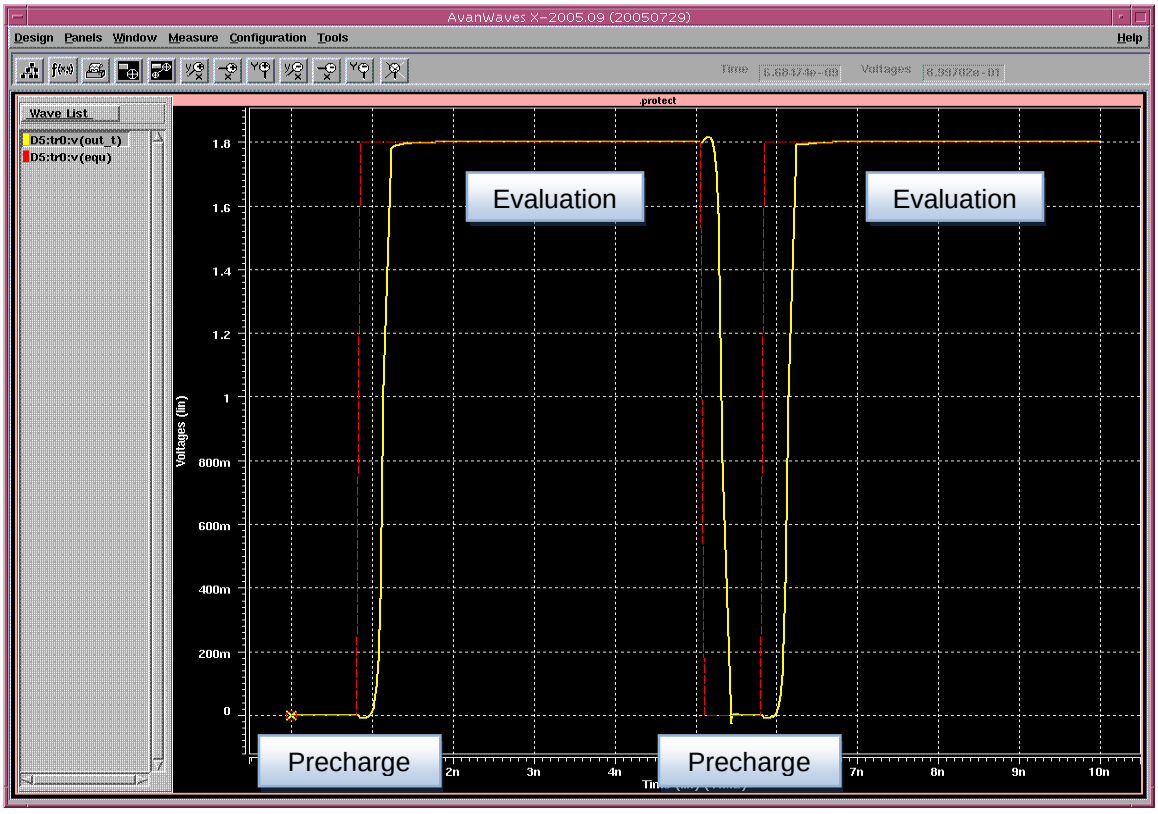


Figure 49: Hspice simulation for and_or_domino (Red: Clock; Yellow: Signal)

5.4 Area

Area is an important factor in this thesis. Data input of the AES encryption is 128 bits wide. Therefore, AES requires either 16 S-BOXes or 48 T-BOXes to perform all bits in parallel. Thus, by reducing the area of G-BOX, the drawback of T-BOXes can be solved. Moreover, reducing the size of the chip can increase the yield percentage and decreases the cost.

Area can be measured by either Synopsys or Encounter. However, Synopsys only provides an approximate estimation while Encounter calculates the whole chip area.

Synopsys estimates the area by adding up the combinational and non-combinational area. It does not include the area occupied by interconnection wires or spaces between standard cells. Thus, Synopsys only provides a roughly estimated area. A summary of area is listed in Table 5.

	Encryption_raw	***Encryption_de***	***Encryption_ao_diff***	***Encryption_ao_domino***
Area (μm^2)	45,551	22,308	11,651	11,553

Table 5: Area summary from Synopsys

Encounter calculates the area occupied by the chip based on two factors: the aspect ratio and the row density. The aspect ratio is defined as W/L where W is the width and L is the length of the chip. For example, a square chip has an aspect ratio of 1.0. Since most of the standard cells used in the *encryption_raw* and *encryption_de* have high aspect ratios, the aspect ratio is set as 1.0 in *encryption_raw* and *encryption_de*. On the other hand, since the and_or circuit has a narrow rectangular shape with an aspect ratio of 0.053, a rectangular shape of chip might be more suitable for and_or circuits to fit in. Therefore, the

aspect ratio was set as 0.6 at the beginning and increased gradually to 1.0. Meanwhile, the row density is increasing to achieve the most optimized area.

The row density is defined as “standard cell area/allocated area”. If the value of this term is larger than the one actually needed, some space is wasted and there will be some empty area within the chip. Figure 22 was obtained with aspect ratio of 0.6 and row density of 0.6 and there was some space wasted around the right region of the chip. On the other hand, Figure 50 was derived with aspect ratio of 1.0 and row density of 0.98 and the later one clearly shows a more compact layout view. Thus, the most optimized-area layout can be obtained by increasing the density to the maximum value. Encounter log file shows violation warning if the density specified exceeds the maximum feasible value. Table 6 lists the maximum row density and area for all four algorithms.

	Encryption_raw	***Encryption_de***	***Encryption_ao_diff***	***Encryption_ao_domino***
Max density	0.981	0.981	0.942	0.963
Area (μm^2)	60,228	32,826	20,067	19,304

Table 6: Maximum density and area

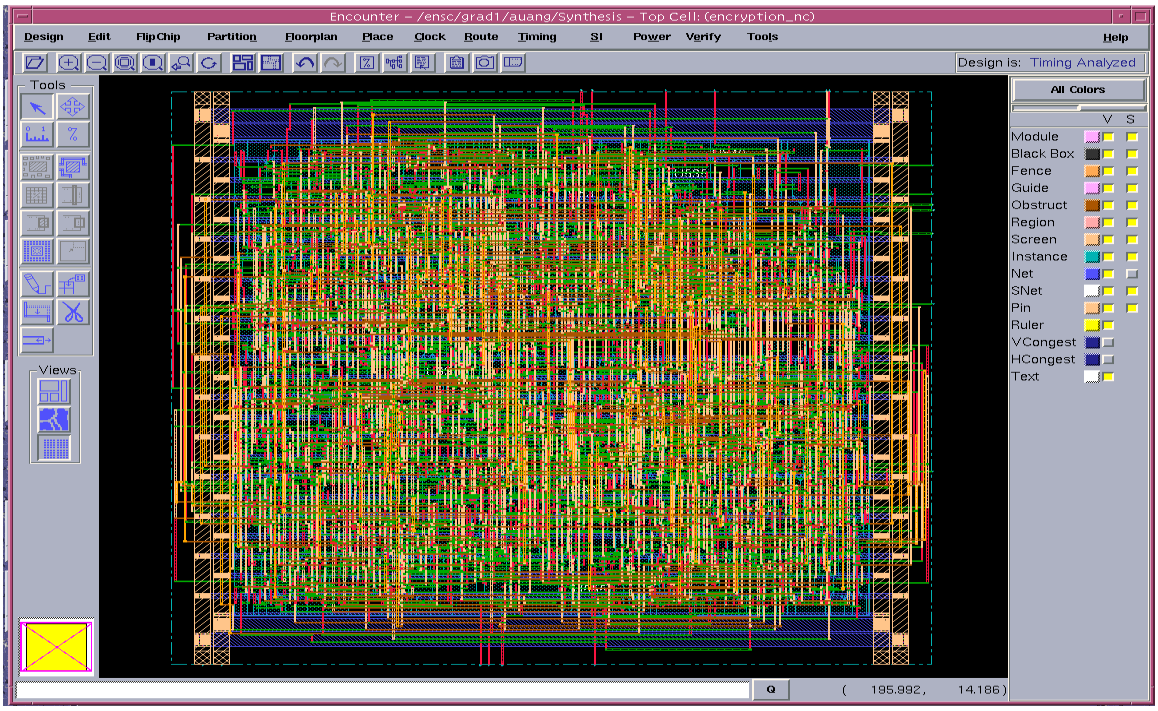


Figure 50: Encounter for *encryption_de* with aspect ratio of 1.0 and density of 0.98

The row density of the *encryption_ao* algorithm cannot go over 0.9 when the aspect ratio is set at 1.0. The limitation might be because the and_or circuits have very long rectangular shapes with very low aspect ratio. Therefore, the aspect ratio is set at 0.9 for *encryption_ao* algorithms.

There is a huge improvement from *encryption_raw* to *encryption_de*. The chip area is reduced almost to half. Thus, the custom decoder is proven area-efficient. The improvement is due to the much less number of MOSFETs used in *encryption_de*. A summary of number of MOSFETs used is in Table 7. *Encryption_ao_diff* and *encryption_ao_domino* have very close value of area since the area of the domino circuit is slightly smaller than the one of the

differential circuit. More area reduction is obtained the custom and_or circuit. The chip area is further reduced to only one third of the reference layout.

	Encryption_raw	***Encryption_de***	***Encryption_ao_diff***	***Encryption_ao_domino***
Number of MOSFETs	17161	8364	3940 3684 (WT)	3060

Table 7: Summary of number of MOSFETs

5.5 Speed

Like area, speed is also an important factor to be considered. In a iterative rounds of computing work system like AES, even a small improvement of speed in a round could cause a significant time reduction for the whole system. Speed can be measured by all three software, Synopsys, Encounter and Hspice. Hspice usually gives the most accurate estimation and Synopsys usually provides the least accurate estimation.

5.5.1 Critical Path

Critical path is a standard method to compare speed performance. Critical paths can be derived by both Synopsys and Encounter. However, Synopsys estimates the delay only by summing up the delay of each cell while Encounter estimates the delay by not only summing up the delay but also including the wire capacitance and load. Thus, different results might be produced. Therefore,

comparing both results helps to retrieve more accurate analyses about the algorithms. Due to the lack of custom and_or cell libraries in Synopsys and Encounter, very brief custom and_or cell libraries were written and included. However, Synopsys and Encounter seemed to have trouble fully utilizing the custom and_or cell libraries and generating correct speed estimation. Thus, speed performance of *encryption_ao* algorithms is only simulated by Hspice.

Figure 51 and Figure 52 depict the critical path generated by Synopsys for *encryption_raw* and *encryption_de*. The time estimated for *encryption_raw* is 1.85 ns and the critical path contain 11 standard cells. In contrast, the time estimated for *encryption_de* is 1.86 ns and the critical path possesses 14 standard cells. The number at the end of each cell name represents the size of the standard cell. Usually, a larger size is required if the fan-out of the cell is large.

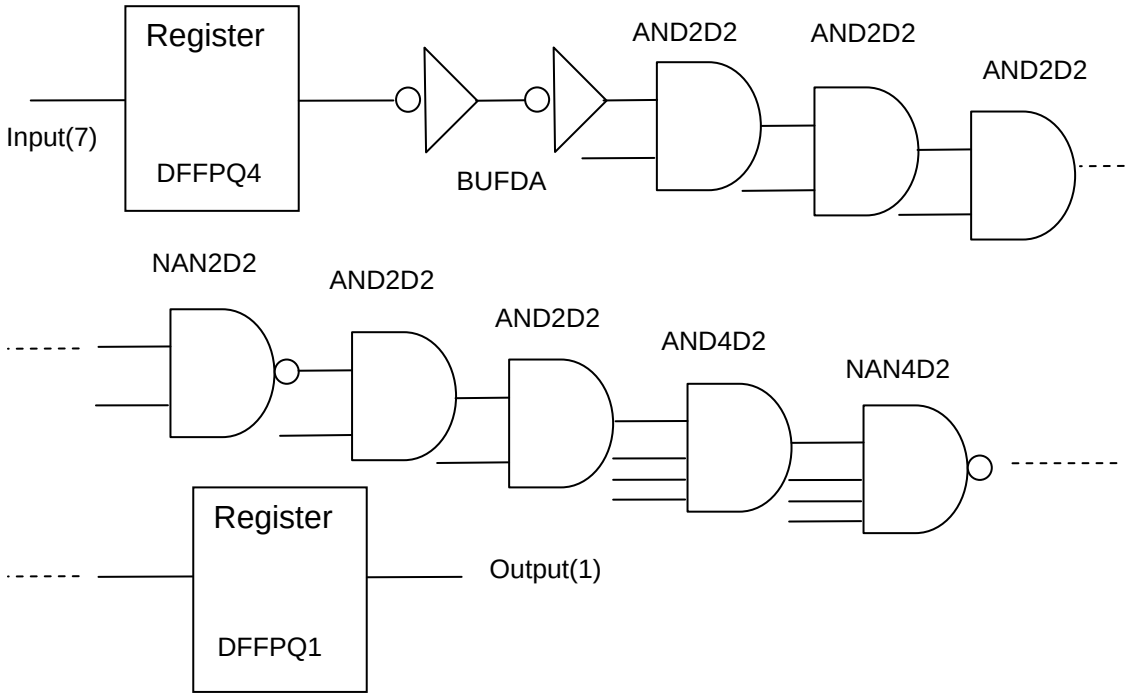


Figure 51: Critical path for *encryption_raw*

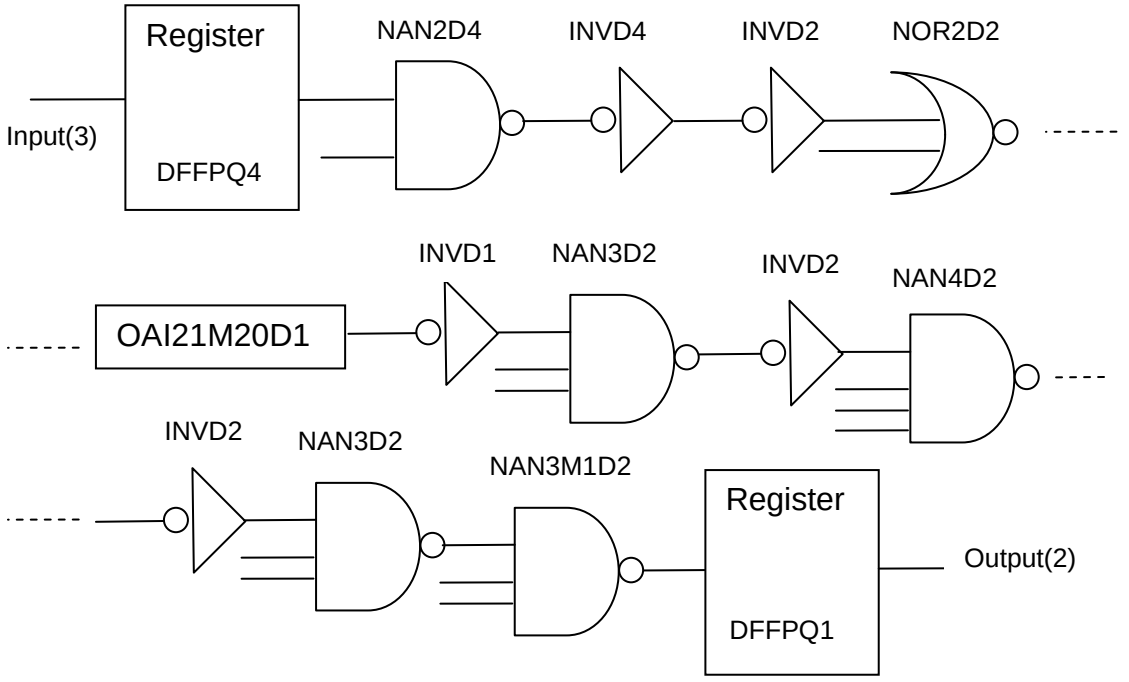


Figure 52: Critical path for *encryption_de*

Figure 53 and Figure 54 illustrate the critical path generated by Encounter for *encryption_raw* and *encryption_de*. There are two critical paths consuming the same amount of time in *encryption_raw* and are both shown in Figure 53. The time estimated for *encryption_raw* is 1.64 ns and the path contains 8 standard cells. The time estimated for *encryption_de* is 1.46 ns and the critical path has 12 standard cells.

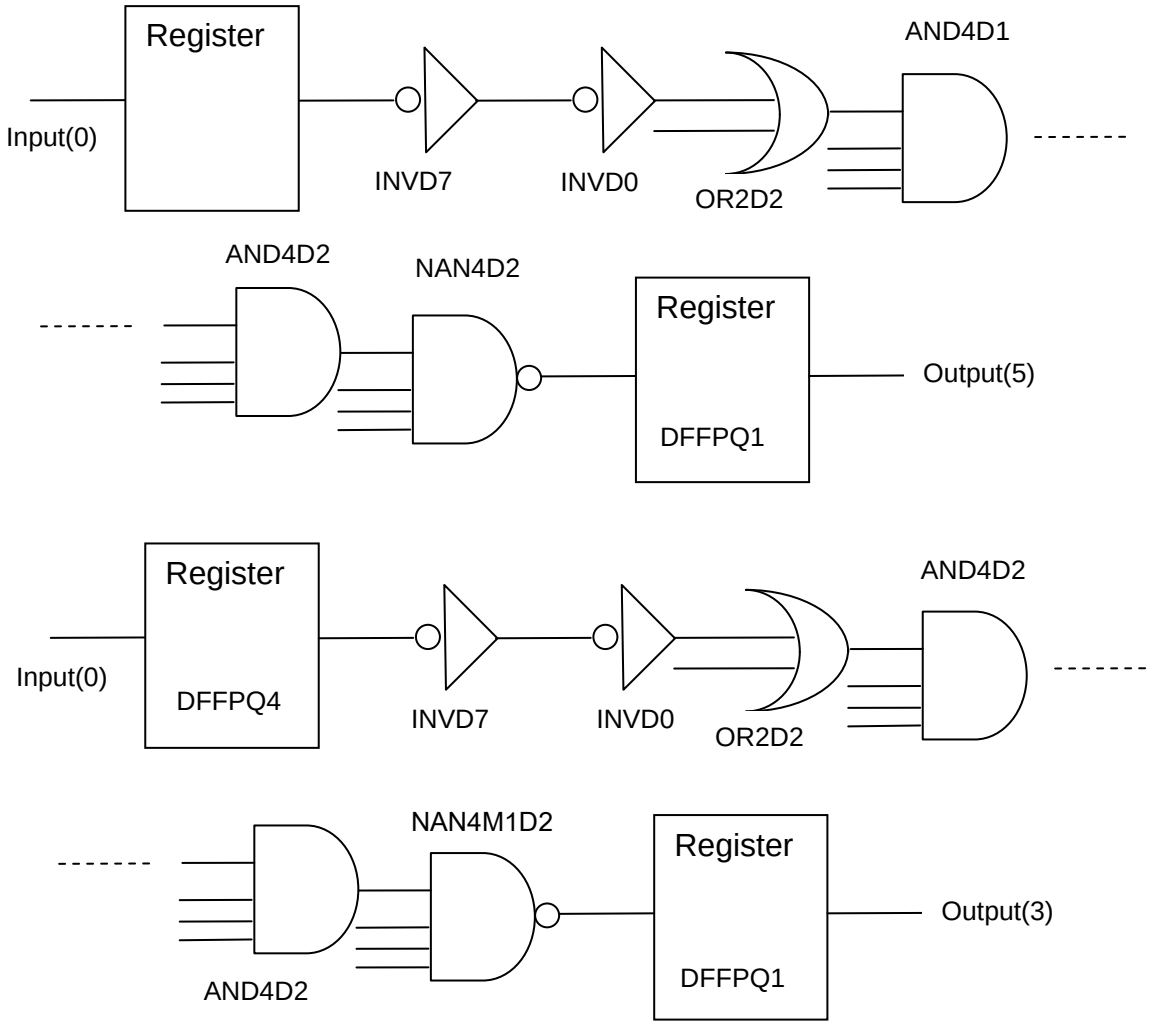


Figure 53: Critical path for *encryption_raw*

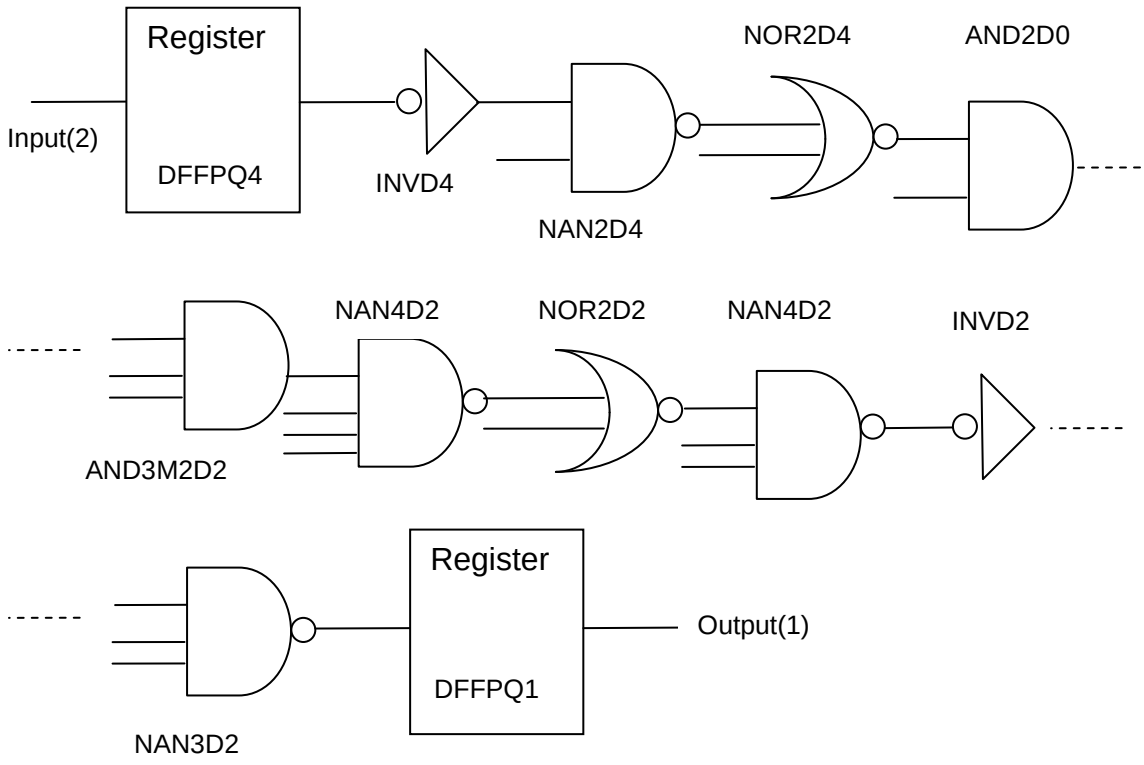


Figure 54: Critical Path for *encryption_de*

Table 8 summaries the time and number of cells estimated by Synopsys and Encounter. Synopsys estimates the time for both algorithms with almost the same value. On the other hand, Encounter estimates the time for *Encryption_de* with 0.18 ns shorter. However, both estimations show more numbers of cells are in the critical path within the *encryption_de* algorithm.

	Encryption_raw		***Encryption_de***	
	Time (ns)	Cells	Time (ns)	Cells
Synopsys	1.85	11	1.86	14
Encounter	1.64	8	1.46	12

Table 8: Summary of time and number of cells

Encryption_ao is the only algorithm, which requires two clock inputs; one is for registers and the other one is for “Equ”. As mentioned previously, the pulse placement of “Equ” is very important. Thus, an optimized pulse placement of “Equ” is required.

Figure 55 shows a possible way of generating “Equ” based on “CLK”. An ‘1’ is fed into a register and “Equ” becomes ‘1’ after t_s ns from the CLK edge. The output is fed back into the “reset” pin through buffers. Buffers can be cascaded to control “Equ” pulse width, t_w. Optimized pulse widths of “Equ” are listed in Table 9.

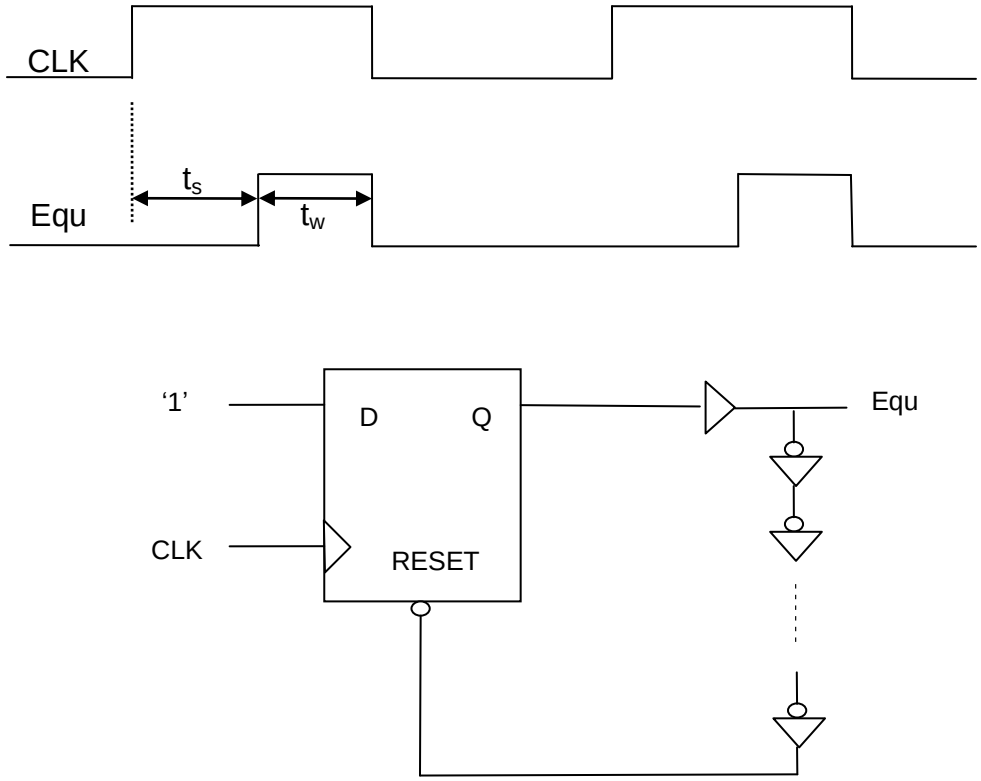


Figure 55: Equ generation

	Encryption_ao_diff	***Encryption_ao_domino***
"Equ" Period (ns)	0.4	0.8

Table 9: Periods of "Equ"

All four algorithms were simulated by Hspice and the critical times for all algorithms are summarized in Table 10.

	Encryption_raw	***Encryption_de***	***Encryption_ao_diff***	***Encryption_ao_domino***
Time (ns)	1.25	1.14	1.52 1.85 (WT)	1.61

Table 10: Summary of critical time by Hspice

From Table 10, the critical time for *encryption_de* is the fastest, slightly faster than *encryption_raw*. *Encryption_ao_diff*_WT is the slowest one. More detail information is required to study the longer delay in *encryption_ao* algorithms. Signals after the custom decoder before feeding into the custom and_or circuits were measured and displayed in Table 11.

	Encryption_de	***Encryption_ao_diff***	***Encryption_ao_domino***
Custom decoder	0.8	1.2	1.1
Remaining and_or circuit	0.34	0.32 0.65 (WT)	0.51

Table 11: Critical time (ns) consumed in the custom decoder

Table 11 clearly shows that the custom decoders in the two *encryption_ao* algorithms consume much more time than the one in *encryption_de*. The reason might be due to the incompletely specified cell library for custom and_or circuits included during Synopsys synthesis so that the most optimized standard cells

and their sizes within the custom decoder cannot be chosen correctly. In fact, the custom and_or_diff is actually 0.02 ns faster than the synthesized and_or circuit.

Moreover, the ideal critical time consumed in the custom decoder should be equal or less than the equalization period and precharge phase of and_or_diff and and_or_domino. However, the custom decoders within the *encryption_ao* algorithms consume much more time than the optimized time periods list in Table 9. Thus, the total critical time in *encryption_ao* algorithms is highly delayed by the custom decoder. A possible solution for it is discussed in Section 6.2.

5.6 Power

Power can be measured either by Encounter or by Hspice. However, Encounter does not contain precise transistor models and it utilizes toggle probability to estimate power for signal transition so that the estimation is not as precise as the one generated by Hspice. Thus, only Hspice is used to estimate power. Hspice can measure the average power by adding the command line ".meas tran avg_pwr avg power" inside the source code. The power consumed changes with different set of inputs. Table 12 lists the amount of power consumed when the input is initialized to 0xFF and toggles to 0x00 then back to 0xFF. The total simulation time is 14.9ns (3 cycles).

	Encryption_raw	***Encryption_de***	***Encryption_ao_diff***	***Encryption_ao_domino***
Power (W)	0.0163	0.00338	0.00318 0.00283 (WT)	0.00361

Table 12: Summary of power

The power consumption is significantly reduced from *encryption_raw* to *encryption_de*. The reason is the same as in Section 5.4: less number of MOSFETs are used. The power dissipation of *encryption_ao_diff*_WT is the lowest since it doesn't possess the feature as voltage equalizer or precharging like the other and_or circuits. The voltage equalization and precharging could waste power since voltage equalization and precharging occur periodically and there is a chance that outputs stay the same before and after the voltage equalization and precharge phase. In this case, outputs have to return to the original values after equalization or precharge phase and power is wasted. Thus, by taking out the voltage equalizer in *encryption_ao_diff*, unnecessarily consumed power can be saved. *Encryption_ao_domino* consumes higher power than *encryption_ao_diff* since the outputs of *encryption_ao_domino* have to be pull-up to VDD during precharge phase while outputs in *encryption_ao_diff* only swing to approximately half of VDD during equalization phase.

5.7 Comparison

A summary for *encryption_ao_diff*, *encryption_ao_domino*, *encryption_de* and *encryption_raw* in terms of area, power and speed is list in Table 13.

	Area (μm^2)	**Power (W)**	**Speed (ns)**	**Product ($\mu m^2 \cdot W \cdot ns$)**
Encryption_ao_domino	**19,304**	0.00361	1.61	112
Encryption_ao_diff	20,067	0.00318 **0.00283 (WT)**	1.52 1.85 (WT)	**97** 105
Encryption_de	32,826	0.00338	**1.14**	126
Encryption_raw	60,228	0.0163	1.25	1227

Table 13: Summary for comparison

The custom decoder is proven to have improvement in area, power and speed. The area is significantly reduced to half, power is decreased by 80% and propagation delay is shorten by 8.8%.

The maximum row density of *encryption_raw* and *encryption_de* is the same and is just slightly higher than the ones in *encryption_ao* algorithms. However, *encryption_ao* algorithms occupy much less chip area due to the custom and_or circuits. The power consumption of *encryption_ao_diff* is lower than the one of *encryption_de* and encryption_domino. *Encryption_ao_diff*_WT consumes the least amount of power but has the longest delay.

Encryption_ao_domino has the least amount of area but slightly higher power consumption and delay than *encryption_ao_diff* does. The main drawback of *encryption_ao* algorithms is the long delay through the custom decoder.

Besides the reference algorithm, each algorithm ranks the best in one category: *encryption_de* in speed, *encryption_ao_diff* in power and *encryption_ao_domino* in area. Based on the objective of the design, the most suitable algorithm can be selected. The product of area, speed and power can be used as a reference to compare the overall performance between the four algorithms and *encryption_ao_diff* has the best overall performance.

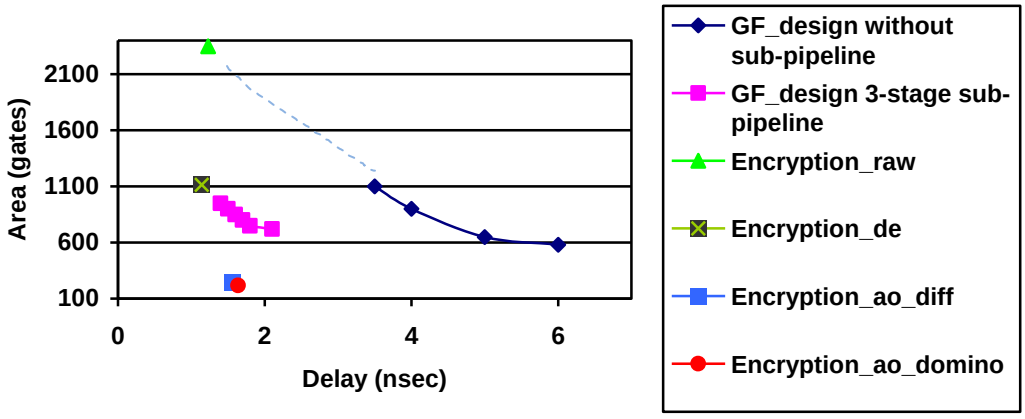


Figure 56: S-BOX Performance comparison [22]

Figure 56 displays the performance comparison between different encryption algorithms and results from another paper, which was also based on 0.18 µm CMOS technology [22]. *Encryption_raw* sits on the extension line of the

GF design without sub-pipeline while *encryption_de* sits on the extension line of the GF design with 3-stage sub-pipeline. Without any sub-pipeline within *encryption_de*, *encryption_de* can already perform as well as a 3-stage sub-pipeline GF design. Moreover, *Encryption_ao* algorithms without sub-pipelines locate on the very bottom of the graph proving that *encryption_ao* algorithms have much more area reduction than other ones while maintaining good speed performance. Section 6.4 further discusses sub-pipelines.

5.8 Area compensation of T-BOX architecture

The custom decoder and custom and_or circuits are shown to provide an area-efficient encryption algorithm and yet to maintain a satisfactory speed performance compared with other work mentioned above. The area can be reduced to only one third of the reference algorithm. Thus, fitting 3 T-BOXes in the same area as one traditional S-BOX becomes feasible. Figure 57 demonstrates the two algorithms.

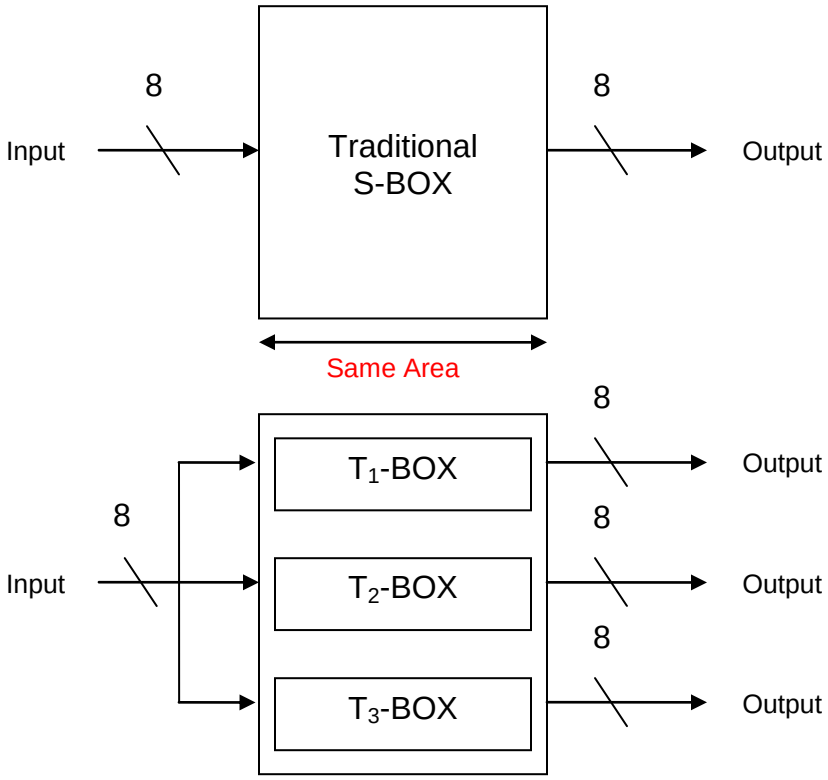


Figure 57: Demonstration of S-BOX and T-BOXes

6: FUTURE IMPROVEMENT

Even though the custom designs are proven to have huge area reduction and speed improvement, there are still some steps that may provide further improvement.

6.1 Cell library

When Synopsys and Encounter interpret the cells used in the algorithm, they look for the information of each cell from cell libraries. Very brief cell libraries containing the input capacitance and output delay for and_or_diff and and_or_domino were written for Synopsys and Encounter. However, how well the synthesis process was performed by Synopsys and Encounter remains questionable since a more detailed cell library could certainly result in a better synthesis performance. The Hspice simulation showed that the custom decoder in *encryption_ao* consumed up to 0.4 ns more than the one in *encryption_de*. Therefore, a possible improvement can simply be achieved with more detailed and accurate cell libraries.

6.2 Full-Custom decoder

The custom decoders in *encryption_ao* algorithms consist of different kinds of standard cells. The cells are chosen by Synopsys. However, a full-custom decoder designed to support the custom and_or circuit might provide a faster speed to the whole algorithm. Therefore, a full-custom decoder could be a next step to further improve the encryption algorithm.

6.3 Area Improvement on Domino Circuit

As shown in Figure 36, the left bottom part is empty. Thus, an even more area reduced algorithm can be achieved if half of the above sub-circuits are moved to the bottom so that the size of the and_or_domino is decreased to half. The schematic view of this new algorithm is depicted in Figure 58. Each new and_or_domino can save up to 339 μm^2, resulting a total area reduction of 2716 μm^2. The area is further reduced to 28% of the one in the reference algorithm so that an even smaller chip area is feasible.

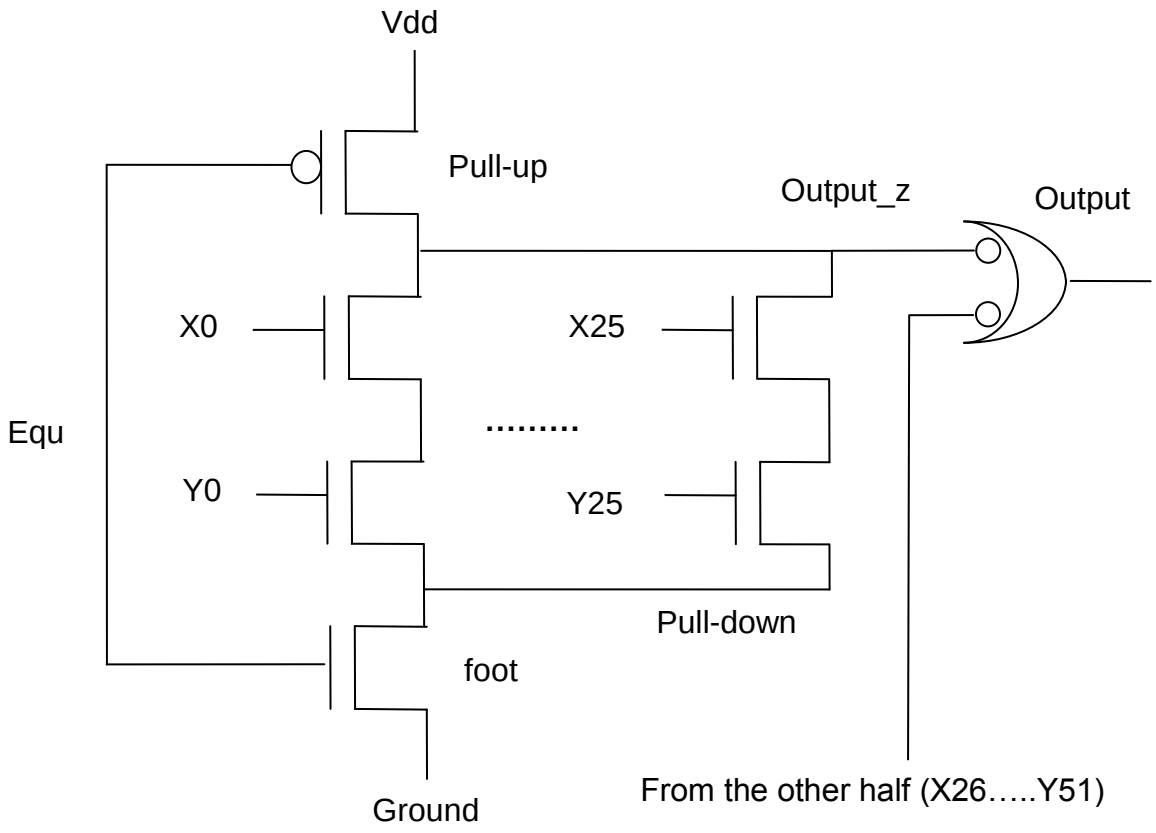


Figure 58: Schematic of an improved version of and_or_diff

6.4 Sub-pipeline for *encryption_de*

Encryption_de is the fastest one among all algorithms. If a register is inserted between the custom decoder and synthesized and_or circuits, a sub-pipelined structure is formed. The new delay depends on the highest critical delay time among all stages. Therefore, the new delay of *encryption_de* can be reduced to 0.8 ns. Since the remaining synthesized and_or circuit only uses 0.34 ns, the XOR operations after the T-BOXes shown in Figure 14 can be cascaded with the and_or circuits. The XOR operations roughly take 0.3 ~ 0.4 ns by Hspice

simulation. Thus, the T-BOX architecture can perform even faster. The sub-pipeline structure is shown in Figure 59.

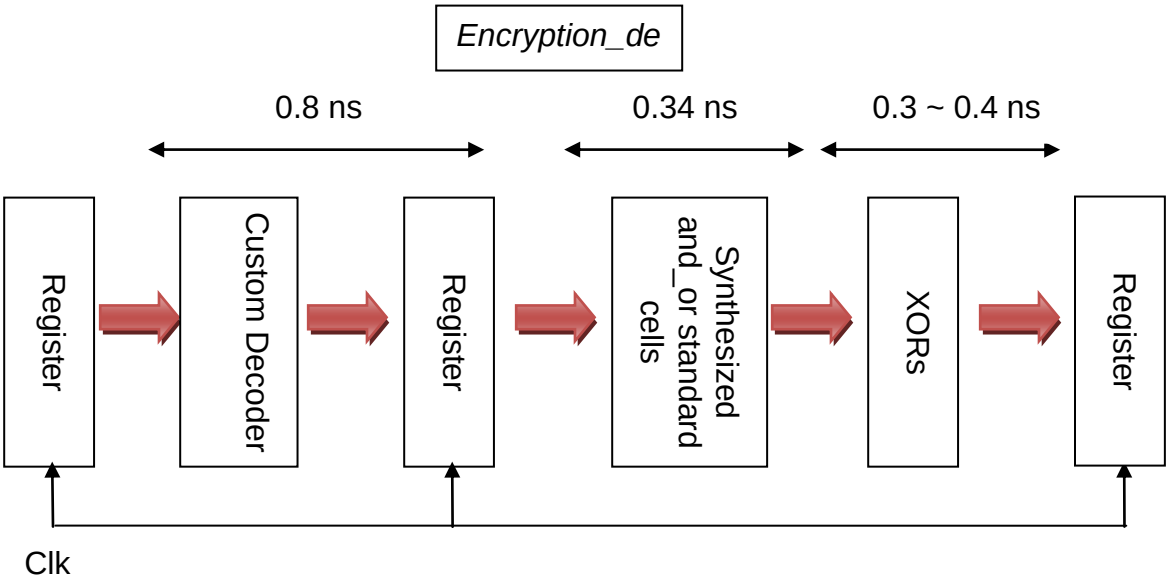


Figure 59: Sub-pipeline structure

7: CONCLUSION

Due to the increasing need of confidential information being transmitted over the Internet, a more robust and strong cryptographic security is needed. AES has replaced the old secure algorithm and has been using since 2001. SubBytes operation is proven the critical block and it executes iteratively in the AES algorithm. Therefore, an improved SubBytes algorithm will result a significant improvement in AES.

This thesis focused on implementing new custom hardware algorithms for SubBytes operation. The new custom hardware is based on G-BOX that is compatible for both S-BOX and T-BOXes. Due to the similar properties among all BOXes, S-BOX (T_1-BOX) is selected for G-BOX in this thesis. A custom decoder and eight custom and_or circuits were implemented from the G-BOX truth tables. A generic C++ code was written to convert truth tables to VHDL codes. Four different algorithms were implemented to observe the improvement brought by the custom decoder and custom and_or circuits.

Two powerful synthesis tools, Synopsys and Encounter, were applied on the generated VHDL codes. Synopsys chose the size of standard cells to

achieve an optimized performance. Encounter located components and routed wires based on the specified aspect ratio and row density.

Synopsys, Encounter and Hspice were used to simulate area, speed and power performance for different algorithms. The custom decoder is proven to have significant area and power reduction. The custom and_or circuits are proven to have the most area reduction. The area is at most reduced to one third of the one in the reference algorithm.

The significant reduced area of *encryption_ao* algorithms allows 3 T-BOXes to fit within the same area as 1 traditional S-BOX. This compensates the large area requirement for the T-BOX architecture.

There are several opportunities for future works as a result of this Thesis. More detailed and_or cell library files such as LIB and TLF files might help the synthesis tools to produce more optimized results. A full-custom decoder is another possible way to increase the speed performance for *encryption_ao* algorithms. Further area reduction can be performed on the *encryption_ao_domino* algorithm. Moreover, a sub-pipeline structure can reduce the critical time within each pipeline stage so that more speed improvement can be achieved.

APPENDICES

Appendix 1: T_1-BOX Truth Table Simplification for output bit 0

Un-simplified truth table:

```
.i 8
.o 1
.p 256
00000000 1
00000001 0
00000010 1
00000011 1
00000100 0
00000101 1
00000110 1
00000111 1
00001000 0
00001001 1
00001010 1
00001011 1
00001100 0
00001101 1
00001110 1
00001111 0
00010000 0
00010001 0
00010010 1
00010011 1
00010100 0
00010101 1
00010110 1
00010111 0
00011000 1
00011001 0
00011010 0
00011011 1
00011100 0
00011101 0
00011110 0
00011111 0
00100000 1
00100001 1
00100010 1
00100011 0
00100100 0
00100101 1
00100110 1
00100111 0
00101000 0
00101001 1
00101010 1
00101011 1
00101100 1
00101101 0
00101110 1
00101111 1
00110000 0
00110001 1
00110010 1
00110011 1
00110100 0
00110101 0
00110110 1
00110111 0
00111000 1
00111001 0
00111010 0
00111011 0
00111100 1
00111101 1
00111110 0
00111111 1
01000000 1
01000001 1
01000010 0
01000011 0
01000100 1
01000101 0
01000110 0
01000111 0
01001000 0
01001001 1
01001010 0
01001011 1
01001100 1
01001101 1
01001110 1
01001111 0
01010000 1
01010001 1
01010010 0
01010011 1
01010100 0
01010101 0
01010110 1
01010111 1
01011000 0
01011001 1
01011010 0
01011011 1
01011100 0
01011101 0
01011110 0
01011111 1
01100000 0
01100001 1
01100010 0
01100011 1
01100100 1
01100101 1
01100110 1
01100111 1
01101000 1
01101001 1
01101010 0
01101011 1
01101100 0
01101101 0
01101110 1
01101111 0
01110000 1
01110001 1
01110010 0
01110011 1
01110100 0
01110101 1
01110110 0
01110111 1
01111000 0
01111001 0
01111010 0
01111011 1
01111100 0
01111101 1
01111110 1
01111111 0
10000000 1
10000001 0
10000010 1
10000011 0
10000100 1
10000101 1
10000110 0
10000111 1
10001000 0
10001001 1
10001010 0
10001011 1
10001100 0
10001101 1
10001110 1
10001111 1
10010000 0
10010001 1
10010010 1
10010011 0
10010100 0
10010101 0
10010110 0
10010111 0
10011000 0
10011001 0
10011010 0
10011011 0
10011100 0
10011101 0
10011110 1
10011111 1
10100000 0
10100001 0
10100010 0
10100011 0
10100100 1
10100101 0
10100110 0
10100111 0
10101000 0
10101001 1
10101010 0
10101011 0
10101100 1
10101101 1
10101110 0
10101111 1
10110000 1
10110001 0
10110010 1
10110011 1
10110100 1
10110101 1
10110110 0
10110111 1
10111000 0
10111001 0
10111010 0
10111011 0
10111100 1
10111101 0
10111110 0
10111111 0
11000000 0
11000001 0
11000010 1
11000011 0
11000100 0
11000101 0
11000110 0
11000111 0
11001000 0
11001001 1
11001010 0
11001011 1
11001100 1
11001101 1
11001110 1
11001111 0
11010000 0
```

11010001 0
11010010 1
11010011 0
11010100 0
11010101 1
11010110 0
11010111 0
11011000 1
11011001 1
11011010 1
11011011 1
11011100 0
11011101 1
11011110 1
11011111 0
11100000 1
11100001 0
11100010 0
11100011 1
11100100 1
11100101 1
11100110 0
11100111 0
11101000 1
11101001 0
11101010 1
11101011 1
11101100 0
11101101 1
11101110 0
11101111 1
11110000 0
11110001 1
11110010 1
11110011 1
11110100 1
11110101 0
11110110 0
11110111 0
11111000 1
11111001 1
11111010 1
11111011 1
11111100 0
11111101 0
11111110 1
11111111 0
.e

Simplified truth table:

.i 8
.o 1
.p 44
000-0101 1
000--011 1
001-1111 1
00-000-0 1
00-0101- 1
00-11000 1
00--0-10 1
01000-00 1
0101--11 1
010--001 1
0110100- 1
011-0--1 1
01-1000- 1
0-010110 1
0-100-01 1
0-10-110 1
0-111101 1
0-1-0001 1
10000-00 1
10010001 1
100-111- 1
101101-1 1
10110-00 1
1101-101 1
1110-000 1
111-10-0 1
11-110-- 1
1-0-0010 1
1-0-1110 1
1-1011-1 1
1-1-0100 1
1--10010 1
-00001-1 1
-01-1100 1
-0-01001 1
-10011-0 1
-110010- 1
-11100-1 1
-1111110 1
-11--011 1
--0010-1 1
--001110 1
--001-01 1
--110011 1
.e

Appendix 2: VHDL, SKILL, and C++ codes:

VHDL code for *encryption_raw*:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
USE ieee.std_logic_arith.ALL;

LIBRARY vs18sc;
USE vs18sc.ALL;

entity encryption_raw is
Port (
     data_in : in std_logic_vector(7 downto 0);
     clk_1 : in std_logic;
     data_out : out std_logic_vector(7 downto 0)
);
end encryption_raw;

architecture behaviour of encryption_raw is
signal input: std_logic_vector(7 downto 0);
signal output: std_logic_vector(7 downto 0);
begin
     process(clk_1)
       begin
         if (clk_1='1' and clk_1'event) then
            input <= data_in;
            data_out <= output;
         end if;
     end process;

process(input)
            begin
            if (input(7) = '0' and input(6) = '0' and input(5)
= '0' and input(3) = '0' and input(2) = '1' and input(1) =
'0' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '0' and
input(5) = '0' and input(2) = '0' and input(1) = '1' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '0' and
input(5) = '1' and input(3) = '1' and input(2) = '1' and
input(1) = '1' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '0' and
input(4) = '0' and input(3) = '0' and input(2) = '0' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '0' and
input(4) = '0' and input(3) = '1' and input(2) = '0' and
input(1) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '0' and
input(4) = '1' and input(3) = '1' and input(2) = '0' and
input(1) = '0' and input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '0' and
input(3) = '0' and input(1) = '1' and input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '1' and
input(5) = '0' and input(4) = '0' and input(3) = '0' and
input(1) = '0' and input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '1' and
input(5) = '0' and input(4) = '1' and input(1) = '1' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '1' and
input(5) = '0' and input(2) = '0' and input(1) = '0' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '1' and
input(5) = '1' and input(4) = '0' and input(3) = '1' and
input(2) = '0' and input(1) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '1' and
input(5) = '1' and input(3) = '0' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(6) = '1' and
input(4) = '1' and input(3) = '0' and input(2) = '0' and
input(1) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(5) = '0' and
input(4) = '1' and input(3) = '0' and input(2) = '1' and
input(1) = '1' and input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(5) = '1' and
input(4) = '0' and input(3) = '0' and input(1) = '0' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(5) = '1' and
input(4) = '0' and input(2) = '1' and input(1) = '1' and
input(0) = '0'  ) then

            output(0) <= '1';
```

```
            elsif (input(7) = '0' and input(5) = '1' and
input(4) = '1' and input(3) = '1' and input(2) = '1' and
input(1) = '0' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '0' and input(5) = '1' and
input(3) = '0' and input(2) = '0' and input(1) = '0' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '0' and
input(5) = '0' and input(4) = '0' and input(3) = '0' and
input(1) = '0' and input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '0' and
input(5) = '0' and input(4) = '1' and input(3) = '0' and
input(2) = '0' and input(1) = '0' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '0' and
input(5) = '0' and input(3) = '1' and input(2) = '1' and
input(1) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '0' and
input(5) = '1' and input(4) = '1' and input(3) = '0' and
input(2) = '1' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '0' and
input(5) = '1' and input(4) = '1' and input(3) = '0' and
input(1) = '0' and input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '1' and
input(5) = '0' and input(4) = '1' and input(2) = '1' and
input(1) = '0' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '1' and
input(5) = '1' and input(4) = '0' and input(2) = '0' and
input(1) = '0' and input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '1' and
input(5) = '1' and input(3) = '1' and input(2) = '0' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(6) = '1' and
input(4) = '1' and input(3) = '1' and input(2) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(5) = '0' and
input(3) = '0' and input(2) = '0' and input(1) = '1' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(5) = '0' and
input(3) = '1' and input(2) = '1' and input(1) = '1' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(5) = '1' and
input(4) = '0' and input(3) = '1' and input(2) = '1' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(5) = '1' and
input(3) = '0' and input(2) = '1' and input(1) = '0' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(7) = '1' and input(4) = '1' and
input(3) = '0' and input(2) = '0' and input(1) = '1' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(6) = '0' and input(5) = '0' and
input(4) = '0' and input(3) = '0' and input(2) = '1' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(6) = '0' and input(5) = '1' and
input(3) = '1' and input(2) = '1' and input(1) = '0' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(6) = '0' and input(4) = '0' and
input(3) = '1' and input(2) = '0' and input(1) = '0' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(6) = '1' and input(5) = '0' and
input(4) = '0' and input(3) = '1' and input(2) = '1' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(6) = '1' and input(5) = '1' and
input(4) = '0' and input(3) = '0' and input(2) = '1' and
input(1) = '0'  ) then

            output(0) <= '1';

            elsif (input(6) = '1' and input(5) = '1' and
input(4) = '1' and input(3) = '0' and input(2) = '0' and
input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(6) = '1' and input(5) = '1' and
input(4) = '1' and input(3) = '1' and input(2) = '1' and
input(1) = '1' and input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(6) = '1' and input(5) = '1' and
input(2) = '0' and input(1) = '1' and input(0) = '1'  ) then
```

```
            output(0) <= '1';

            elsif (input(5) = '0' and input(4) = '0' and
input(3) = '1' and input(2) = '0' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(5) = '0' and input(4) = '0' and
input(3) = '1' and input(2) = '1' and input(1) = '1' and
input(0) = '0'  ) then

            output(0) <= '1';

            elsif (input(5) = '0' and input(4) = '0' and
input(3) = '1' and input(1) = '0' and input(0) = '1'  ) then

            output(0) <= '1';

            elsif (input(5) = '1' and input(4) = '1' and
input(3) = '0' and input(2) = '0' and input(1) = '1' and
input(0) = '1'  ) then

            output(0) <= '1';

            else

      output(0) <= '0';

            end if;

…… Similar coding pattern up to output(7)

end process;
end behaviour;
```

VHDL code for *encryption_de*:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
USE ieee.std_logic_arith.ALL;

LIBRARY vs18sc;
USE vs18sc.ALL;

entity encryption_nc is
Port (
     data_in : in std_logic_vector(7 downto 0);
     clk_1 : in std_logic;
     data_out : out std_logic_vector(7 downto 0)
);
end encryption_nc;

architecture structure of encryption_nc is
signal input: std_logic_vector(7 downto 0);
signal output: std_logic_vector(7 downto 0);
signal input0z, input1z, input2z, input3z, input4z,
input5z, input6z, input7z: STD_LOGIC;
signal out_76, out_75, out_74, out_76z, out_75z,
out_74z, out_65, out_64, out_65z, out_64z, out_54,
out_54z, out_7z6, out_7z5, out_7z4, out_7z6z,
out_7z5z, out_7z4z, out_6z5, out_6z4, out_6z5z,
out_6z4z, out_5z4, out_5z4z : STD_LOGIC;
signal out_32, out_31, out_30, out_32z, out_31z,
out_30z, out_21, out_20, out_21z, out_20z, out_10,
out_10z, out_3z2, out_3z1, out_3z0, out_3z2z,
out_3z1z, out_3z0z, out_2z1, out_2z0, out_2z1z,
out_2z0z, out_1z0, out_1z0z : STD_LOGIC;
signal x_00, x_01, x_02, x_03, x_04, x_05, x_06, x_07,
x_08, x_09, x_010, x_011, x_012, x_013, x_014, x_015,
x_016, x_017, x_018, x_019, x_020, x_021, x_022,
x_023, x_024, x_025, x_026, x_027, x_028, x_029,
x_030, x_031, x_032, x_033, x_034, x_035, x_036,
x_037, x_038, x_039, x_040, x_041, x_042, x_043:
std_logic;
signal y_00, y_01, y_02, y_03, y_04, y_05, y_06, y_07,
y_08, y_09, y_010, y_011, y_012, y_013, y_014, y_015,
y_016, y_017, y_018, y_019, y_020, y_021, y_022,
y_023, y_024, y_025, y_026, y_027, y_028, y_029,
y_030, y_031, y_032, y_033, y_034, y_035, y_036,
y_037, y_038, y_039, y_040, y_041, y_042, y_043:
std_logic;
signal x_10, x_11, x_12, x_13, x_14, x_15, x_16, x_17,
x_18, x_19, x_110, x_111, x_112, x_113, x_114, x_115,
x_116, x_117, x_118, x_119, x_120, x_121, x_122,
x_123, x_124, x_125, x_126, x_127, x_128, x_129,
x_130, x_131, x_132, x_133, x_134, x_135, x_136,
x_137, x_138, x_139, x_140, x_141, x_142, x_143,
x_144, x_145, x_146, x_147, x_148, x_149: std_logic;
signal y_10, y_11, y_12, y_13, y_14, y_15, y_16, y_17,
y_18, y_19, y_110, y_111, y_112, y_113, y_114, y_115,
y_116, y_117, y_118, y_119, y_120, y_121, y_122,
y_123, y_124, y_125, y_126, y_127, y_128, y_129,
y_130, y_131, y_132, y_133, y_134, y_135, y_136,
y_137, y_138, y_139, y_140, y_141, y_142, y_143,
y_144, y_145, y_146, y_147, y_148, y_149: std_logic;
signal x_20, x_21, x_22, x_23, x_24, x_25, x_26, x_27,
x_28, x_29, x_210, x_211, x_212, x_213, x_214, x_215,
x_216, x_217, x_218, x_219, x_220, x_221, x_222,
x_223, x_224, x_225, x_226, x_227, x_228, x_229,
x_230, x_231, x_232, x_233, x_234, x_235, x_236,
x_237, x_238, x_239, x_240, x_241, x_242, x_243,
x_244: std_logic;
signal y_20, y_21, y_22, y_23, y_24, y_25, y_26, y_27,
y_28, y_29, y_210, y_211, y_212, y_213, y_214, y_215,
y_216, y_217, y_218, y_219, y_220, y_221, y_222,
y_223, y_224, y_225, y_226, y_227, y_228, y_229,
y_230, y_231, y_232, y_233, y_234, y_235, y_236,
y_237, y_238, y_239, y_240, y_241, y_242, y_243,
y_244: std_logic;
signal x_30, x_31, x_32, x_33, x_34, x_35, x_36, x_37,
x_38, x_39, x_310, x_311, x_312, x_313, x_314, x_315,
x_316, x_317, x_318, x_319, x_320, x_321, x_322,
x_323, x_324, x_325, x_326, x_327, x_328, x_329,
x_330, x_331, x_332, x_333, x_334, x_335, x_336,
x_337, x_338, x_339, x_340, x_341, x_342, x_343,
x_344, x_345, x_346, x_347, x_348: std_logic;
signal y_30, y_31, y_32, y_33, y_34, y_35, y_36, y_37,
y_38, y_39, y_310, y_311, y_312, y_313, y_314, y_315,
y_316, y_317, y_318, y_319, y_320, y_321, y_322,
y_323, y_324, y_325, y_326, y_327, y_328, y_329,
y_330, y_331, y_332, y_333, y_334, y_335, y_336,
y_337, y_338, y_339, y_340, y_341, y_342, y_343,
y_344, y_345, y_346, y_347, y_348: std_logic;
signal x_40, x_41, x_42, x_43, x_44, x_45, x_46, x_47,
x_48, x_49, x_410, x_411, x_412, x_413, x_414, x_415,
x_416, x_417, x_418, x_419, x_420, x_421, x_422,
x_423, x_424, x_425, x_426, x_427, x_428, x_429,
x_430, x_431, x_432, x_433, x_434, x_435, x_436,
x_437, x_438, x_439, x_440, x_441, x_442, x_443,
x_444: std_logic;
signal y_40, y_41, y_42, y_43, y_44, y_45, y_46, y_47,
y_48, y_49, y_410, y_411, y_412, y_413, y_414, y_415,
y_416, y_417, y_418, y_419, y_420, y_421, y_422,
y_423, y_424, y_425, y_426, y_427, y_428, y_429,
y_430, y_431, y_432, y_433, y_434, y_435, y_436,
```

```
y_437, y_438, y_439, y_440, y_441, y_442, y_443,
y_444: std_logic;
signal x_50, x_51, x_52, x_53, x_54, x_55, x_56, x_57,
x_58, x_59, x_510, x_511, x_512, x_513, x_514, x_515,
x_516, x_517, x_518, x_519, x_520, x_521, x_522,
x_523, x_524, x_525, x_526, x_527, x_528, x_529,
x_530, x_531, x_532, x_533, x_534, x_535, x_536,
x_537, x_538, x_539, x_540, x_541, x_542, x_543,
x_544, x_545, x_546, x_547, x_548, x_549, x_550,
x_551: std_logic;
signal y_50, y_51, y_52, y_53, y_54, y_55, y_56, y_57,
y_58, y_59, y_510, y_511, y_512, y_513, y_514, y_515,
y_516, y_517, y_518, y_519, y_520, y_521, y_522,
y_523, y_524, y_525, y_526, y_527, y_528, y_529,
y_530, y_531, y_532, y_533, y_534, y_535, y_536,
y_537, y_538, y_539, y_540, y_541, y_542, y_543,
y_544, y_545, y_546, y_547, y_548, y_549, y_550,
y_551: std_logic;
signal x_60, x_61, x_62, x_63, x_64, x_65, x_66, x_67,
x_68, x_69, x_610, x_611, x_612, x_613, x_614, x_615,
x_616, x_617, x_618, x_619, x_620, x_621, x_622,
x_623, x_624, x_625, x_626, x_627, x_628, x_629,
x_630, x_631, x_632, x_633, x_634, x_635, x_636,
x_637, x_638, x_639, x_640, x_641, x_642, x_643,
x_644, x_645, x_646, x_647: std_logic;
signal y_60, y_61, y_62, y_63, y_64, y_65, y_66, y_67,
y_68, y_69, y_610, y_611, y_612, y_613, y_614, y_615,
y_616, y_617, y_618, y_619, y_620, y_621, y_622,
y_623, y_624, y_625, y_626, y_627, y_628, y_629,
y_630, y_631, y_632, y_633, y_634, y_635, y_636,
y_637, y_638, y_639, y_640, y_641, y_642, y_643,
y_644, y_645, y_646, y_647: std_logic;
signal x_70, x_71, x_72, x_73, x_74, x_75, x_76, x_77,
x_78, x_79, x_710, x_711, x_712, x_713, x_714, x_715,
x_716, x_717, x_718, x_719, x_720, x_721, x_722,
x_723, x_724, x_725, x_726, x_727, x_728, x_729,
x_730, x_731, x_732, x_733, x_734, x_735, x_736,
x_737, x_738, x_739, x_740, x_741, x_742, x_743,
x_744, x_745, x_746, x_747, x_748: std_logic;
signal y_70, y_71, y_72, y_73, y_74, y_75, y_76, y_77,
y_78, y_79, y_710, y_711, y_712, y_713, y_714, y_715,
y_716, y_717, y_718, y_719, y_720, y_721, y_722,
y_723, y_724, y_725, y_726, y_727, y_728, y_729,
y_730, y_731, y_732, y_733, y_734, y_735, y_736,
y_737, y_738, y_739, y_740, y_741, y_742, y_743,
y_744, y_745, y_746, y_747, y_748: std_logic;

begin
    process(clk_1)
      begin
        if (clk_1='1' and clk_1'event) then
           input <= data_in;
           data_out <= output;
        end if;
    end process;

input0z <=  not input(0);

input1z <=  not input(1);

input2z <=  not input(2);

input3z <=  not input(3);

input4z <=  not input(4);

input5z <=  not input(5);

input6z <=  not input(6);

input7z <=  not input(7);

out_76 <= input(7) nand input(6);

out_75 <= input(7) nand input(5);

out_74 <= input(7) nand input(4);

out_76z <= input(7) nand input6z;

out_75z <= input(7) nand input5z;

out_74z <= input(7) nand input4z;

out_65 <= input(6) nand input(5);

out_64 <= input(6) nand input(4);

out_65z <= input(6) nand input5z;

out_64z <= input(6) nand input4z;

out_54 <= input(5) nand input(4);

out_54z <= input(5) nand input4z;

out_7z6 <= input7z nand input(6);

out_7z5 <= input7z nand input(5);

out_7z4 <= input7z nand input(4);

out_7z6z <= input7z nand input6z;

out_7z5z <= input7z nand input5z;

out_7z4z <= input7z nand input4z;

out_6z5 <= input6z nand input(5);

out_6z4 <= input6z nand input(4);

out_6z5z <= input6z nand input5z;

out_6z4z <= input6z nand input4z;

out_5z4 <= input5z nand input(4);

out_5z4z <= input5z nand input4z;

out_32 <= input(3) nand input(2);

out_31 <= input(3) nand input(1);

out_30 <= input(3) nand input(0);

out_32z <= input(3) nand input2z;

out_31z <= input(3) nand input1z;

out_30z <= input(3) nand input0z;

out_21 <= input(2) nand input(1);

out_20 <= input(2) nand input(0);

out_21z <= input(2) nand input1z;

out_20z <= input(2) nand input0z;
```

```
out_10 <= input(1) nand input(0);
out_10z <= input(1) nand input0z;
out_3z2 <= input3z nand input(2);
out_3z1 <= input3z nand input(1);
out_3z0 <= input3z nand input(0);
out_3z2z <= input3z nand input2z;
out_3z1z <= input3z nand input1z;
out_3z0z <= input3z nand input0z;
out_2z1 <= input2z nand input(1);
out_2z0 <= input2z nand input(0);
out_2z1z <= input2z nand input1z;
out_2z0z <= input2z nand input0z;
out_1z0 <= input1z nand input(0);
out_1z0z <= input1z nand input0z;
x_00 <= out_7z6z nor input(5) ;
y_00 <= out_3z2 nor out_1z0 ;
x_01 <= x_00;
y_01  <= out_2z1 nor input0z ;
x_02 <= out_7z6z nor input5z ;
y_02 <= out_32 nor out_10 ;
x_03 <= out_7z6z nor input(4) ;
y_03  <= out_3z2z nor input(0) ;
x_04 <= x_03;
y_04  <= out_32z nor input1z ;
x_05 <= out_7z6z nor input4z ;
y_05 <= out_32z nor out_1z0z ;
x_06 <= not out_7z6z ;
y_06  <= out_3z1 nor input(0) ;
x_07 <= out_7z6 nor out_5z4z ;
y_07  <= out_3z1z nor input(0) ;
x_08 <= out_7z6 nor out_5z4 ;
y_08 <= not out_10 ;
x_09 <= out_7z6 nor input(5) ;
y_09  <= out_2z1z nor input0z ;
x_010 <= out_7z6 nor out_54z ;
y_010  <= out_32z nor input(1) ;
x_011 <= out_7z6 nor input5z ;
y_011 <= not out_3z0 ;
x_012 <= out_7z6 nor input4z ;
y_012  <= out_3z2z nor input(1) ;
x_013 <= out_7z5z nor input4z ;
y_013 <= out_3z2 nor out_10z ;
x_014 <= out_7z5 nor input(4) ;
y_014  <= out_3z1z nor input0z ;
x_015 <= x_014;
y_015  <= out_21 nor input(0) ;
x_016 <= out_7z5 nor input4z ;
y_016 <= out_32 nor out_1z0 ;
x_017 <= not out_7z5 ;
y_017 <= out_3z2z nor out_1z0 ;
x_018 <= out_76z nor out_5z4z ;
y_018 <= y_07;
x_019 <= out_76z nor out_5z4 ;
y_019 <= y_017;
x_020 <= out_76z nor input(5) ;
y_020  <= out_32 nor input1z ;
x_021 <= out_76z nor out_54 ;
y_021  <= out_3z2 nor input0z ;
x_022 <= x_021;
y_022 <= y_07;
x_023 <= out_76 nor out_5z4 ;
y_023  <= out_21z nor input0z ;
x_024 <= out_76 nor out_54z ;
y_024  <= out_2z1z nor input(0) ;
x_025 <= out_76 nor input5z ;
y_025  <= out_32z nor input(0) ;
x_026 <= out_76 nor input4z ;
y_026 <= not out_32z ;
x_027 <= not out_75z ;
y_027 <= out_3z2z nor out_10z ;
```

```
x_028 <= x_027;

y_028 <= out_32 nor out_10z ;

x_029 <= out_75 nor input(4) ;

y_029  <= out_32 nor input0z ;

x_030 <= not out_75 ;

y_030 <= out_3z2 nor out_1z0z ;

x_031 <= not out_74 ;

y_031 <= y_027;

x_032 <= out_6z5z nor input(4) ;

y_032 <= y_021;

x_033 <= not out_6z5 ;

y_033 <= out_32 nor out_1z0z ;

x_034 <= not out_6z4z ;

y_034 <= out_32z nor out_1z0 ;

x_035 <= out_65z nor input(4) ;

y_035  <= out_32 nor input(0) ;

x_036 <= out_65 nor input(4) ;

y_036  <= out_3z2 nor input(1) ;

x_037 <= out_65 nor input4z ;

y_037  <= out_3z2z nor input0z ;

x_038 <= x_037;

y_038 <= y_028;

x_039 <= not out_65 ;

y_039 <= y_01;

x_040 <= not out_5z4z ;

y_040  <= out_32z nor input0z ;

x_041 <= x_040;

y_041 <= y_028;

x_042 <= x_040;

y_042  <= out_31z nor input0z ;

x_043 <= not out_54 ;

y_043 <= out_3z2z nor out_10 ;

output(0) <= ( x_00 and y_00) or ( x_01 and y_01) or
( x_02 and y_02) or ( x_03 and y_03) or ( x_04 and
y_04) or ( x_05 and y_05) or ( x_06 and y_06) or ( x_07
and y_07) or ( x_08 and y_08) or ( x_09 and y_09) or
( x_010 and y_010) or ( x_011 and y_011) or ( x_012
and y_012) or ( x_013 and y_013) or ( x_014 and
y_014) or ( x_015 and y_015) or ( x_016 and y_016) or
( x_017 and y_017) or ( x_018 and y_018) or ( x_019
and y_019) or ( x_020 and y_020) or ( x_021 and
y_021) or ( x_022 and y_022) or ( x_023 and y_023) or
( x_024 and y_024) or ( x_025 and y_025) or ( x_026
and y_026) or
( x_027 and y_027) or ( x_028 and y_028) or ( x_029
and y_029) or ( x_030 and y_030) or ( x_031 and
y_031) or ( x_032 and y_032) or ( x_033 and y_033) or
( x_034 and y_034) or ( x_035 and y_035) or ( x_036
and y_036) or ( x_037 and y_037) or ( x_038 and
y_038) or ( x_039 and y_039) or ( x_040 and y_040) or
( x_041 and y_041) or ( x_042 and y_042) or ( x_043
and y_043);

……. Similar coding pattern up to output(7)

end structure;
```

VHDL code for *encryption_ao_diff*:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
USE ieee.std_logic_arith.ALL;

LIBRARY vs18sc;
USE vs18sc.ALL;

entity encryption is
Port (
     data_in : in std_logic_vector(7 downto 0);
     clk_1 : in std_logic;
     data_outz : out std_logic_vector(7 downto 0);
     data_out : out std_logic_vector(7 downto 0)
);
end encryption;

architecture structure of encryption is
 component and_or0
  port (
    x2 : in std_logic;
    x4 : in std_logic;
    x5 : in std_logic;
    x6 : in std_logic;
    x7 : in std_logic;
    x8 : in std_logic;
    x9 : in std_logic;
    x10 : in std_logic;
    x11 : in std_logic;
    x12 : in std_logic;
    x13 : in std_logic;
    x14 : in std_logic;
    x16 : in std_logic;
    x17 : in std_logic;
    x19 : in std_logic;
    x21 : in std_logic;
    x22 : in std_logic;
    x23 : in std_logic;
    x27 : in std_logic;
    x28 : in std_logic;
    x29 : in std_logic;
    x30 : in std_logic;
    x31 : in std_logic;
    x32 : in std_logic;
```

```
x34 : in std_logic;
x35 : in std_logic;
x37 : in std_logic;
x38 : in std_logic;
x39 : in std_logic;
x40 : in std_logic;
x41 : in std_logic;
x43 : in std_logic;
x44 : in std_logic;
x47 : in std_logic;
x48 : in std_logic;
x49 : in std_logic;
x50 : in std_logic;
x52 : in std_logic;
x53 : in std_logic;
x54 : in std_logic;
x55 : in std_logic;
x56 : in std_logic;
x57 : in std_logic;
x58 : in std_logic;
x59 : in std_logic;
x60 : in std_logic;
x63 : in std_logic;
x66 : in std_logic;
x67 : in std_logic;
x68 : in std_logic;
x72 : in std_logic;
x73 : in std_logic;
x76 : in std_logic;
y0 : in std_logic;
y1 : in std_logic;
y2 : in std_logic;
y3 : in std_logic;
y4 : in std_logic;
y5 : in std_logic;
y6 : in std_logic;
y7 : in std_logic;
y8 : in std_logic;
y9 : in std_logic;
y10 : in std_logic;
y11 : in std_logic;
y12 : in std_logic;
y13 : in std_logic;
y14 : in std_logic;
y15 : in std_logic;
y16 : in std_logic;
y17 : in std_logic;
y18 : in std_logic;
y19 : in std_logic;
y20 : in std_logic;
y21 : in std_logic;
y22 : in std_logic;
y23 : in std_logic;
y24 : in std_logic;
y25 : in std_logic;
y26 : in std_logic;
y27 : in std_logic;
y28 : in std_logic;
y29 : in std_logic;
y30 : in std_logic;
y31 : in std_logic;
y32 : in std_logic;
y33 : in std_logic;
y34 : in std_logic;
y35 : in std_logic;
y36 : in std_logic;
y37 : in std_logic;
y38 : in std_logic;
y39 : in std_logic;
y40 : in std_logic;
y41 : in std_logic;
y42 : in std_logic;
y43 : in std_logic;
y52 : in std_logic;
y53 : in std_logic;
y54 : in std_logic;
y55 : in std_logic;
y56 : in std_logic;
y57 : in std_logic;
y58 : in std_logic;
y59 : in std_logic;
y60 : in std_logic;
y61 : in std_logic;
y62 : in std_logic;
y63 : in std_logic;
y64 : in std_logic;
y65 : in std_logic;
y66 : in std_logic;
y67 : in std_logic;
y68 : in std_logic;
y69 : in std_logic;
y70 : in std_logic;
y71 : in std_logic;
y72 : in std_logic;
y73 : in std_logic;
y74 : in std_logic;
y75 : in std_logic;
y76 : in std_logic;
y77 : in std_logic;
y78 : in std_logic;
y79 : in std_logic;
y80 : in std_logic;
y81 : in std_logic;
y82 : in std_logic;
y83 : in std_logic;
y84 : in std_logic;
y85 : in std_logic;
y86 : in std_logic;
y87 : in std_logic;
y88 : in std_logic;
y89 : in std_logic;
y90 : in std_logic;
y91 : in std_logic;
y92 : in std_logic;
y93 : in std_logic;
y94 : in std_logic;
y95 : in std_logic;
y96 : in std_logic;
y97 : in std_logic;
y98 : in std_logic;
y99 : in std_logic;
equ : in std_logic;
out_t : out std_logic;
out_z : out std_logic);
end component;

…….. Similar coding pattern up to and_or7

signal input: std_logic_vector(7 downto 0);
signal output: std_logic_vector(7 downto 0);
signal outputz: std_logic_vector(7 downto 0);
signal input0z, input1z, input2z, input3z, input4z,
input5z, input6z, input7z: STD_LOGIC;
signal out_76, out_75, out_74, out_76z, out_75z,
out_74z, out_65, out_64, out_65z, out_64z, out_54,
out_54z, out_7z6, out_7z5, out_7z4, out_7z6z,
out_7z5z, out_7z4z, out_6z5, out_6z4, out_6z5z,
out_6z4z, out_5z4, out_5z4z : STD_LOGIC;
signal out_32, out_31, out_30, out_32z, out_31z,
out_30z, out_21, out_20, out_21z, out_20z, out_10,
```

```
out_10z, out_3z2, out_3z1, out_3z0, out_3z2z,
out_3z1z, out_3z0z, out_2z1, out_2z0, out_2z1z,
out_2z0z, out_1z0, out_1z0z : STD_LOGIC;
signal x_02, x_04, x_05, x_06, x_07, x_08, x_09, x_010,
x_011, x_012, x_013, x_014, x_016, x_017, x_019,
x_021, x_022, x_023, x_027, x_028, x_029, x_030,
x_031, x_032, x_034, x_035, x_037, x_038, x_039,
x_040, x_041, x_043, x_044, x_047, x_048, x_049,
x_050, x_052, x_053, x_054, x_055, x_056, x_057,
x_058, x_059, x_060, x_063, x_066, x_067, x_068,
x_072, x_073, x_076 : std_logic;
signal y_00, y_01, y_02, y_03, y_04, y_05, y_06, y_07,
y_08, y_09, y_010, y_011, y_012, y_013, y_014, y_015,
y_016, y_017, y_018, y_019, y_020, y_021, y_022,
y_023, y_024, y_025, y_026, y_027, y_028, y_029,
y_030, y_031, y_032, y_033, y_034, y_035, y_036,
y_037, y_038, y_039, y_040, y_041, y_042, y_043,
y_052, y_053, y_054, y_055, y_056, y_057, y_058,
y_059, y_060, y_061, y_062, y_063, y_064, y_065,
y_066, y_067, y_068, y_069, y_070, y_071, y_072,
y_073, y_074, y_075, y_076, y_077, y_078, y_079,
y_080, y_081, y_082, y_083, y_084, y_085, y_086,
y_087, y_088, y_089, y_090, y_091, y_092, y_093,
y_094, y_095, y_096, y_097, y_098, y_099: std_logic;

…… Similar coding pattern up to x_7 and y_7

begin
    process(clk_1)
      begin
        if (clk_1='1' and clk_1'event) then
           input <= data_in;
           data_out <= output;
           data_outz <= outputz;
        end if;
    end process;

input0z <=  not input(0);

input1z <=  not input(1);

input2z <=  not input(2);

input3z <=  not input(3);

input4z <=  not input(4);

input5z <=  not input(5);

input6z <=  not input(6);

input7z <=  not input(7);

out_76 <= input(7) nand input(6);

out_75 <= input(7) nand input(5);

out_74 <= input(7) nand input(4);

out_76z <= input(7) nand input6z;

out_75z <= input(7) nand input5z;

out_74z <= input(7) nand input4z;

out_65 <= input(6) nand input(5);

out_64 <= input(6) nand input(4);

out_65z <= input(6) nand input5z;

out_64z <= input(6) nand input4z;

out_54 <= input(5) nand input(4);

out_54z <= input(5) nand input4z;

out_7z6 <= input7z nand input(6);

out_7z5 <= input7z nand input(5);

out_7z4 <= input7z nand input(4);

out_7z6z <= input7z nand input6z;

out_7z5z <= input7z nand input5z;

out_7z4z <= input7z nand input4z;

out_6z5 <= input6z nand input(5);

out_6z4 <= input6z nand input(4);

out_6z5z <= input6z nand input5z;

out_6z4z <= input6z nand input4z;

out_5z4 <= input5z nand input(4);

out_5z4z <= input5z nand input4z;

out_32 <= input(3) nand input(2);

out_31 <= input(3) nand input(1);

out_30 <= input(3) nand input(0);

out_32z <= input(3) nand input2z;

out_31z <= input(3) nand input1z;

out_30z <= input(3) nand input0z;

out_21 <= input(2) nand input(1);

out_20 <= input(2) nand input(0);

out_21z <= input(2) nand input1z;

out_20z <= input(2) nand input0z;

out_10 <= input(1) nand input(0);

out_10z <= input(1) nand input0z;

out_3z2 <= input3z nand input(2);

out_3z1 <= input3z nand input(1);

out_3z0 <= input3z nand input(0);

out_3z2z <= input3z nand input2z;

out_3z1z <= input3z nand input1z;

out_3z0z <= input3z nand input0z;

out_2z1 <= input2z nand input(1);

out_2z0 <= input2z nand input(0);
```

```
out_2z1z <= input2z nand input1z;
out_2z0z <= input2z nand input0z;
out_1z0 <= input1z nand input(0);
out_1z0z <= input1z nand input0z;
x_02 <= out_7z6z nor input(5) ;
y_00 <= out_3z2 nor out_1z0 ;
y_01  <= out_2z1 nor input0z ;
x_05 <= out_7z6z nor input5z ;
y_02 <= out_32 nor out_10 ;
x_06 <= out_7z6z nor input(4) ;
y_03  <= out_3z2z nor input(0) ;
y_04  <= out_32z nor input1z ;
x_07 <= out_7z6z nor input4z ;
y_05 <= out_32z nor out_1z0z ;
x_08 <= not out_7z6z ;
y_06  <= out_3z1 nor input(0) ;
x_09 <= out_7z6 nor out_5z4z ;
y_07  <= out_3z1z nor input(0) ;
x_010 <= out_7z6 nor out_5z4 ;
y_08 <= not out_10 ;
x_011 <= out_7z6 nor input(5) ;
y_09  <= out_2z1z nor input0z ;
x_012 <= out_7z6 nor out_54z ;
y_010  <= out_32z nor input(1) ;
x_014 <= out_7z6 nor input5z ;
y_011 <= not out_3z0 ;
x_016 <= out_7z6 nor input4z ;
y_012  <= out_3z2z nor input(1) ;
x_019 <= out_7z5z nor input4z ;
y_013 <= out_3z2 nor out_10z ;
x_021 <= out_7z5 nor input(4) ;
y_014  <= out_3z1z nor input0z ;
y_015  <= out_21 nor input(0) ;
x_022 <= out_7z5 nor input4z ;
y_016 <= out_32 nor out_1z0 ;
x_023 <= not out_7z5 ;
y_017 <= out_3z2z nor out_1z0 ;
x_027 <= out_76z nor out_5z4z ;
y_018 <= y_07;
x_028 <= out_76z nor out_5z4 ;
y_019 <= y_017;
x_029 <= out_76z nor input(5) ;
y_020  <= out_32 nor input1z ;
x_031 <= out_76z nor out_54 ;
y_021  <= out_3z2 nor input0z ;
y_022 <= y_07;
x_037 <= out_76 nor out_5z4 ;
y_023  <= out_21z nor input0z ;
x_039 <= out_76 nor out_54z ;
y_024  <= out_2z1z nor input(0) ;
x_041 <= out_76 nor input5z ;
y_025  <= out_32z nor input(0) ;
x_043 <= out_76 nor input4z ;
y_026 <= not out_32z ;
x_047 <= not out_75z ;
y_027 <= out_3z2z nor out_10z ;
y_028 <= out_32 nor out_10z ;
x_048 <= out_75 nor input(4) ;
y_029  <= out_32 nor input0z ;
x_050 <= not out_75 ;
y_030 <= out_3z2 nor out_1z0z ;
x_052 <= not out_74 ;
y_031 <= y_027;
x_054 <= out_6z5z nor input(4) ;
y_032 <= y_021;
x_059 <= not out_6z5 ;
y_033 <= out_32 nor out_1z0z ;
x_060 <= not out_6z4z ;
y_034 <= out_32z nor out_1z0 ;
x_063 <= out_65z nor input(4) ;
```

```
y_035  <= out_32 nor input(0) ;
x_066 <= out_65 nor input(4) ;
y_036  <= out_3z2 nor input(1) ;
x_067 <= out_65 nor input4z ;
y_037  <= out_3z2z nor input0z ;
y_038 <= y_028;
x_068 <= not out_65 ;
y_039 <= y_01;
x_072 <= not out_5z4z ;
y_040  <= out_32z nor input0z ;
y_041 <= y_028;
y_042  <= out_31z nor input0z ;
x_076 <= not out_54 ;
y_043 <= out_3z2z nor out_10 ;
y_052 <= y_017;
y_053 <= y_02;
x_04 <= out_7z6z nor out_54 ;
y_054 <= y_021;
y_055 <= y_07;
y_056 <= y_030;
y_057 <= not out_3z1 ;
y_058 <= not out_21z ;
y_059 <= y_03;
x_013 <= out_7z6 nor out_54 ;
y_060  <= out_3z2 nor input(0) ;
y_061 <= y_02;
y_062  <= out_31z nor input(0) ;
x_017 <= not out_7z6 ;
y_063  <= out_2z1 nor input(0) ;
y_064  <= out_31 nor input(0) ;
y_065 <= y_016;
y_066 <= y_034;
y_067 <= y_058;
x_030 <= out_76z nor out_54z ;
y_068 <= y_011;
y_069 <= not out_2z0z ;
x_032 <= out_76z nor input5z ;
y_070 <= y_017;
y_071 <= y_04;
x_034 <= out_76z nor input4z ;
y_072 <= y_026;
x_035 <= not out_76z ;
y_073 <= y_025;
x_038 <= out_76 nor input(5) ;
y_074 <= y_037;
y_075 <= y_07;
y_076 <= y_09;
x_040 <= out_76 nor out_54 ;
y_077 <= not out_20 ;
y_078 <= out_3z2z nor out_1z0z ;
y_079  <= out_21 nor input0z ;
x_044 <= not out_76 ;
y_080  <= out_3z2 nor input1z ;
y_081 <= y_043;
y_082 <= y_06;
y_083 <= y_015;
x_049 <= out_75 nor input4z ;
y_084 <= y_029;
x_053 <=  input(7);
y_085 <= y_013;
y_086 <= y_017;
x_055 <= out_6z5z nor input4z ;
y_087 <= out_3z2 nor out_10 ;
y_088 <= y_07;
y_089 <= y_042;
x_056 <= not out_6z5z ;
y_090 <= y_033;
x_057 <= out_6z5 nor input(4) ;
y_091  <= out_3z1 nor input0z ;
x_058 <= out_6z5 nor input4z ;
```

```
y_092 <= y_04;

y_093 <= y_064;

y_094 <= y_05;

y_095 <= y_021;

y_096 <= y_025;

y_097 <= y_079;

y_098 <= y_033;

x_073 <= not out_5z4 ;

y_099  <= out_21z nor input(0) ;

ao0: and_or0
port map (x_02, x_04, x_05, x_06, x_07, x_08, x_09,
x_010, x_011, x_012, x_013, x_014, x_016, x_017,
x_019, x_021, x_022, x_023, x_027, x_028, x_029,
x_030, x_031, x_032, x_034, x_035, x_037, x_038,
x_039, x_040, x_041, x_043, x_044, x_047, x_048,
x_049, x_050, x_052, x_053, x_054, x_055, x_056,
x_057, x_058, x_059, x_060, x_063, x_066, x_067,
x_068, x_072, x_073, x_076,
y_00, y_01, y_02, y_03, y_04, y_05, y_06, y_07, y_08,
y_09, y_010, y_011, y_012, y_013, y_014, y_015,
y_016, y_017, y_018, y_019, y_020, y_021, y_022,
y_023, y_024, y_025, y_026, y_027, y_028, y_029,
y_030, y_031, y_032, y_033, y_034, y_035, y_036,
y_037, y_038, y_039, y_040, y_041, y_042, y_043,
y_052, y_053, y_054, y_055, y_056, y_057, y_058,
y_059, y_060, y_061, y_062, y_063, y_064, y_065,
y_066, y_067, y_068, y_069, y_070, y_071, y_072,
y_073, y_074, y_075, y_076, y_077, y_078, y_079,
y_080, y_081, y_082, y_083, y_084, y_085, y_086,
y_087, y_088, y_089, y_090, y_091, y_092, y_093,
y_094, y_095, y_096, y_097, y_098, y_099,  clk_1,
output(0), outputz(0) );

…… Similar coding pattern up to output(7)

end structure;
```

VHDL code for *encryption_ao_domino*

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
USE ieee.std_logic_arith.ALL;

LIBRARY vs18sc;
USE vs18sc.ALL;

entity encryption3 is
Port (
     data_in : in std_logic_vector(7 downto 0);
     clk_1 : in std_logic;
     data_out : out std_logic_vector(7 downto 0)
);
end encryption3;

architecture structure of encryption3 is
 component and_or0
  port (
    x2 : in std_logic;
    x5 : in std_logic;
    x6 : in std_logic;
    x7 : in std_logic;
    x8 : in std_logic;
    x9 : in std_logic;
    x10 : in std_logic;
    x11 : in std_logic;
    x12 : in std_logic;
    x14 : in std_logic;
    x16 : in std_logic;
    x19 : in std_logic;
    x21 : in std_logic;
    x22 : in std_logic;
    x23 : in std_logic;
    x27 : in std_logic;
    x28 : in std_logic;
    x29 : in std_logic;
    x31 : in std_logic;
    x37 : in std_logic;
    x39 : in std_logic;
    x41 : in std_logic;
    x43 : in std_logic;
    x47 : in std_logic;
    x48 : in std_logic;
    x50 : in std_logic;
    x52 : in std_logic;
    x54 : in std_logic;
    x59 : in std_logic;
    x60 : in std_logic;
    x63 : in std_logic;
    x66 : in std_logic;
    x67 : in std_logic;
    x68 : in std_logic;
    x72 : in std_logic;
    x76 : in std_logic;
    y0 : in std_logic;
    y1 : in std_logic;
    y2 : in std_logic;
    y3 : in std_logic;
    y4 : in std_logic;
    y5 : in std_logic;
    y6 : in std_logic;
    y7 : in std_logic;
    y8 : in std_logic;
    y9 : in std_logic;
    y10 : in std_logic;
    y11 : in std_logic;
    y12 : in std_logic;
    y13 : in std_logic;
    y14 : in std_logic;
    y15 : in std_logic;
    y16 : in std_logic;
    y17 : in std_logic;
    y18 : in std_logic;
    y19 : in std_logic;
    y20 : in std_logic;
    y21 : in std_logic;
    y22 : in std_logic;
    y23 : in std_logic;
    y24 : in std_logic;
    y25 : in std_logic;
    y26 : in std_logic;
    y27 : in std_logic;
    y28 : in std_logic;
    y29 : in std_logic;
    y30 : in std_logic;
    y31 : in std_logic;
    y32 : in std_logic;
```

```
    y33 : in std_logic;
    y34 : in std_logic;
    y35 : in std_logic;
    y36 : in std_logic;
    y37 : in std_logic;
    y38 : in std_logic;
    y39 : in std_logic;
    y40 : in std_logic;
    y41 : in std_logic;
    y42 : in std_logic;
    y43 : in std_logic;
    equ : in std_logic;
    out_t : out std_logic);
 end component;

 …….. Similar coding pattern up to and_or7


signal input: std_logic_vector(7 downto 0);
signal output: std_logic_vector(7 downto 0);

signal input0z, input1z, input2z, input3z, input4z,
input5z, input6z, input7z: STD_LOGIC;
signal out_76, out_75, out_74, out_76z, out_75z,
out_74z, out_65, out_64, out_65z, out_64z, out_54,
out_54z, out_7z6, out_7z5, out_7z4, out_7z6z,
out_7z5z, out_7z4z, out_6z5, out_6z4, out_6z5z,
out_6z4z, out_5z4, out_5z4z : STD_LOGIC;
signal out_32, out_31, out_30, out_32z, out_31z,
out_30z, out_21, out_20, out_21z, out_20z, out_10,
out_10z, out_3z2, out_3z1, out_3z0, out_3z2z,
out_3z1z, out_3z0z, out_2z1, out_2z0, out_2z1z,
out_2z0z, out_1z0, out_1z0z : STD_LOGIC;
signal x_02, x_05, x_06, x_07, x_08, x_09, x_010,
x_011, x_012, x_014, x_016, x_019, x_021, x_022,
x_023, x_027, x_028, x_029, x_031, x_037, x_039,
x_041, x_043, x_047, x_048, x_050, x_052, x_054,
x_059, x_060, x_063, x_066, x_067, x_068, x_072,
x_076 : std_logic;
signal y_00, y_01, y_02, y_03, y_04, y_05, y_06, y_07,
y_08, y_09, y_010, y_011, y_012, y_013, y_014, y_015,
y_016, y_017, y_018, y_019, y_020, y_021, y_022,
y_023, y_024, y_025, y_026, y_027, y_028, y_029,
y_030, y_031, y_032, y_033, y_034, y_035, y_036,
y_037, y_038, y_039, y_040, y_041, y_042, y_043,
y_051: std_logic;

…….. Similar coding pattern up to x_7 and y_7

begin
     process(clk_1)
       begin
         if (clk_1='1' and clk_1'event) then
            input <= data_in;
            data_out <= output;

         end if;
     end process;

input0z <=  not input(0);

input1z <=  not input(1);

input2z <=  not input(2);

input3z <=  not input(3);

input4z <=  not input(4);

input5z <=  not input(5);

input6z <=  not input(6);

input7z <=  not input(7);

out_76 <= input(7) nand input(6);

out_75 <= input(7) nand input(5);

out_74 <= input(7) nand input(4);

out_76z <= input(7) nand input6z;

out_75z <= input(7) nand input5z;

out_74z <= input(7) nand input4z;

out_65 <= input(6) nand input(5);

out_64 <= input(6) nand input(4);

out_65z <= input(6) nand input5z;

out_64z <= input(6) nand input4z;

out_54 <= input(5) nand input(4);

out_54z <= input(5) nand input4z;

out_7z6 <= input7z nand input(6);

out_7z5 <= input7z nand input(5);

out_7z4 <= input7z nand input(4);

out_7z6z <= input7z nand input6z;

out_7z5z <= input7z nand input5z;

out_7z4z <= input7z nand input4z;

out_6z5 <= input6z nand input(5);

out_6z4 <= input6z nand input(4);

out_6z5z <= input6z nand input5z;

out_6z4z <= input6z nand input4z;

out_5z4 <= input5z nand input(4);

out_5z4z <= input5z nand input4z;

out_32 <= input(3) nand input(2);

out_31 <= input(3) nand input(1);

out_30 <= input(3) nand input(0);

out_32z <= input(3) nand input2z;

out_31z <= input(3) nand input1z;

out_30z <= input(3) nand input0z;

out_21 <= input(2) nand input(1);

out_20 <= input(2) nand input(0);

out_21z <= input(2) nand input1z;
```

```
out_20z <= input(2) nand input0z;
out_10 <= input(1) nand input(0);
out_10z <= input(1) nand input0z;
out_3z2 <= input3z nand input(2);
out_3z1 <= input3z nand input(1);
out_3z0 <= input3z nand input(0);
out_3z2z <= input3z nand input2z;
out_3z1z <= input3z nand input1z;
out_3z0z <= input3z nand input0z;
out_2z1 <= input2z nand input(1);
out_2z0 <= input2z nand input(0);
out_2z1z <= input2z nand input1z;
out_2z0z <= input2z nand input0z;
out_1z0 <= input1z nand input(0);
out_1z0z <= input1z nand input0z;
x_02 <= out_7z6z nor input(5) ;
y_00 <= out_3z2 nor out_1z0 ;
y_01  <= out_2z1 nor input0z ;
x_05 <= out_7z6z nor input5z ;
y_02 <= out_32 nor out_10 ;
x_06 <= out_7z6z nor input(4) ;
y_03  <= out_3z2z nor input(0) ;
y_04  <= out_32z nor input1z ;
x_07 <= out_7z6z nor input4z ;
y_05 <= out_32z nor out_1z0z ;
x_08 <= not out_7z6z ;
y_06  <= out_3z1 nor input(0) ;
x_09 <= out_7z6 nor out_5z4z ;
y_07  <= out_3z1z nor input(0) ;
x_010 <= out_7z6 nor out_5z4 ;
y_08 <= not out_10 ;
x_011 <= out_7z6 nor input(5) ;
y_09  <= out_2z1z nor input0z ;
x_012 <= out_7z6 nor out_54z ;
y_010  <= out_32z nor input(1) ;
x_014 <= out_7z6 nor input5z ;
y_011 <= not out_3z0 ;
x_016 <= out_7z6 nor input4z ;
y_012  <= out_3z2z nor input(1) ;
x_019 <= out_7z5z nor input4z ;
y_013 <= out_3z2 nor out_10z ;
x_021 <= out_7z5 nor input(4) ;
y_014  <= out_3z1z nor input0z ;
y_015  <= out_21 nor input(0) ;
x_022 <= out_7z5 nor input4z ;
y_016 <= out_32 nor out_1z0 ;
x_023 <= not out_7z5 ;
y_017 <= out_3z2z nor out_1z0 ;
x_027 <= out_76z nor out_5z4z ;
y_018 <= y_07;
x_028 <= out_76z nor out_5z4 ;
y_019 <= y_017;
x_029 <= out_76z nor input(5) ;
y_020  <= out_32 nor input1z ;
x_031 <= out_76z nor out_54 ;
y_021  <= out_3z2 nor input0z ;
y_022 <= y_07;
x_037 <= out_76 nor out_5z4 ;
y_023  <= out_21z nor input0z ;
x_039 <= out_76 nor out_54z ;
y_024  <= out_2z1z nor input(0) ;
x_041 <= out_76 nor input5z ;
y_025  <= out_32z nor input(0) ;
x_043 <= out_76 nor input4z ;
y_026 <= not out_32z ;
x_047 <= not out_75z ;
y_027 <= out_3z2z nor out_10z ;
y_028 <= out_32 nor out_10z ;
x_048 <= out_75 nor input(4) ;
y_029  <= out_32 nor input0z ;
```

```
x_050 <= not out_75 ;

y_030 <= out_3z2 nor out_1z0z ;

x_052 <= not out_74 ;

y_031 <= y_027;

x_054 <= out_6z5z nor input(4) ;

y_032 <= y_021;

x_059 <= not out_6z5 ;

y_033 <= out_32 nor out_1z0z ;

x_060 <= not out_6z4z ;

y_034 <= out_32z nor out_1z0 ;

x_063 <= out_65z nor input(4) ;

y_035  <= out_32 nor input(0) ;

x_066 <= out_65 nor input(4) ;

y_036  <= out_3z2 nor input(1) ;

x_067 <= out_65 nor input4z ;

y_037  <= out_3z2z nor input0z ;

y_038 <= y_028;

x_068 <= not out_65 ;

y_039 <= y_01;

x_072 <= not out_5z4z ;

y_040  <= out_32z nor input0z ;

y_041 <= y_028;

y_042  <= out_31z nor input0z ;

x_076 <= not out_54 ;

y_043 <= out_3z2z nor out_10 ;

ao0: and_or0
port map (x_02, x_05, x_06, x_07, x_08, x_09, x_010,
x_011, x_012, x_014, x_016, x_019, x_021, x_022,
x_023, x_027, x_028, x_029, x_031, x_037, x_039,
x_041, x_043, x_047, x_048, x_050, x_052, x_054,
x_059, x_060, x_063, x_066, x_067, x_068, x_072,
x_076,
y_00, y_01, y_02, y_03, y_04, y_05, y_06, y_07, y_08,
y_09, y_010, y_011, y_012, y_013, y_014, y_015,
y_016, y_017, y_018, y_019, y_020, y_021, y_022,
y_023, y_024, y_025, y_026, y_027, y_028, y_029,
y_030, y_031, y_032, y_033, y_034, y_035, y_036,
y_037, y_038, y_039, y_040, y_041, y_042, y_043,
clk_1, output(0));

…….. Similar coding pattern up to output(7)

end structure;
```

Skill code for and_or_diff0:

```
library="sfulib3"

procedure(and_or0( )
let( ( cvw cnm x y x1 y1 I0 I1 rod1 rod2 pt1 pt2 )

 sprintf(cnm "and_or0" )

 cell=dbOpenCellViewByType(library cnm "layout"
"maskLayout" "w")
 printf("Cellname: %s" cnm)

x = 0
y = 0

cvw = dbOpenCellViewByType( library "feTop" "layout"
"maskLayout" "r" )

I0 =  dbCreateInst(cell cvw "I0" list( R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Top" "layout"
"maskLayout" "r" )

I2 =  dbCreateInst(cell cvw "I2" list(4*R->Xsp+R->Xsp
0 ) "R0" )
I5 =  dbCreateInst(cell cvw "I5" list(12*R->Xsp+R->Xsp
0 ) "MY" )
I6 =  dbCreateInst(cell cvw "I6" list(12*R->Xsp+R->Xsp
0 ) "R0" )
I7 =  dbCreateInst(cell cvw "I7" list(16*R->Xsp+R->Xsp
0 ) "MY" )
I8 =  dbCreateInst(cell cvw "I8" list(16*R->Xsp+R->Xsp
0 ) "R0" )
I9 =  dbCreateInst(cell cvw "I9" list(20*R->Xsp+R->Xsp
0 ) "MY" )
I10 =  dbCreateInst(cell cvw "I10" list(20*R->Xsp+R-
>Xsp 0 ) "R0" )
I11 =  dbCreateInst(cell cvw "I11" list(24*R->Xsp+R-
>Xsp 0 ) "MY" )
I12 =  dbCreateInst(cell cvw "I12" list(24*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eTop" "layout"
"maskLayout" "r" )

I3 =  dbCreateInst(cell cvw "I3" list(6*R->Xsp+R->Xsp
0 ) "R0" )
I13 =  dbCreateInst(cell cvw "I13" list(26*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop2" "layout"
"maskLayout" "r" )

I4 =  dbCreateInst(cell cvw "I4" list(8*R->Xsp+R->Xsp
0 ) "R0" )

 dbClose( cvw )
```

```
cvw = dbOpenCellViewByType( library "sTop" "layout"
"maskLayout" "r" )

I1 =  dbCreateInst(cell cvw "I1" list(4*R->Xsp+R->Xsp
0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "pSub" "layout"
"maskLayout" "r" )

I198 =  dbCreateInst(cell cvw "I198" list( 28*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Top" "layout"
"maskLayout" "r" )

I14 =  dbCreateInst(cell cvw "I14" list(32*R->Xsp+R-
>Xsp 0 ) "R0" )
I16 =  dbCreateInst(cell cvw "I16" list(36*R->Xsp+R-
>Xsp 0 ) "R0" )
I19 =  dbCreateInst(cell cvw "I19" list(44*R->Xsp+R-
>Xsp 0 ) "MY" )
I21 =  dbCreateInst(cell cvw "I21" list(48*R->Xsp+R-
>Xsp 0 ) "MY" )
I22 =  dbCreateInst(cell cvw "I22" list(48*R->Xsp+R-
>Xsp 0 ) "R0" )
I23 =  dbCreateInst(cell cvw "I23" list(52*R->Xsp+R-
>Xsp 0 ) "MY" )
I27 =  dbCreateInst(cell cvw "I27" list(60*R->Xsp+R-
>Xsp 0 ) "MY" )
I28 =  dbCreateInst(cell cvw "I28" list(60*R->Xsp+R-
>Xsp 0 ) "R0" )
I29 =  dbCreateInst(cell cvw "I29" list(64*R->Xsp+R-
>Xsp 0 ) "MY" )
I31 =  dbCreateInst(cell cvw "I31" list(68*R->Xsp+R-
>Xsp 0 ) "MY" )
I37 =  dbCreateInst(cell cvw "I37" list(80*R->Xsp+R-
>Xsp 0 ) "MY" )
I39 =  dbCreateInst(cell cvw "I39" list(84*R->Xsp+R-
>Xsp 0 ) "MY" )
I41 =  dbCreateInst(cell cvw "I41" list(88*R->Xsp+R-
>Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eTop" "layout"
"maskLayout" "r" )

I15 =  dbCreateInst(cell cvw "I15" list(34*R->Xsp+R-
>Xsp 0 ) "R0" )
I17 =  dbCreateInst(cell cvw "I17" list(38*R->Xsp+R-
>Xsp 0 ) "R0" )
I18 =  dbCreateInst(cell cvw "I18" list(40*R->Xsp+R-
>Xsp 0 ) "R0" )
I24 =  dbCreateInst(cell cvw "I24" list(52*R->Xsp+R-
>Xsp 0 ) "R0" )
I25 =  dbCreateInst(cell cvw "I25" list(54*R->Xsp+R-
>Xsp 0 ) "R0" )
I26 =  dbCreateInst(cell cvw "I26" list(56*R->Xsp+R-
>Xsp 0 ) "R0" )
I32 =  dbCreateInst(cell cvw "I32" list(68*R->Xsp+R-
>Xsp 0 ) "R0" )
I33 =  dbCreateInst(cell cvw "I33" list(70*R->Xsp+R-
>Xsp 0 ) "R0" )
I34 =  dbCreateInst(cell cvw "I34" list(72*R->Xsp+R-
>Xsp 0 ) "R0" )
I35 =  dbCreateInst(cell cvw "I35" list(74*R->Xsp+R-
>Xsp 0 ) "R0" )
I36 =  dbCreateInst(cell cvw "I36" list(76*R->Xsp+R-
>Xsp 0 ) "R0" )
I38 =  dbCreateInst(cell cvw "I38" list(80*R->Xsp+R-
>Xsp 0 ) "R0" )
I40 =  dbCreateInst(cell cvw "I40" list(84*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop" "layout"
"maskLayout" "r" )

I20 =  dbCreateInst(cell cvw "I20" list(44*R->Xsp+R-
>Xsp 0 ) "R0" )
I30 =  dbCreateInst(cell cvw "I30" list(64*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "pSub" "layout"
"maskLayout" "r" )

I199 =  dbCreateInst(cell cvw "I199" list( 88*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Top" "layout"
"maskLayout" "r" )

I43 =  dbCreateInst(cell cvw "I43" list(96*R->Xsp+R-
>Xsp 0 ) "MY" )
I47 =  dbCreateInst(cell cvw "I47" list(104*R->Xsp+R-
>Xsp 0 ) "MY" )
I48 =  dbCreateInst(cell cvw "I48" list(104*R->Xsp+R-
>Xsp 0 ) "R0" )
I50 =  dbCreateInst(cell cvw "I50" list(108*R->Xsp+R-
>Xsp 0 ) "R0" )
I52 =  dbCreateInst(cell cvw "I52" list(112*R->Xsp+R-
>Xsp 0 ) "R0" )
I54 =  dbCreateInst(cell cvw "I54" list(116*R->Xsp+R-
>Xsp 0 ) "R0" )
I59 =  dbCreateInst(cell cvw "I59" list(128*R->Xsp+R-
>Xsp 0 ) "MY" )
I60 =  dbCreateInst(cell cvw "I60" list(128*R->Xsp+R-
>Xsp 0 ) "R0" )
I63 =  dbCreateInst(cell cvw "I63" list(136*R->Xsp+R-
>Xsp 0 ) "MY" )
I66 =  dbCreateInst(cell cvw "I66" list(140*R->Xsp+R-
>Xsp 0 ) "R0" )
I67 =  dbCreateInst(cell cvw "I67" list(144*R->Xsp+R-
>Xsp 0 ) "MY" )
I68 =  dbCreateInst(cell cvw "I68" list(144*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eTop" "layout"
"maskLayout" "r" )

I42 =  dbCreateInst(cell cvw "I42" list(92*R->Xsp+R-
>Xsp 0 ) "R0" )
I44 =  dbCreateInst(cell cvw "I44" list(96*R->Xsp+R-
>Xsp 0 ) "R0" )
I45 =  dbCreateInst(cell cvw "I45" list(98*R->Xsp+R-
>Xsp 0 ) "R0" )
I49 =  dbCreateInst(cell cvw "I49" list(106*R->Xsp+R-
>Xsp 0 ) "R0" )
```

```
I51 =  dbCreateInst(cell cvw "I51" list(110*R->Xsp+R->Xsp 0 ) "R0" )
I53 =  dbCreateInst(cell cvw "I53" list(114*R->Xsp+R->Xsp 0 ) "R0" )
I55 =  dbCreateInst(cell cvw "I55" list(118*R->Xsp+R->Xsp 0 ) "R0" )
I56 =  dbCreateInst(cell cvw "I56" list(120*R->Xsp+R->Xsp 0 ) "R0" )
I57 =  dbCreateInst(cell cvw "I57" list(122*R->Xsp+R->Xsp 0 ) "R0" )
I58 =  dbCreateInst(cell cvw "I58" list(124*R->Xsp+R->Xsp 0 ) "R0" )
I61 =  dbCreateInst(cell cvw "I61" list(130*R->Xsp+R->Xsp 0 ) "R0" )
I62 =  dbCreateInst(cell cvw "I62" list(132*R->Xsp+R->Xsp 0 ) "R0" )
I64 =  dbCreateInst(cell cvw "I64" list(136*R->Xsp+R->Xsp 0 ) "R0" )
I69 =  dbCreateInst(cell cvw "I69" list(146*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop" "layout" "maskLayout" "r" )

I46 =  dbCreateInst(cell cvw "I46" list(100*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop2" "layout" "maskLayout" "r" )

I65 =  dbCreateInst(cell cvw "I65" list(140*R->Xsp+R->Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "pSub" "layout" "maskLayout" "r" )

I200 =  dbCreateInst(cell cvw "I200" list( 148*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Top" "layout" "maskLayout" "r" )

I72 =  dbCreateInst(cell cvw "I72" list(156*R->Xsp+R->Xsp 0 ) "R0" )
I76 =  dbCreateInst(cell cvw "I76" list(164*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eTop" "layout" "maskLayout" "r" )

I70 =  dbCreateInst(cell cvw "I70" list(152*R->Xsp+R->Xsp 0 ) "R0" )
I74 =  dbCreateInst(cell cvw "I74" list(160*R->Xsp+R->Xsp 0 ) "R0" )
I75 =  dbCreateInst(cell cvw "I75" list(162*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop" "layout" "maskLayout" "r" )

I71 =  dbCreateInst(cell cvw "I71" list(156*R->Xsp+R->Xsp 0 ) "MY" )
I73 =  dbCreateInst(cell cvw "I73" list(160*R->Xsp+R->Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "feBot" "layout" "maskLayout" "r" )

I100 =  dbCreateInst(cell cvw "I100" list( R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Bot" "layout" "maskLayout" "r" )

I102 =  dbCreateInst(cell cvw "I102" list(4*R->Xsp+R->Xsp 0 ) "R0" )
I104 =  dbCreateInst(cell cvw "I104" list(8*R->Xsp+R->Xsp 0 ) "R0" )
I107 =  dbCreateInst(cell cvw "I107" list(16*R->Xsp+R->Xsp 0 ) "MY" )
I108 =  dbCreateInst(cell cvw "I108" list(16*R->Xsp+R->Xsp 0 ) "R0" )
I109 =  dbCreateInst(cell cvw "I109" list(20*R->Xsp+R->Xsp 0 ) "MY" )
I110 =  dbCreateInst(cell cvw "I110" list(20*R->Xsp+R->Xsp 0 ) "R0" )
I112 =  dbCreateInst(cell cvw "I112" list(24*R->Xsp+R->Xsp 0 ) "R0" )
I113 =  dbCreateInst(cell cvw "I113" list(28*R->Xsp+R->Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I103 =  dbCreateInst(cell cvw "I103" list(6*R->Xsp+R->Xsp 0 ) "R0" )
I105 =  dbCreateInst(cell cvw "I105" list(10*R->Xsp+R->Xsp 0 ) "R0" )
I106 =  dbCreateInst(cell cvw "I106" list(12*R->Xsp+R->Xsp 0 ) "R0" )
I111 =  dbCreateInst(cell cvw "I111" list(22*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sBot" "layout" "maskLayout" "r" )

I101 =  dbCreateInst(cell cvw "I101" list(4*R->Xsp+R->Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Bot" "layout" "maskLayout" "r" )

I114 =  dbCreateInst(cell cvw "I114" list(32*R->Xsp+R->Xsp 0 ) "R0" )
I116 =  dbCreateInst(cell cvw "I116" list(36*R->Xsp+R->Xsp 0 ) "R0" )
```

```
I117 =  dbCreateInst(cell cvw "I117" list(40*R->Xsp+R->Xsp 0 ) "MY" )
I119 =  dbCreateInst(cell cvw "I119" list(44*R->Xsp+R->Xsp 0 ) "MY" )
I121 =  dbCreateInst(cell cvw "I121" list(48*R->Xsp+R->Xsp 0 ) "MY" )
I122 =  dbCreateInst(cell cvw "I122" list(48*R->Xsp+R->Xsp 0 ) "R0" )
I128 =  dbCreateInst(cell cvw "I128" list(60*R->Xsp+R->Xsp 0 ) "R0" )
I130 =  dbCreateInst(cell cvw "I130" list(64*R->Xsp+R->Xsp 0 ) "R0" )
I132 =  dbCreateInst(cell cvw "I132" list(68*R->Xsp+R->Xsp 0 ) "R0" )
I134 =  dbCreateInst(cell cvw "I134" list(72*R->Xsp+R->Xsp 0 ) "R0" )
I135 =  dbCreateInst(cell cvw "I135" list(76*R->Xsp+R->Xsp 0 ) "MY" )
I138 =  dbCreateInst(cell cvw "I138" list(80*R->Xsp+R->Xsp 0 ) "R0" )
I139 =  dbCreateInst(cell cvw "I139" list(84*R->Xsp+R->Xsp 0 ) "MY" )
I140 =  dbCreateInst(cell cvw "I140" list(84*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I115 =  dbCreateInst(cell cvw "I115" list(34*R->Xsp+R->Xsp 0 ) "R0" )
I118 =  dbCreateInst(cell cvw "I118" list(40*R->Xsp+R->Xsp 0 ) "R0" )
I120 =  dbCreateInst(cell cvw "I120" list(44*R->Xsp+R->Xsp 0 ) "R0" )
I123 =  dbCreateInst(cell cvw "I123" list(50*R->Xsp+R->Xsp 0 ) "R0" )
I124 =  dbCreateInst(cell cvw "I124" list(52*R->Xsp+R->Xsp 0 ) "R0" )
I125 =  dbCreateInst(cell cvw "I125" list(54*R->Xsp+R->Xsp 0 ) "R0" )
I126 =  dbCreateInst(cell cvw "I126" list(56*R->Xsp+R->Xsp 0 ) "R0" )
I127 =  dbCreateInst(cell cvw "I127" list(58*R->Xsp+R->Xsp 0 ) "R0" )
I133 =  dbCreateInst(cell cvw "I133" list(70*R->Xsp+R->Xsp 0 ) "R0" )
I136 =  dbCreateInst(cell cvw "I136" list(76*R->Xsp+R->Xsp 0 ) "R0" )
I141 =  dbCreateInst(cell cvw "I141" list(86*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sBot" "layout" "maskLayout" "r" )

I129 =  dbCreateInst(cell cvw "I129" list(64*R->Xsp+R->Xsp 0 ) "MY" )
I131 =  dbCreateInst(cell cvw "I131" list(68*R->Xsp+R->Xsp 0 ) "MY" )
I137 =  dbCreateInst(cell cvw "I137" list(80*R->Xsp+R->Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Bot" "layout" "maskLayout" "r" )

I143 =  dbCreateInst(cell cvw "I143" list(96*R->Xsp+R->Xsp 0 ) "MY" )
I144 =  dbCreateInst(cell cvw "I144" list(96*R->Xsp+R->Xsp 0 ) "R0" )
I147 =  dbCreateInst(cell cvw "I147" list(104*R->Xsp+R->Xsp 0 ) "MY" )
I148 =  dbCreateInst(cell cvw "I148" list(104*R->Xsp+R->Xsp 0 ) "R0" )
I149 =  dbCreateInst(cell cvw "I149" list(108*R->Xsp+R->Xsp 0 ) "MY" )
I153 =  dbCreateInst(cell cvw "I153" list(116*R->Xsp+R->Xsp 0 ) "MY" )
I154 =  dbCreateInst(cell cvw "I154" list(116*R->Xsp+R->Xsp 0 ) "R0" )
I155 =  dbCreateInst(cell cvw "I155" list(120*R->Xsp+R->Xsp 0 ) "MY" )
I156 =  dbCreateInst(cell cvw "I156" list(120*R->Xsp+R->Xsp 0 ) "R0" )
I157 =  dbCreateInst(cell cvw "I157" list(124*R->Xsp+R->Xsp 0 ) "MY" )
I158 =  dbCreateInst(cell cvw "I158" list(124*R->Xsp+R->Xsp 0 ) "R0" )
I160 =  dbCreateInst(cell cvw "I160" list(128*R->Xsp+R->Xsp 0 ) "R0" )
I163 =  dbCreateInst(cell cvw "I163" list(136*R->Xsp+R->Xsp 0 ) "MY" )
I168 =  dbCreateInst(cell cvw "I168" list(144*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I145 =  dbCreateInst(cell cvw "I145" list(98*R->Xsp+R->Xsp 0 ) "R0" )
I150 =  dbCreateInst(cell cvw "I150" list(108*R->Xsp+R->Xsp 0 ) "R0" )
I161 =  dbCreateInst(cell cvw "I161" list(130*R->Xsp+R->Xsp 0 ) "R0" )
I165 =  dbCreateInst(cell cvw "I165" list(138*R->Xsp+R->Xsp 0 ) "R0" )
I166 =  dbCreateInst(cell cvw "I166" list(140*R->Xsp+R->Xsp 0 ) "R0" )
I167 =  dbCreateInst(cell cvw "I167" list(142*R->Xsp+R->Xsp 0 ) "R0" )
I169 =  dbCreateInst(cell cvw "I169" list(146*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sBot" "layout" "maskLayout" "r" )

I142 =  dbCreateInst(cell cvw "I142" list(92*R->Xsp+R->Xsp 0 ) "R0" )
I159 =  dbCreateInst(cell cvw "I159" list(128*R->Xsp+R->Xsp 0 ) "MY" )
I162 =  dbCreateInst(cell cvw "I162" list(132*R->Xsp+R->Xsp 0 ) "R0" )
I164 =  dbCreateInst(cell cvw "I164" list(136*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sBot2" "layout" "maskLayout" "r" )

I146 =  dbCreateInst(cell cvw "I146" list(100*R->Xsp+R->Xsp 0 ) "R0" )
```

```
I151 =  dbCreateInst(cell cvw "I151" list(112*R->Xsp+R->Xsp 0 ) "MY" )
I152 =  dbCreateInst(cell cvw "I152" list(112*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Bot" "layout" "maskLayout" "r" )

I173 =  dbCreateInst(cell cvw "I173" list(160*R->Xsp+R->Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I170 =  dbCreateInst(cell cvw "I170" list(152*R->Xsp+R->Xsp 0 ) "R0" )
I171 =  dbCreateInst(cell cvw "I171" list(154*R->Xsp+R->Xsp 0 ) "R0" )
I172 =  dbCreateInst(cell cvw "I172" list(156*R->Xsp+R->Xsp 0 ) "R0" )
I174 =  dbCreateInst(cell cvw "I174" list(160*R->Xsp+R->Xsp 0 ) "R0" )
I175 =  dbCreateInst(cell cvw "I175" list(162*R->Xsp+R->Xsp 0 ) "R0" )
I176 =  dbCreateInst(cell cvw "I176" list(164*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sBot" "layout" "maskLayout" "r" )


 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Dff" "layout" "maskLayout" "r" )

I201 =  dbCreateInst(cell cvw "I201" list( 166*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

mkm2RodPin(cell "equ" "metal2" "input" list( "none" ) list(169*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "out_t" "metal2" "output" list( "none" ) list(172*R->Xsp+R->Xsp2 5*R->Ysp))
mkm2RodPin(cell "out_z" "metal2" "output" list( "none" ) list(175*R->Xsp+R->Xsp2 5*R->Ysp))
mkPWR(cell "VSS" "metal1" "inputOutput" list(80*R->Xsp R->HNsub) list(0:-0.2 177*R->Xsp:R->HNsub+R->MaIcv) )
mkPWR(cell "VDD" "metal1" "inputOutput" list(80*R->Xsp R->HPsub) list(0:R->HPsub-R->MaIcv 177*R->Xsp:11*R->Ysp+0.2) )

mkm2RodPin(cell "x2" "metal2" "input" list( "none" ) list(6*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x4" "metal2" "input" list( "none" ) list(10*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x5" "metal2" "input" list( "none" ) list(11*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x6" "metal2" "input" list( "none" ) list(14*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x7" "metal2" "input" list( "none" ) list(15*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x8" "metal2" "input" list( "none" ) list(18*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x9" "metal2" "input" list( "none" ) list(19*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x10" "metal2" "input" list( "none" ) list(22*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x11" "metal2" "input" list( "none" ) list(23*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x12" "metal2" "input" list( "none" ) list(26*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x13" "metal2" "input" list( "none" ) list(27*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x14" "metal2" "input" list( "none" ) list(34*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x16" "metal2" "input" list( "none" ) list(38*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x17" "metal2" "input" list( "none") list(39*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x19" "metal2" "input" list( "none") list(43*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x21" "metal2" "input" list( "none") list(47*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x22" "metal2" "input" list( "none" ) list(50*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x23" "metal2" "input" list( "none") list(51*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x27" "metal2" "input" list( "none") list(59*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x28" "metal2" "input" list( "none" ) list(62*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x29" "metal2" "input" list( "none") list(63*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x30" "metal2" "input" list( "none" ) list(66*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x31" "metal2" "input" list( "none") list(67*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x32" "metal2" "input" list( "none" ) list(70*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x34" "metal2" "input" list( "none" ) list(74*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x35" "metal2" "input" list( "none") list(75*R->Xsp+R->Xsp2 5*R->Ysp))
```

```
mkm2RodPin(cell "x37" "metal2" "input" list( "none")
list(79*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x38" "metal2" "input" list( "none" )
list(82*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x39" "metal2" "input" list( "none")
list(83*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x40" "metal2" "input" list( "none" )
list(86*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x41" "metal2" "input" list( "none")
list(87*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x43" "metal2" "input" list( "none")
list(95*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x44" "metal2" "input" list( "none")
list(98*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x47" "metal2" "input" list( "none")
list(103*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x48" "metal2" "input" list( "none")
list(106*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x49" "metal2" "input" list( "none")
list(107*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x50" "metal2" "input" list( "none")
list(110*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x52" "metal2" "input" list( "none")
list(114*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x53" "metal2" "input" list( "none")
list(115*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x54" "metal2" "input" list( "none")
list(118*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x55" "metal2" "input" list( "none")
list(119*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x56" "metal2" "input" list( "none")
list(122*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x57" "metal2" "input" list( "none")
list(123*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x58" "metal2" "input" list( "none")
list(126*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x59" "metal2" "input" list( "none")
list(127*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x60" "metal2" "input" list( "none")
list(130*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x63" "metal2" "input" list( "none")
list(135*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x66" "metal2" "input" list( "none")
list(142*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x67" "metal2" "input" list( "none")
list(143*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x68" "metal2" "input" list( "none")
list(146*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x72" "metal2" "input" list( "none")
list(158*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x73" "metal2" "input" list( "none")
list(159*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x76" "metal2" "input" list( "none")
list(166*R->Xsp+R->Xsp2 5*R->Ysp))


mkm2RodPin(cell "y0" "metal2" "input" list( "none" )
list(4*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y52" "metal2" "input" list( "none" )
list(4*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y1" "metal2" "input" list( "none" )
list(5*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y53" "metal2" "input" list( "none" )
list(5*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y3" "metal2" "input" list( "none" )
list(9*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y54" "metal2" "input" list( "none" )
list(9*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y2" "metal2" "input" list( "none" )
list(12*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y4" "metal2" "input" list( "none" )
list(13*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y5" "metal2" "input" list( "none" )
list(16*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y55" "metal2" "input" list( "none" )
list(16*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y6" "metal2" "input" list( "none" )
list(17*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y56" "metal2" "input" list( "none" )
list(17*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y7" "metal2" "input" list( "none" )
list(20*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y57" "metal2" "input" list( "none" )
list(20*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y8" "metal2" "input" list( "none" )
list(21*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y58" "metal2" "input" list( "none" )
list(21*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y9" "metal2" "input" list( "none" )
list(24*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y10" "metal2" "input" list( "none" )
list(25*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y59" "metal2" "input" list( "none" )
list(25*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y60" "metal2" "input" list( "none" )
list(28*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y11" "metal2" "input" list( "none" )
list(33*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y61" "metal2" "input" list( "none" )
list(33*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y12" "metal2" "input" list( "none" )
list(37*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y62" "metal2" "input" list( "none" )
list(37*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y63" "metal2" "input" list( "none" )
list(40*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y13" "metal2" "input" list( "none" )
list(44*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y64" "metal2" "input" list( "none" )
list(44*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y14" "metal2" "input" list( "none" )
list(45*R->Xsp+R->Xsp2 7*R->Ysp))
```

```
mkm2RodPin(cell "y15" "metal2" "input" list( "none" )
list(48*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y65" "metal2" "input" list( "none" )
list(48*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y16" "metal2" "input" list( "none" )
list(49*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y66" "metal2" "input" list( "none" )
list(49*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y17" "metal2" "input" list( "none" )
list(52*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y18" "metal2" "input" list( "none" )
list(60*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y19" "metal2" "input" list( "none" )
list(61*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y67" "metal2" "input" list( "none" )
list(61*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y20" "metal2" "input" list( "none" )
list(64*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y68" "metal2" "input" list( "none" )
list(64*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y21" "metal2" "input" list( "none" )
list(65*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y69" "metal2" "input" list( "none" )
list(65*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y22" "metal2" "input" list( "none" )
list(68*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y70" "metal2" "input" list( "none" )
list(68*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y71" "metal2" "input" list( "none" )
list(69*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y72" "metal2" "input" list( "none" )
list(73*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y73" "metal2" "input" list( "none" )
list(76*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y23" "metal2" "input" list( "none" )
list(80*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y74" "metal2" "input" list( "none" )
list(80*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y75" "metal2" "input" list( "none" )
list(81*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y24" "metal2" "input" list( "none" )
list(84*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y76" "metal2" "input" list( "none" )
list(84*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y77" "metal2" "input" list( "none" )
list(85*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y25" "metal2" "input" list( "none" )
list(88*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y78" "metal2" "input" list( "none" )
list(93*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y26" "metal2" "input" list( "none" )
list(96*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y79" "metal2" "input" list( "none" )
list(96*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y80" "metal2" "input" list( "none" )
list(97*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y27" "metal2" "input" list( "none" )
list(101*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y82" "metal2" "input" list( "none" )
list(101*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y28" "metal2" "input" list( "none" )
list(104*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y81" "metal2" "input" list( "none" )
list(104*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y29" "metal2" "input" list( "none" )
list(105*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y83" "metal2" "input" list( "none" )
list(105*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y84" "metal2" "input" list( "none" )
list(108*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y30" "metal2" "input" list( "none" )
list(109*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y87" "metal2" "input" list( "none" )
list(112*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y31" "metal2" "input" list( "none" )
list(113*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y88" "metal2" "input" list( "none" )
list(113*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y85" "metal2" "input" list( "none" )
list(116*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y32" "metal2" "input" list( "none" )
list(117*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y86" "metal2" "input" list( "none" )
list(117*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y89" "metal2" "input" list( "none" )
list(120*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y90" "metal2" "input" list( "none" )
list(121*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y91" "metal2" "input" list( "none" )
list(124*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y92" "metal2" "input" list( "none" )
list(125*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y33" "metal2" "input" list( "none" )
list(128*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y93" "metal2" "input" list( "none" )
list(128*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y34" "metal2" "input" list( "none" )
list(129*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y94" "metal2" "input" list( "none" )
list(129*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y95" "metal2" "input" list( "none" )
list(133*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y35" "metal2" "input" list( "none" )
list(136*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y96" "metal2" "input" list( "none" )
list(136*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y97" "metal2" "input" list( "none" )
list(137*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y37" "metal2" "input" list( "none" )
list(140*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y36" "metal2" "input" list( "none" )
list(141*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y38" "metal2" "input" list( "none" )
list(144*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y39" "metal2" "input" list( "none" )
list(145*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y98" "metal2" "input" list( "none" )
list(145*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y40" "metal2" "input" list( "none" )
list(156*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y41" "metal2" "input" list( "none" )
list(157*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y42" "metal2" "input" list( "none" )
list(160*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y99" "metal2" "input" list( "none" )
list(160*R->Xsp+R->Xsp2 3*R->Ysp))
mkm2RodPin(cell "y43" "metal2" "input" list( "none" )
list(165*R->Xsp+R->Xsp2 7*R->Ysp))

dbCreateRect(cell list("poly1" "drawing")
list(3*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 6*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))

dbCreateRect(cell list("metal1" "drawing")
;list(14*R->Xsp-0.01-R->Mbxl2+R->Xsp2-0.033:5*R-
>Ysp 14*R->Xsp-0.01+R->Mbxl2+R->Xsp2:6*R-
>Ysp+R->Mwd2 ))
list(14*R->Xsp-R->Mwd2+R->Xsp2:5*R->Ysp-0.17
14*R->Xsp+R->Mwd2+R->Xsp2:6*R->Ysp+R->Mwd2 ))
```

```
dbCreateRect(cell list("metal1" "drawing")
list(10*R->Xsp+0.02-0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 10*R->Xsp+0.02+0.16+R->Xsp2:5*R->Ysp+R-
>Abx+R->Psp+R->Mbx2))

dbCreateRect(cell list("metal1" "drawing")
list(10*R->Xsp-0.02-0.16+R->Xsp2:6*R->Ysp-R->Mwd2
14*R->Xsp-0.01+R->Xsp2:6*R->Ysp+R->Mwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(46*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 47*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(66*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 67*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(102*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 103*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))

dbCreateRect(cell list("metal1" "drawing")
list(143*R->Xsp+0.01-R->Mbxl2+R->Xsp2:5*R->Ysp
143*R->Xsp+0.01+R->Mbxl2+R->Xsp2+0.033:6*R-
>Ysp+R->Mwd2 ))

dbCreateRect(cell list("metal1" "drawing")
list(139*R->Xsp-0.02-0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 139*R->Xsp-0.02+0.16+R->Xsp2:5*R->Ysp+R-
>Abx+R->Psp+R->Mbx2 ))

dbCreateRect(cell list("metal1" "drawing")
list(139*R->Xsp-0.02-0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 143*R->Xsp-0.01+R->Xsp2:6*R->Ysp+R-
>Mwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(155*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 158*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(158*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 159*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(3*R->Xsp+R->Xsp2:4*R->Ysp-R->Pwd2 6*R-
>Xsp+R->Xsp2:4*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(63*R->Xsp+R->Xsp2:4*R->Ysp-R->Pwd2 66*R-
>Xsp+R->Xsp2:4*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(67*R->Xsp+R->Xsp2:4*R->Ysp-R->Pwd2 70*R-
>Xsp+R->Xsp2:4*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(79*R->Xsp+R->Xsp2:4*R->Ysp-R->Pwd2 82*R-
>Xsp+R->Xsp2:4*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(94*R->Xsp+R->Xsp2:4*R->Ysp-R->Pwd2 95*R-
>Xsp+R->Xsp2:4*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("metal1" "drawing")
list(106*R->Xsp-0.01-R->Mbxl2+R->Xsp2-0.033:4*R-
>Ysp-R->Mwd2 106*R->Xsp-0.01+R->Mbxl2+R-
>Xsp2:5*R->Ysp ))

dbCreateRect(cell list("metal1" "drawing")
list(102*R->Xsp+0.02-0.16+R->Xsp2:5*R->Ysp-R-
>Abx-R->Psp-R->Mbx2  102*R->Xsp+0.02+0.16+R-
>Xsp2:4*R->Ysp+R->Mwd2))

dbCreateRect(cell list("metal1" "drawing")
list(102*R->Xsp-0.02-0.16+R->Xsp2:4*R->Ysp-R-
>Mwd2 106*R->Xsp-0.01+R->Xsp2:4*R->Ysp+R-
>Mwd2 ))

dbCreateRect(cell list("metal1" "drawing")
list(119*R->Xsp+0.01-R->Mbxl2+R->Xsp2:4*R->Ysp-R-
>Mwd2 119*R->Xsp+0.01+R->Mbxl2+R-
>Xsp2+0.033:5*R->Ysp ))

dbCreateRect(cell list("metal1" "drawing")
list(111*R->Xsp-0.02-0.16+R->Xsp2:5*R->Ysp-R->Abx-
R->Psp-R->Mbx2 111*R->Xsp-0.02+0.16+R-
>Xsp2:4*R->Ysp+R->Mwd2 ))

dbCreateRect(cell list("metal1" "drawing")
list(111*R->Xsp-0.02-0.16+R->Xsp2:4*R->Ysp-R-
>Mwd2 119*R->Xsp-0.01+R->Xsp2:4*R->Ysp+R-
>Mwd2 ))

dbCreateRect(cell list("metal1" "drawing")
list(119*R->Xsp+0.01-R->Mbxl2+R->Xsp2:4*R->Ysp-R-
>Mwd2 119*R->Xsp+0.01+R->Mbxl2+R-
>Xsp2+0.033:5*R->Ysp ))

dbCreateRect(cell list("metal1" "drawing")
list(114*R->Xsp+0.02-0.16+R->Xsp2:5*R->Ysp-R-
>Abx-R->Psp-R->Mbx2  114*R->Xsp+0.02+0.16+R-
>Xsp2:4*R->Ysp+R->Mwd2))

dbCreateRect(cell list("metal1" "drawing")
list(114*R->Xsp-0.02-0.16+R->Xsp2:4*R->Ysp-R-
>Mwd2 119*R->Xsp-0.01+R->Xsp2:4*R->Ysp+R-
>Mwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(126*R->Xsp+R->Xsp2:4*R->Ysp-R->Pwd2 127*R-
>Xsp+R->Xsp2:4*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(134*R->Xsp+R->Xsp2:4*R->Ysp-R->Pwd2 135*R-
>Xsp+R->Xsp2:4*R->Ysp+R->Pwd2 ))



dbCreateRect(cell list("poly1" "drawing")
list(135*R->Xsp+R->Xsp2:4*R->Ysp-R->Pwd2 138*R-
>Xsp+R->Xsp2:4*R->Ysp+R->Pwd2 ))

dbCreateRect(cell list("prBoundary" "drawing")
list(0:0  177*R->Xsp:11*R->Ysp ))

 dbSave(cell)
```

```
 dbClose(cell)
 )
```

Skill code for and_or_domino0

```
library="sfulib3"

procedure(and_or0( )
let( ( cvw cnm x y x1 y1 I0 I1 rod1 rod2 pt1 pt2 )

 sprintf(cnm "and_or0" )

 cell=dbOpenCellViewByType(library cnm "layout"
"maskLayout" "w")
 printf("Cellname: %s" cnm)

x = 0
y = 0

cvw = dbOpenCellViewByType( library "feTop" "layout"
"maskLayout" "r" )

I0 =  dbCreateInst(cell cvw "I0" list( R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Top" "layout"
"maskLayout" "r" )

I2 =  dbCreateInst(cell cvw "I2" list(4*R->Xsp+R->Xsp
0 ) "R0" )
I5 =  dbCreateInst(cell cvw "I5" list(12*R->Xsp+R->Xsp
0 ) "MY" )
I6 =  dbCreateInst(cell cvw "I6" list(12*R->Xsp+R->Xsp
0 ) "R0" )
I7 =  dbCreateInst(cell cvw "I7" list(16*R->Xsp+R->Xsp
0 ) "MY" )
I8 =  dbCreateInst(cell cvw "I8" list(16*R->Xsp+R->Xsp
0 ) "R0" )
I9 =  dbCreateInst(cell cvw "I9" list(20*R->Xsp+R->Xsp
0 ) "MY" )
I10 =  dbCreateInst(cell cvw "I10" list(20*R->Xsp+R-
>Xsp 0 ) "R0" )
I11 =  dbCreateInst(cell cvw "I11" list(24*R->Xsp+R-
>Xsp 0 ) "MY" )
I12 =  dbCreateInst(cell cvw "I12" list(24*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eTop" "layout"
"maskLayout" "r" )

I3 =  dbCreateInst(cell cvw "I3" list(6*R->Xsp+R->Xsp
0 ) "R0" )
I13 =  dbCreateInst(cell cvw "I13" list(26*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop2" "layout"
"maskLayout" "r" )

I4 =  dbCreateInst(cell cvw "I4" list(8*R->Xsp+R->Xsp
0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop" "layout"
"maskLayout" "r" )
)

I1 =  dbCreateInst(cell cvw "I1" list(4*R->Xsp+R->Xsp
0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "pSub" "layout"
"maskLayout" "r" )

I198 =  dbCreateInst(cell cvw "I198" list( 28*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Top" "layout"
"maskLayout" "r" )

I14 =  dbCreateInst(cell cvw "I14" list(32*R->Xsp+R-
>Xsp 0 ) "R0" )
I16 =  dbCreateInst(cell cvw "I16" list(36*R->Xsp+R-
>Xsp 0 ) "R0" )
I19 =  dbCreateInst(cell cvw "I19" list(44*R->Xsp+R-
>Xsp 0 ) "MY" )
I21 =  dbCreateInst(cell cvw "I21" list(48*R->Xsp+R-
>Xsp 0 ) "MY" )
I22 =  dbCreateInst(cell cvw "I22" list(48*R->Xsp+R-
>Xsp 0 ) "R0" )
I23 =  dbCreateInst(cell cvw "I23" list(52*R->Xsp+R-
>Xsp 0 ) "MY" )
I27 =  dbCreateInst(cell cvw "I27" list(60*R->Xsp+R-
>Xsp 0 ) "MY" )
I28 =  dbCreateInst(cell cvw "I28" list(60*R->Xsp+R-
>Xsp 0 ) "R0" )
I29 =  dbCreateInst(cell cvw "I29" list(64*R->Xsp+R-
>Xsp 0 ) "MY" )
I31 =  dbCreateInst(cell cvw "I31" list(68*R->Xsp+R-
>Xsp 0 ) "MY" )
I37 =  dbCreateInst(cell cvw "I37" list(80*R->Xsp+R-
>Xsp 0 ) "MY" )
I39 =  dbCreateInst(cell cvw "I39" list(84*R->Xsp+R-
>Xsp 0 ) "MY" )
I41 =  dbCreateInst(cell cvw "I41" list(88*R->Xsp+R-
>Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eTop" "layout"
"maskLayout" "r" )

I15 =  dbCreateInst(cell cvw "I15" list(34*R->Xsp+R-
>Xsp 0 ) "R0" )
I17 =  dbCreateInst(cell cvw "I17" list(38*R->Xsp+R-
>Xsp 0 ) "R0" )
I18 =  dbCreateInst(cell cvw "I18" list(40*R->Xsp+R-
>Xsp 0 ) "R0" )
I24 =  dbCreateInst(cell cvw "I24" list(52*R->Xsp+R-
>Xsp 0 ) "R0" )
I25 =  dbCreateInst(cell cvw "I25" list(54*R->Xsp+R-
>Xsp 0 ) "R0" )
I26 =  dbCreateInst(cell cvw "I26" list(56*R->Xsp+R-
>Xsp 0 ) "R0" )
I32 =  dbCreateInst(cell cvw "I32" list(68*R->Xsp+R-
>Xsp 0 ) "R0" )
I33 =  dbCreateInst(cell cvw "I33" list(70*R->Xsp+R-
>Xsp 0 ) "R0" )
```

```
I34 =  dbCreateInst(cell cvw "I34" list(72*R->Xsp+R->Xsp 0 ) "R0" )
I35 =  dbCreateInst(cell cvw "I35" list(74*R->Xsp+R->Xsp 0 ) "R0" )
I36 =  dbCreateInst(cell cvw "I36" list(76*R->Xsp+R->Xsp 0 ) "R0" )
I38 =  dbCreateInst(cell cvw "I38" list(80*R->Xsp+R->Xsp 0 ) "R0" )
I40 =  dbCreateInst(cell cvw "I40" list(84*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop" "layout" "maskLayout" "r" )

I20 =  dbCreateInst(cell cvw "I20" list(44*R->Xsp+R->Xsp 0 ) "R0" )
I30 =  dbCreateInst(cell cvw "I30" list(64*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "pSub" "layout" "maskLayout" "r" )

I199 =  dbCreateInst(cell cvw "I199" list( 88*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Top" "layout" "maskLayout" "r" )

I43 =  dbCreateInst(cell cvw "I43" list(96*R->Xsp+R->Xsp 0 ) "MY" )
I47 =  dbCreateInst(cell cvw "I47" list(104*R->Xsp+R->Xsp 0 ) "MY" )
I48 =  dbCreateInst(cell cvw "I48" list(104*R->Xsp+R->Xsp 0 ) "R0" )
I50 =  dbCreateInst(cell cvw "I50" list(108*R->Xsp+R->Xsp 0 ) "R0" )
I52 =  dbCreateInst(cell cvw "I52" list(112*R->Xsp+R->Xsp 0 ) "R0" )
I54 =  dbCreateInst(cell cvw "I54" list(116*R->Xsp+R->Xsp 0 ) "R0" )
I59 =  dbCreateInst(cell cvw "I59" list(128*R->Xsp+R->Xsp 0 ) "MY" )
I60 =  dbCreateInst(cell cvw "I60" list(128*R->Xsp+R->Xsp 0 ) "R0" )
I63 =  dbCreateInst(cell cvw "I63" list(136*R->Xsp+R->Xsp 0 ) "MY" )
I66 =  dbCreateInst(cell cvw "I66" list(140*R->Xsp+R->Xsp 0 ) "R0" )
I67 =  dbCreateInst(cell cvw "I67" list(144*R->Xsp+R->Xsp 0 ) "MY" )
I68 =  dbCreateInst(cell cvw "I68" list(144*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eTop" "layout" "maskLayout" "r" )

I42 =  dbCreateInst(cell cvw "I42" list(92*R->Xsp+R->Xsp 0 ) "R0" )
I44 =  dbCreateInst(cell cvw "I44" list(96*R->Xsp+R->Xsp 0 ) "R0" )
I45 =  dbCreateInst(cell cvw "I45" list(98*R->Xsp+R->Xsp 0 ) "R0" )
I49 =  dbCreateInst(cell cvw "I49" list(106*R->Xsp+R->Xsp 0 ) "R0" )
I51 =  dbCreateInst(cell cvw "I51" list(110*R->Xsp+R->Xsp 0 ) "R0" )
I53 =  dbCreateInst(cell cvw "I53" list(114*R->Xsp+R->Xsp 0 ) "R0" )
I55 =  dbCreateInst(cell cvw "I55" list(118*R->Xsp+R->Xsp 0 ) "R0" )
I56 =  dbCreateInst(cell cvw "I56" list(120*R->Xsp+R->Xsp 0 ) "R0" )
I57 =  dbCreateInst(cell cvw "I57" list(122*R->Xsp+R->Xsp 0 ) "R0" )
I58 =  dbCreateInst(cell cvw "I58" list(124*R->Xsp+R->Xsp 0 ) "R0" )
I61 =  dbCreateInst(cell cvw "I61" list(130*R->Xsp+R->Xsp 0 ) "R0" )
I62 =  dbCreateInst(cell cvw "I62" list(132*R->Xsp+R->Xsp 0 ) "R0" )
I64 =  dbCreateInst(cell cvw "I64" list(136*R->Xsp+R->Xsp 0 ) "R0" )
I69 =  dbCreateInst(cell cvw "I69" list(146*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop" "layout" "maskLayout" "r" )

I46 =  dbCreateInst(cell cvw "I46" list(100*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "sTop2" "layout" "maskLayout" "r" )

I65 =  dbCreateInst(cell cvw "I65" list(140*R->Xsp+R->Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "pSub" "layout" "maskLayout" "r" )

I200 =  dbCreateInst(cell cvw "I200" list( 148*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Top" "layout" "maskLayout" "r" )

I72 =  dbCreateInst(cell cvw "I72" list(156*R->Xsp+R->Xsp 0 ) "R0" )
I76 =  dbCreateInst(cell cvw "I76" list(164*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eTop" "layout" "maskLayout" "r" )

I70 =  dbCreateInst(cell cvw "I70" list(152*R->Xsp+R->Xsp 0 ) "R0" )
I74 =  dbCreateInst(cell cvw "I74" list(160*R->Xsp+R->Xsp 0 ) "R0" )
I75 =  dbCreateInst(cell cvw "I75" list(162*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )
```

```
cvw = dbOpenCellViewByType( library "sTop" "layout"
"maskLayout" "r" )

I71 =  dbCreateInst(cell cvw "I71" list(156*R->Xsp+R-
>Xsp 0 ) "MY" )
I73 =  dbCreateInst(cell cvw "I73" list(160*R->Xsp+R-
>Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "feBot" "layout"
"maskLayout" "r" )

I100 =  dbCreateInst(cell cvw "I100" list( R->Xsp 0 )
"R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout"
"maskLayout" "r" )

I102 =  dbCreateInst(cell cvw "I102" list(4*R->Xsp+R-
>Xsp 0 ) "R0" )
I104 =  dbCreateInst(cell cvw "I104" list(8*R->Xsp+R-
>Xsp 0 ) "R0" )
I107 =  dbCreateInst(cell cvw "I107" list(16*R->Xsp+R-
>Xsp 0 ) "MY" )
I108 =  dbCreateInst(cell cvw "I108" list(16*R->Xsp+R-
>Xsp 0 ) "R0" )
I109 =  dbCreateInst(cell cvw "I109" list(20*R->Xsp+R-
>Xsp 0 ) "MY" )
I110 =  dbCreateInst(cell cvw "I110" list(20*R->Xsp+R-
>Xsp 0 ) "R0" )
I112 =  dbCreateInst(cell cvw "I112" list(24*R->Xsp+R-
>Xsp 0 ) "R0" )
I113 =  dbCreateInst(cell cvw "I113" list(28*R->Xsp+R-
>Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout"
"maskLayout" "r" )

I103 =  dbCreateInst(cell cvw "I103" list(6*R->Xsp+R-
>Xsp 0 ) "R0" )
I105 =  dbCreateInst(cell cvw "I105" list(10*R->Xsp+R-
>Xsp 0 ) "R0" )
I106 =  dbCreateInst(cell cvw "I106" list(12*R->Xsp+R-
>Xsp 0 ) "R0" )
I111 =  dbCreateInst(cell cvw "I111" list(22*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout"
"maskLayout" "r" )

I101 =  dbCreateInst(cell cvw "I101" list(4*R->Xsp+R-
>Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout"
"maskLayout" "r" )

I114 =  dbCreateInst(cell cvw "I114" list(32*R->Xsp+R-
>Xsp 0 ) "R0" )
I116 =  dbCreateInst(cell cvw "I116" list(36*R->Xsp+R-
>Xsp 0 ) "R0" )
I117 =  dbCreateInst(cell cvw "I117" list(40*R->Xsp+R-
>Xsp 0 ) "MY" )
I119 =  dbCreateInst(cell cvw "I119" list(44*R->Xsp+R-
>Xsp 0 ) "MY" )
I121 =  dbCreateInst(cell cvw "I121" list(48*R->Xsp+R-
>Xsp 0 ) "MY" )
I122 =  dbCreateInst(cell cvw "I122" list(48*R->Xsp+R-
>Xsp 0 ) "R0" )
I128 =  dbCreateInst(cell cvw "I128" list(60*R->Xsp+R-
>Xsp 0 ) "R0" )
I130 =  dbCreateInst(cell cvw "I130" list(64*R->Xsp+R-
>Xsp 0 ) "R0" )
I132 =  dbCreateInst(cell cvw "I132" list(68*R->Xsp+R-
>Xsp 0 ) "R0" )
I134 =  dbCreateInst(cell cvw "I134" list(72*R->Xsp+R-
>Xsp 0 ) "R0" )
I135 =  dbCreateInst(cell cvw "I135" list(76*R->Xsp+R-
>Xsp 0 ) "MY" )
I138 =  dbCreateInst(cell cvw "I138" list(80*R->Xsp+R-
>Xsp 0 ) "R0" )
I139 =  dbCreateInst(cell cvw "I139" list(84*R->Xsp+R-
>Xsp 0 ) "MY" )
I140 =  dbCreateInst(cell cvw "I140" list(84*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout"
"maskLayout" "r" )

I115 =  dbCreateInst(cell cvw "I115" list(34*R->Xsp+R-
>Xsp 0 ) "R0" )
I118 =  dbCreateInst(cell cvw "I118" list(40*R->Xsp+R-
>Xsp 0 ) "R0" )
I120 =  dbCreateInst(cell cvw "I120" list(44*R->Xsp+R-
>Xsp 0 ) "R0" )
I123 =  dbCreateInst(cell cvw "I123" list(50*R->Xsp+R-
>Xsp 0 ) "R0" )
I124 =  dbCreateInst(cell cvw "I124" list(52*R->Xsp+R-
>Xsp 0 ) "R0" )
I125 =  dbCreateInst(cell cvw "I125" list(54*R->Xsp+R-
>Xsp 0 ) "R0" )
I126 =  dbCreateInst(cell cvw "I126" list(56*R->Xsp+R-
>Xsp 0 ) "R0" )
I127 =  dbCreateInst(cell cvw "I127" list(58*R->Xsp+R-
>Xsp 0 ) "R0" )
I133 =  dbCreateInst(cell cvw "I133" list(70*R->Xsp+R-
>Xsp 0 ) "R0" )
I136 =  dbCreateInst(cell cvw "I136" list(76*R->Xsp+R-
>Xsp 0 ) "R0" )
I141 =  dbCreateInst(cell cvw "I141" list(86*R->Xsp+R-
>Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout"
"maskLayout" "r" )

I129 =  dbCreateInst(cell cvw "I129" list(64*R->Xsp+R-
>Xsp 0 ) "MY" )
I131 =  dbCreateInst(cell cvw "I131" list(68*R->Xsp+R-
>Xsp 0 ) "MY" )
I137 =  dbCreateInst(cell cvw "I137" list(80*R->Xsp+R-
>Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout"
"maskLayout" "r" )
```

```
I143 =  dbCreateInst(cell cvw "I143" list(96*R->Xsp+R->Xsp 0 ) "MY" )
I144 =  dbCreateInst(cell cvw "I144" list(96*R->Xsp+R->Xsp 0 ) "R0" )
I147 =  dbCreateInst(cell cvw "I147" list(104*R->Xsp+R->Xsp 0 ) "MY" )
I148 =  dbCreateInst(cell cvw "I148" list(104*R->Xsp+R->Xsp 0 ) "R0" )
I149 =  dbCreateInst(cell cvw "I149" list(108*R->Xsp+R->Xsp 0 ) "MY" )
I153 =  dbCreateInst(cell cvw "I153" list(116*R->Xsp+R->Xsp 0 ) "MY" )
I154 =  dbCreateInst(cell cvw "I154" list(116*R->Xsp+R->Xsp 0 ) "R0" )
I155 =  dbCreateInst(cell cvw "I155" list(120*R->Xsp+R->Xsp 0 ) "MY" )
I156 =  dbCreateInst(cell cvw "I156" list(120*R->Xsp+R->Xsp 0 ) "R0" )
I157 =  dbCreateInst(cell cvw "I157" list(124*R->Xsp+R->Xsp 0 ) "MY" )
I158 =  dbCreateInst(cell cvw "I158" list(124*R->Xsp+R->Xsp 0 ) "R0" )
I160 =  dbCreateInst(cell cvw "I160" list(128*R->Xsp+R->Xsp 0 ) "R0" )
I163 =  dbCreateInst(cell cvw "I163" list(136*R->Xsp+R->Xsp 0 ) "MY" )
I168 =  dbCreateInst(cell cvw "I168" list(144*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I145 =  dbCreateInst(cell cvw "I145" list(98*R->Xsp+R->Xsp 0 ) "R0" )
I150 =  dbCreateInst(cell cvw "I150" list(108*R->Xsp+R->Xsp 0 ) "R0" )
I161 =  dbCreateInst(cell cvw "I161" list(130*R->Xsp+R->Xsp 0 ) "R0" )
I165 =  dbCreateInst(cell cvw "I165" list(138*R->Xsp+R->Xsp 0 ) "R0" )
I166 =  dbCreateInst(cell cvw "I166" list(140*R->Xsp+R->Xsp 0 ) "R0" )
I167 =  dbCreateInst(cell cvw "I167" list(142*R->Xsp+R->Xsp 0 ) "R0" )
I169 =  dbCreateInst(cell cvw "I169" list(146*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I142 =  dbCreateInst(cell cvw "I142" list(92*R->Xsp+R->Xsp 0 ) "R0" )
I159 =  dbCreateInst(cell cvw "I159" list(128*R->Xsp+R->Xsp 0 ) "MY" )
I162 =  dbCreateInst(cell cvw "I162" list(132*R->Xsp+R->Xsp 0 ) "R0" )
I164 =  dbCreateInst(cell cvw "I164" list(136*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I146 =  dbCreateInst(cell cvw "I146" list(100*R->Xsp+R->Xsp 0 ) "R0" )
I151 =  dbCreateInst(cell cvw "I151" list(112*R->Xsp+R->Xsp 0 ) "MY" )
I152 =  dbCreateInst(cell cvw "I152" list(112*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I173 =  dbCreateInst(cell cvw "I173" list(160*R->Xsp+R->Xsp 0 ) "MY" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )

I170 =  dbCreateInst(cell cvw "I170" list(152*R->Xsp+R->Xsp 0 ) "R0" )
I171 =  dbCreateInst(cell cvw "I171" list(154*R->Xsp+R->Xsp 0 ) "R0" )
I172 =  dbCreateInst(cell cvw "I172" list(156*R->Xsp+R->Xsp 0 ) "R0" )
I174 =  dbCreateInst(cell cvw "I174" list(160*R->Xsp+R->Xsp 0 ) "R0" )
I175 =  dbCreateInst(cell cvw "I175" list(162*R->Xsp+R->Xsp 0 ) "R0" )
I176 =  dbCreateInst(cell cvw "I176" list(164*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

cvw = dbOpenCellViewByType( library "eBot" "layout" "maskLayout" "r" )


 dbClose( cvw )

cvw = dbOpenCellViewByType( library "Domin" "layout" "maskLayout" "r" )

I201 =  dbCreateInst(cell cvw "I201" list( 166*R->Xsp+R->Xsp 0 ) "R0" )

 dbClose( cvw )

mkm2RodPin(cell "equ" "metal2" "input" list( "none" ) list(169*R->Xsp+R->Xsp2 5*R->Ysp))
mkm2RodPin(cell "out_t" "metal2" "output" list( "none" ) list(172*R->Xsp+R->Xsp2 5*R->Ysp))

mkPWR(cell "VSS" "metal1" "inputOutput" list(80*R->Xsp R->HNsub) list(0:-0.2 174*R->Xsp:R->HNsub+R->MaIcv) )
mkPWR(cell "VDD" "metal1" "inputOutput" list(80*R->Xsp R->HPsub) list(0:R->HPsub-R->MaIcv 174*R->Xsp:11*R->Ysp+0.2) )

mkm2RodPin(cell "x2" "metal2" "input" list( "none" ) list(6*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x5" "metal2" "input" list( "none" ) list(11*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x6" "metal2" "input" list( "none" ) list(14*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x7" "metal2" "input" list( "none" ) list(15*R->Xsp+R->Xsp2 5*R->Ysp))
```

```
mkm2RodPin(cell "x8" "metal2" "input" list( "none" )
list(18*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x9" "metal2" "input" list( "none" )
list(19*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x10" "metal2" "input" list( "none" )
list(22*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x11" "metal2" "input" list( "none" )
list(23*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x12" "metal2" "input" list( "none" )
list(26*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x14" "metal2" "input" list( "none" )
list(34*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x16" "metal2" "input" list( "none" )
list(38*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x19" "metal2" "input" list( "none")
list(43*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x21" "metal2" "input" list( "none")
list(47*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x22" "metal2" "input" list( "none" )
list(50*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x23" "metal2" "input" list( "none")
list(51*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x27" "metal2" "input" list( "none")
list(59*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x28" "metal2" "input" list( "none" )
list(62*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x29" "metal2" "input" list( "none")
list(63*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x31" "metal2" "input" list( "none")
list(67*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x37" "metal2" "input" list( "none")
list(79*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x39" "metal2" "input" list( "none")
list(83*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x41" "metal2" "input" list( "none")
list(87*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x43" "metal2" "input" list( "none")
list(95*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x47" "metal2" "input" list( "none")
list(103*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x48" "metal2" "input" list( "none")
list(106*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x50" "metal2" "input" list( "none")
list(110*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x52" "metal2" "input" list( "none")
list(114*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x54" "metal2" "input" list( "none")
list(118*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x59" "metal2" "input" list( "none")
list(127*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x60" "metal2" "input" list( "none")
list(130*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x63" "metal2" "input" list( "none")
list(135*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x66" "metal2" "input" list( "none")
list(142*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x67" "metal2" "input" list( "none")
list(143*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x68" "metal2" "input" list( "none")
list(146*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x72" "metal2" "input" list( "none")
list(158*R->Xsp+R->Xsp2 5*R->Ysp))

mkm2RodPin(cell "x76" "metal2" "input" list( "none")
list(166*R->Xsp+R->Xsp2 5*R->Ysp))


mkm2RodPin(cell "y0" "metal2" "input" list( "none" )
list(4*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y1" "metal2" "input" list( "none" )
list(5*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y3" "metal2" "input" list( "none" )
list(9*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y2" "metal2" "input" list( "none" )
list(12*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y4" "metal2" "input" list( "none" )
list(13*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y5" "metal2" "input" list( "none" )
list(16*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y6" "metal2" "input" list( "none" )
list(17*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y7" "metal2" "input" list( "none" )
list(20*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y8" "metal2" "input" list( "none" )
list(21*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y9" "metal2" "input" list( "none" )
list(24*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y10" "metal2" "input" list( "none" )
list(25*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y11" "metal2" "input" list( "none" )
list(33*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y12" "metal2" "input" list( "none" )
list(37*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y13" "metal2" "input" list( "none" )
list(44*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y14" "metal2" "input" list( "none" )
list(45*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y15" "metal2" "input" list( "none" )
list(48*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y16" "metal2" "input" list( "none" )
list(49*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y17" "metal2" "input" list( "none" )
list(52*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y18" "metal2" "input" list( "none" )
list(60*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y19" "metal2" "input" list( "none" )
list(61*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y20" "metal2" "input" list( "none" )
list(64*R->Xsp+R->Xsp2 7*R->Ysp))
```

```
mkm2RodPin(cell "y21" "metal2" "input" list( "none" )
list(65*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y22" "metal2" "input" list( "none" )
list(68*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y23" "metal2" "input" list( "none" )
list(80*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y24" "metal2" "input" list( "none" )
list(84*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y25" "metal2" "input" list( "none" )
list(88*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y26" "metal2" "input" list( "none" )
list(96*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y27" "metal2" "input" list( "none" )
list(101*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y28" "metal2" "input" list( "none" )
list(104*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y29" "metal2" "input" list( "none" )
list(105*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y30" "metal2" "input" list( "none" )
list(109*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y31" "metal2" "input" list( "none" )
list(113*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y32" "metal2" "input" list( "none" )
list(117*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y33" "metal2" "input" list( "none" )
list(128*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y34" "metal2" "input" list( "none" )
list(129*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y35" "metal2" "input" list( "none" )
list(136*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y37" "metal2" "input" list( "none" )
list(140*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y36" "metal2" "input" list( "none" )
list(141*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y38" "metal2" "input" list( "none" )
list(144*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y39" "metal2" "input" list( "none" )
list(145*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y40" "metal2" "input" list( "none" )
list(156*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y41" "metal2" "input" list( "none" )
list(157*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y42" "metal2" "input" list( "none" )
list(160*R->Xsp+R->Xsp2 7*R->Ysp))
mkm2RodPin(cell "y43" "metal2" "input" list( "none" )
list(165*R->Xsp+R->Xsp2 7*R->Ysp))

dbCreateRect(cell list("poly1" "drawing")
list(3*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 6*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))

dbCreateRect(cell list("metal1" "drawing")
;list(14*R->Xsp-0.01-R->Mbxl2+R->Xsp2-0.033:5*R-
>Ysp 14*R->Xsp-0.01+R->Mbxl2+R->Xsp2:6*R-
>Ysp+R->Mwd2 ))
list(14*R->Xsp-R->Mwd2+R->Xsp2:5*R->Ysp-0.17
14*R->Xsp+R->Mwd2+R->Xsp2:6*R->Ysp+R->Mwd2 ))

dbCreateRect(cell list("metal1" "drawing")
list(10*R->Xsp+0.02-0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 10*R->Xsp+0.02+0.16+R->Xsp2:5*R->Ysp+R-
>Abx+R->Psp+R->Mbx2))

dbCreateRect(cell list("metal1" "drawing")
list(10*R->Xsp-0.02-0.16+R->Xsp2:6*R->Ysp-R->Mwd2
14*R->Xsp-0.01+R->Xsp2:6*R->Ysp+R->Mwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(46*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 47*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(66*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 67*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(102*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 103*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))

dbCreateRect(cell list("metal1" "drawing")
list(143*R->Xsp+0.01-R->Mbxl2+R->Xsp2:5*R->Ysp
143*R->Xsp+0.01+R->Mbxl2+R->Xsp2+0.033:6*R-
>Ysp+R->Mwd2 ))

dbCreateRect(cell list("metal1" "drawing")
list(139*R->Xsp-0.02-0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 139*R->Xsp-0.02+0.16+R->Xsp2:5*R->Ysp+R-
>Abx+R->Psp+R->Mbx2 ))

dbCreateRect(cell list("metal1" "drawing")
list(139*R->Xsp-0.02-0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 143*R->Xsp-0.01+R->Xsp2:6*R->Ysp+R-
>Mwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(155*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 158*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("poly1" "drawing")
list(158*R->Xsp+R->Xsp2:6*R->Ysp-R->Pwd2 159*R-
>Xsp+R->Xsp2:6*R->Ysp+R->Pwd2 ))


dbCreateRect(cell list("prBoundary" "drawing")
list(0:0  174*R->Xsp:11*R->Ysp ))

 dbSave(cell)
 dbClose(cell)
 )
)
```

C++ code for reordering process:

```
#include <fstream>
#include <string>
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

using namespace std;

int main()
{
string line;
int number_term = 0;

int exist [6561] = {0};

int bit = 2187;
int sum = 0;
int reminder = 0;
int coef = 0;
int true_term = 0;

ifstream fin;
ofstream fout;

for (int q = 0; q <16;
q++)
{
        if (q == 0)
```

```
                {
fin.open("tbits0.out");
fout.open("tbits0r.out");
                }
                else if (q ==
1)
                {
fin.open("tbits0z.out");
fout.open("tbits0zr.out");
                }
                else if (q ==
2)
                {
fin.open("tbits1.out");
fout.open("tbits1r.out");
                }
                else if (q ==
3)
                {
fin.open("tbits1z.out");
fout.open("tbits1zr.out");
                }
                else if (q ==
4)
                {
fin.open("tbits2.out");
fout.open("tbits2r.out");
                }
                else if (q ==
5)
                {
fin.open("tbits2z.out");
fout.open("tbits2zr.out");
                }
                else if (q ==
6)
                {
fin.open("tbits3.out");
fout.open("tbits3r.out");
                }
                else if (q ==
7)
                {
fin.open("tbits3z.out");
fout.open("tbits3zr.out");
                }
                else if (q ==
8)
                {
fin.open("tbits4.out");
fout.open("tbits4r.out");
                }
                else if (q ==
9)
                {
fin.open("tbits4z.out");
fout.open("tbits4zr.out");
                }
                else if (q ==
10)
                {
fin.open("tbits5.out");
fout.open("tbits5r.out");
                }
                else if (q ==
11)
                {
fin.open("tbits5z.out");
fout.open("tbits5zr.out");
                }
                else if (q ==
12)
                {
fin.open("tbits6.out");
fout.open("tbits6r.out");
                }
                else if (q ==
13)
                {
fin.open("tbits6z.out");
fout.open("tbits6zr.out");
                }
                else if (q ==
14)
                {
fin.open("tbits7.out");
fout.open("tbits7r.out");
                }
                else if (q ==
15)
                {
fin.open("tbits7z.out");
fout.open("tbits7zr.out");
                }
                getline(fin,
line);
                getline(fin,
line);
                getline(fin,
line);
                number_term
= atoi(&line[3]);
                fout << ".i
8"<<endl;
                fout << ".o
1"<<endl;
                fout << ".p
"<<number_term<<endl;
                for (int i = 0; i
< number_term; i++)
                {
        getline(fin, line);
                        for
(int j = 0; j<8; j++)
                        {
        if (line[j] == '1')
        {
                sum = sum +
bit;
        }

        else if (line[j] == '-')
        {
                sum = sum +
2*bit;
        }

        bit = bit/3;
                        }
                        bit
= 2187;

        exist[sum] = exist[sum]
+ 1;

        sum = 0;
                }
                fin.close();
                for (i = 0; i <
6560; i++)
                {
```

```
            reminder = i;

                                    for
(int j =0; j < exist[i];j++)
                                    {
            for (int k = 0; k < 8; k++)

            {
                        coef =
reminder/bit;
                        reminder =
reminder%bit;



                        if (coef == 2)

                        {


            fout << "-";

                        }
                                    else if (coef
== 1)

                                    {

                        fout <<"1";

                                    }

                                    else

                                    {

                        fout << "0";

                                    }


                                    bit = bit/3;

                        }
                        bit = 2187;

                        reminder = i;



                        fout << " 1" << endl;
                                                }

                                    }

                                    for (int l = 0; l
<6560; l++)
                                    {

                        exist[l] = 0;
                                    }


                                    fout << ".e"
<< endl;
                                    fout.close();
                        }

                        return 0;
}
```

C++ code for *encryption_raw*:

```
#include <fstream>
#include <string>
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

using namespace std;

int main()
{
            string line;
            int number_term = 0;

            ifstream fin;
            ofstream fout;

            fout.open("encryption_r
aw.vhd");

            fout << "LIBRARY
ieee;" <<endl;
            fout << "USE
ieee.std_logic_1164.ALL;" <<endl;
            fout << "USE
ieee.std_logic_unsigned.ALL;"
<<endl;
            fout << "USE
ieee.std_logic_arith.ALL;" <<endl;
            fout << " " << endl;

            fout << "LIBRARY
vs18sc;" <<endl;
            fout << "USE
vs18sc.ALL;" <<endl;
            fout << " " <<endl;



            fout << "entity
encryption_raw is" << endl;
            fout << "Port (" << endl;
            fout << "      data_in : in
std_logic_vector(7 downto 0);" <<
endl;
            fout << "      clk_1 : in
std_logic;" << endl;
            fout << "      data_out :
out std_logic_vector(7 downto 0)"
<< endl;
            fout << ");" << endl;
            fout << "end
encryption_raw;" << endl;

            fout << " "<< endl;


            fout << "architecture
structure of encryption_raw is" <<
endl;
            fout << "signal input:
std_logic_vector(7 downto
0);"<<endl;
            fout << "signal output:
std_logic_vector(7 downto
0);"<<endl;

            fout << "begin" << endl;
            fout << "
process(clk_1)" << endl;
            fout << "        begin" <<
endl;
            fout << "           if
(clk_1='1' and clk_1'event)
then"<<endl;
            fout << "               input
<= data_in;"<<endl;
            fout << "
data_out <= output;"<<endl;
            fout << "           end if;
"<<endl;
            fout << "      end
process;"<<endl;
   fout << " " <<endl;

            fout << "process(input)"
<< endl;
   fout << "           begin" <<
endl;

            for (int q = 0; q <8; q++)
            {
                        if(q == 0)
                        {

            fin.open("tbits0r.out");

                        }

                        else if (q ==
1)
                        {

            fin.open("tbits1r.out");

                        }

                        else if (q ==
2)
                        {

            fin.open("tbits2r.out");

                        }

                        else if (q ==
3)
                        {
```

```
		fin.open("tbits3r.out");
		}
		else if (q == 4)
		{
		fin.open("tbits4r.out");
		}
		else if (q == 5)
		{
		fin.open("tbits5r.out");
		}
		else if (q == 6)
		{
		fin.open("tbits6r.out");
		}
		else if (q == 7)
		{
		fin.open("tbits7r.out");
		}

		getline(fin, line);
		getline(fin, line);
		getline(fin, line);
		number_term = atoi(&line[3]);
		fout << "  if (";
		getline(fin, line);
		for (int j = 0; j<8; j++)
		{
			if (line[j] == '1')
			{
		fout << "input(" << 7-j << ") = '1' and ";
			}
		else if (line[j] == '0')
			{
		fout << "input(" << 7-j << ") = '0' and ";
			}
		}
		fout << " ) then" << endl;
		fout << " " << endl;
		fout << "
		output(" << q << ") <= '1';" << endl;
		fout << " " << endl;
		for (int i = 0; i < number_term-1; i++)
		{
		fout << "   elsif (";
		getline(fin, line);
		for (int j = 0; j<8; j++)
		{
		if (line[j] == '1')
		{
		fout << "input(" << 7-j << ") = '1' and ";
		}
		else if (line[j] == '0')
		{
		fout << "input(" << 7-j << ") = '0' and ";
		}
		}
		fout << " ) then" << endl;
		fout << " " << endl;
		fout << "   output(" << q << ") <= '1';" << endl;
		fout << " " << endl;
		}
		fout << " end if;" << endl;
		fin.close();
		}
		fout << "end process;" << endl;
		fout << "end structure;" << endl;
		fout.close();
		return 0;
}
```

C++ code for *encryption_ao_diff*:

```
#include <fstream>
#include <string>
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

using namespace std;

int main()
{
		string line;
		int number_term = 0;
		int number_term2 = 0;

		ofstream fout;
		fout.open("encryption.vhd");
		ifstream fin;
		ifstream fin2;

		int exist_x [81] = {0};
		int exist_y [81] = {0};
		int bit = 27;
		int sum_x = 0;
		int sum_y = 0;
		int base = 0;
		int true_term = 0;
		int occur [81] = {0};
		int exist_t [77] = {0};
		int exist_f [77] = {0};
		int sum = 0;
		int run = 0;
		int current [77] = {0};

		run = 8;
		fout << "LIBRARY ieee;" <<endl;
		fout << "USE ieee.std_logic_1164.ALL;"  <<endl;
		fout << "USE ieee.std_logic_unsigned.ALL;" <<endl;
		fout << "USE ieee.std_logic_arith.ALL;" <<endl;
		fout << " " << endl;

		fout << "LIBRARY vs18sc;" <<endl;
		fout << "USE vs18sc.ALL;" <<endl;
		fout << " " <<endl;

		fout << "entity
```

```
encryption is" << endl;
		fout << "Port (" << endl;
		fout << "	data_in : in
std_logic_vector(7 downto 0);" <<
endl;
		fout << "	clk_1 : in
std_logic;" << endl;
		fout << "	data_outz :
out std_logic_vector(7 downto
0);" << endl;
		fout << "	data_out :
out std_logic_vector(7 downto 0)"
<< endl;
		fout << ");" << endl;
		fout << "end
encryption;" << endl;

		fout << " "<< endl;


		fout << "architecture
structure of encryption is" << endl;

		for (int i = 0; i <run; i++)
		{
			if(i == 0)
			{

		fin.open("tbits0r.out");

		fin2.open("tbits0zr.out");
			}

			else if (i == 1)
			{

		fin.open("tbits1r.out");

		fin2.open("tbits1zr.out");
			}

			else if (i == 2)
			{

		fin.open("tbits2r.out");

		fin2.open("tbits2zr.out");
			}

			else if (i == 3)
			{

		fin.open("tbits3r.out");

		fin2.open("tbits3zr.out");
			}

			else if (i == 4)
			{

		fin.open("tbits4r.out");

		fin2.open("tbits4zr.out");
			}

			else if (i == 5)
			{

		fin.open("tbits5r.out");

		fin2.open("tbits5zr.out");
			}

			else if (i == 6)
			{

		fin.open("tbits6r.out");

		fin2.open("tbits6zr.out");
			}

			else if (i == 7)
			{

		fin.open("tbits7r.out");

		fin2.open("tbits7zr.out");
			}

			getline(fin,
line);
			getline(fin,
line);
			getline(fin,
line);
			number_term
= atoi(&line[3]);

			for (int k = 0;
k < number_term; k++)
			{

		getline(fin, line);

					for
(int j = 0; j<4; j++)
					{

		if (line[j] == '1')

		{

			sum = sum +
bit;

		}



		else if (line[j] == '-')

		{

			sum = sum +
2*bit;

		}


		bit = bit/3;
					}


		exist_t[sum]++;

		sum = 0;
					bit
= 27;
			}

			fin.close();

			getline(fin2,
line);
			getline(fin2,
line);
			getline(fin2,
line);

		number_term2 =
atoi(&line[3]);

			for (k = 0; k <
number_term2; k++)
			{

		getline(fin2, line);

					for
(int j = 0; j<4; j++)
					{

		if (line[j] == '1')

		{

			sum = sum +
bit;

		}



		else if (line[j] == '-')

		{

			sum = sum +
2*bit;

		}


		bit = bit/3;
					}


		exist_f[sum]++;

		sum = 0;
					bit
= 27;
			}

			fin2.close();

			fout << "
component and_or" << i << endl;
			fout << "  port
( " <<endl;

			for (int f =0;
f<77;f++)
			{
					if
(((exist_t[f] > 0)&&(exist_t[f] < 90))
|| ((exist_f[f] > 0)&&(exist_f[f] <
90)))
					{
```

```
		fout << "    x"<<f<<" : in
std_logic; " << endl;
                                        }
                        }

                        for (f = 0; f
<number_term; f++)
                        {

		fout << "    y" <<f<<" :
in std_logic; " <<endl;
                        }

                        for (f = 0; f
<number_term2; f++)
                        {

		fout << "    y"
<<52+f<<" : in std_logic; " <<endl;
                        }

                        for (f=0; f<77;
f++)
                        {

		exist_t[f] = 0;
                        }

                        for (f=0; f<77;
f++)
                        {

		exist_f[f] = 0;
                        }

                        fout << "
equ : in std_logic; " <<endl;
                        fout << "
out_t : out std_logic; " <<endl;
                        fout << "
out_z : out std_logic); " <<endl;

                        fout << " end
component;" << endl;
                        fout << " " << "
endl;
		}

		fout << "signal input:
std_logic_vector(7 downto
0);"<<endl;
		fout << "signal output:
std_logic_vector(7 downto
0);"<<endl;
		fout << "signal outputz:
std_logic_vector(7 downto
0);"<<endl;

		fout << "signal ";

		for (i = 0; i<8;i++)
		{
                        fout <<
"input" << i <<"z";

                        if (i != 7)
                        {

		fout <<", ";
                        }
		}

		fout << ":
STD_LOGIC;"<< endl;

		fout << "signal ";

		int t_i = 0;
		int t_j = 0;

		for (i = 0; i<8;i++)
		{
                        for (int j = 0;
j<8; j++)
                        {
                                if
(i<4)
                                {
		t_i =  7-i;
                                }

                                else
                                {
		t_i = 11-i;
                                }

                                if
(j<4)
                                {
		t_j = 7-j;
                                }

                                else
                                {
		t_j = 11-j;
                                }

		if(t_i > t_j)
                                {

		fout << "out_";

		if (i<4)
		{

                                fout << 7-i;
		}

		else
		{

                                fout << 11-
i<<"z";
		}

                                if (j<4)

                                {

                                        fout << 7-j;
                                }

                                else

                                {

                                        fout << 11-
j<<"z";
                                }

                                if ((i != 6) || (j != 7))

                                {

                                        fout <<", ";
                                }
                                        }
                                }
                        }

                        fout << " :
STD_LOGIC;"<< endl;

                        fout << "signal ";

                        for (i = 0; i<8;i++)
                        {
                                for (int j = 0;
j<8; j++)
                                {
                                        if
(i<4)
                                        {

                        t_i =  7-i;
                                        }

                                        else
                                        {

                        t_i = 11-i;
                                        }

                                        if
(j<4)
                                        {

                        t_j = 7-j;
                                        }

                                        else
                                        {

                        t_j = 11-j;
                                        }

                        if(t_i > t_j)
                                        {
```

```
		fout << "out_";

		if (i<4)
		{
			fout << 3-i;
		}

		else
		{
			fout << 7-i<<"z";
		}

		if (j<4)
		{
			fout << 3-j;
		}
		else
		{
			fout << 7-j<<"z";
		}

		if ((i != 6) || (j != 7))
		{
			fout <<", ";
		}
				}
			}
		}
		fout << " : STD_LOGIC;"<< endl;
		for (i = 0; i <run; i++)
		{
			if(i == 0)
			{
		fin.open("tbits0r.out");
		fin2.open("tbits0zr.out");
			}
			else if (i == 1)
			{
		fin.open("tbits1r.out");
		fin2.open("tbits1zr.out");
			}
			else if (i == 2)
			{
		fin.open("tbits2r.out");
		fin2.open("tbits2zr.out");
			}
			else if (i == 3)
			{
		fin.open("tbits3r.out");
		fin2.open("tbits3zr.out");
			}
			else if (i == 4)
			{
		fin.open("tbits4r.out");
		fin2.open("tbits4zr.out");
			}
			else if (i == 5)
			{
		fin.open("tbits5r.out");
		fin2.open("tbits5zr.out");
			}
			else if (i == 6)
			{
		fin.open("tbits6r.out");
		fin2.open("tbits6zr.out");
			}
			else if (i == 7)
			{
		fin.open("tbits7r.out");
		fin2.open("tbits7zr.out");
			}
			getline(fin, line);
			getline(fin, line);
			getline(fin, line);
			number_term = atoi(&line[3]);
			for (int k = 0; k < number_term; k++)
			{
		getline(fin, line);
			for (int j = 0; j<4; j++)
			{
		if (line[j] == '1')
		{
			sum = sum + bit;
		}
		else if (line[j] == '-')
		{
			sum = sum + 2*bit;
		}
		bit = bit/3;
			}
		exist_t[sum]++;
		sum = 0;
			bit = 27;
		}
			fin.close();
			getline(fin2, line);
			getline(fin2, line);
			getline(fin2, line);
		number_term2 = atoi(&line[3]);
			for (k = 0; k < number_term2; k++)
			{
		getline(fin2, line);
			for (int j = 0; j<4; j++)
			{
		if (line[j] == '1')
		{
			sum = sum + bit;
		}
		else if (line[j] == '-')
		{
```

```
                sum = sum +
2*bit;
        }

        bit = bit/3;
                }

        exist_f[sum]++;

        sum = 0;
                bit
= 27;
        }

        fin2.close();

        fout <<
"signal ";

        for (int f =0;
f<77;f++)
        {
                if
(((exist_t[f] > 0)&&(exist_t[f] < 90))
|| ((exist_f[f] > 0)&&(exist_f[f] <
90)))
                {

      fout << "x_"<<i<<f<<", ";
                }
        }

        fout << ":
std_logic; " <<endl;

        for (f=0; f<77;
f++)
        {

        exist_t[f] = 0;
        }

        for (f=0; f<77;
f++)
        {

        exist_f[f] = 0;
        }

        fout <<
"signal ";

        for (f = 0; f
<number_term; f++)
        {

      fout << "y_" << i <<f<<",
";
        }

        for (f = 0; f
<number_term2-1; f++)
        {

       fout << "y_" << i
<<52+f<<", ";
        }

                fout << "y_"
<< i << 52 + number_term2-1;

                fout << ":
std_logic; " <<endl;
        }

        fout << " "<<endl;

        fout << "begin" << endl;

        int num_d_care_x = 0;
        int num_d_care_y = 0;

//////////////////////////////////////////////////
////////////////////////////////
        fout << "
process(clk_1)" << endl;
        fout << "      begin" <<
endl;
        fout << "        if
(clk_1='1' and clk_1'event)
then"<<endl;
        fout << "            input
<= data_in;"<<endl;
        fout << "
data_out <= output;"<<endl;
        fout << "
data_outz <= outputz;"<<endl;
        fout << "        end if;
"<<endl;
        fout << "    end
process;"<<endl;
   fout << " " <<endl;

        for (int j = 0; j<8; j++)
        {
                fout <<
"input"<<j<<"z" << " <= " << " not
input("<<j<<");"<<endl;
                fout <<"
"<<endl;
        }

        for (i = 0; i<8;i++)
        {
                for (int j = 0;
j<8; j++)
                {
                        if
(i<4)
                        {

        t_i =  7-i;
                        }


                        else
                        {

        t_i = 11-i;
                        }


                        if
(j<4)
                        {

        t_j = 7-j;
                }
        else
                {

        t_j = 11-j;
                }

        if(t_i > t_j)
                {

        fout << "out_";

        if (i<4)

        {
                fout << 7-i;
        }

        else

        {
                fout << 11-
i<<"z";
        }


        if (j<4)

        {
                fout << 7-j;
        }

        else

        {
                fout << 11-
j<<"z";
        }


        fout << " <= ";


        if (i<4)

        {
                fout
<<"input("<< 7-i<<") ";

        }

        else
```

```
		{

			fout
<<"input"<< 11-i<<"z ";

		}


		fout << "nand " ;



		if (j<4)

		{

			fout
<<"input("<< 7-j<<"); ";

		}

		else

		{

			fout
<<"input"<< 11-j<<"z"<<"; ";

		}


		fout << " "<<endl;

		fout << " "<<endl;
					}
			}
		}

		for (i = 0; i<8;i++)
		{
			for (int j = 0;
j<8; j++)
			{
				if
(i<4)
				{

		t_i =  7-i;
				}


		else
				{

		t_i = 11-i;
				}


				if
(j<4)
				{

		t_j = 7-j;
				}

		else
				{

		t_j = 11-j;
				}

		if(t_i > t_j)
					{


		fout << "out_";


		if (i<4)

		{

			fout << 3-i;

		}


		else

		{

			fout << 7-
i<<"z";

		}



		if (j<4)

		{

			fout << 3-j;

		}

		else

		{

			fout << 7-
j<<"z";

		}


		fout << " <= " ;



		if (i<4)

		{

			fout
<<"input("<< 3-i<<") ";

		}


		else

		{

			fout
<<"input"<< 7-i<<"z ";

		}

		fout << "nand ";



		if (j<4)

		{

			fout
<<"input("<< 3-j<<"); ";

		}

		else

		{

			fout
<<"input"<< 7-j<<"z"<<"; ";

		}


		fout <<" "<<endl ;

		fout << " "<<endl;
					}
			}
		}

////////////////////////////////////////////////
////////////////////////////////////////
		for (i = 0; i<81;i++)
		{
			exist_x [i] = -
1;
			exist_y [i] = -
1;
		}

		for (int q = 0; q <2*run;
q++)
		{
			if (q == 0)
			{

		fin.open("tbits0r.out");

		base = 0;
			}

			else if (q ==
1)
			{

		fin.open("tbits0zr.out");

		base = 0;
			}

			else if (q ==
2)
			{

		fin.open("tbits1r.out");

		base = 1000;
			}
```

```
                else if (q ==
3)
                {
fin.open("tbits1zr.out");
base = 1000;
                }
                else if (q ==
4)
                {
fin.open("tbits2r.out");
base = 2000;
                }
                else if (q ==
5)
                {
fin.open("tbits2zr.out");
base = 2000;
                }
                else if (q ==
6)
                {
fin.open("tbits3r.out");
base = 3000;
                }
                else if (q ==
7)
                {
fin.open("tbits3zr.out");
base = 3000;
                }
                else if (q ==
8)
                {
fin.open("tbits4r.out");
base = 4000;
                }
                else if (q ==
9)
                {
fin.open("tbits4zr.out");
base = 4000;
                }
                else if (q ==
10)
                {
fin.open("tbits5r.out");
base = 5000;
                }
                else if (q ==
11)
                {
fin.open("tbits5zr.out");
base = 5000;
                }
                else if (q ==
12)
                {
fin.open("tbits6r.out");
base = 6000;
                }
                else if (q ==
13)
                {
fin.open("tbits6zr.out");
base = 6000;
                }
                else if (q ==
14)
                {
fin.open("tbits7r.out");
base = 7000;
                }
                else if (q ==
15)
                {
fin.open("tbits7zr.out");
base = 7000;
                }
                getline(fin,
line);
                getline(fin,
line);
                getline(fin,
line);
                number_term
= atoi(&line[3]);
                for (i = 0; i <
number_term; i++)
                {
        getline(fin, line);
                        for
(int j = 0; j<4; j++)
                        {
        if (line[j] == '1')
        {
                sum_x =
sum_x + bit;
        }
        else if (line[j] == '-')
        {
        num_d_care_x ++;
                sum_x =
sum_x + 2*bit;
        }
        bit = bit/3;
                }
                bit
= 27;
                for
(int k = 4; k<8; k++)
                {
        if (line[k] == '1')
        {
                sum_y =
sum_y + bit;
        }
        else if (line[k] == '-')
        {
        num_d_care_y ++;
                sum_y =
sum_y + 2*bit;
        }
        bit = bit/3;
                }
                bit
= 27;
                if
(current[sum_x] == 0)
                {
        current[sum_x] = 1;
        if (exist_x[sum_x] < 0)
```

```
            {

            exist_x[sum_x] = base
+ sum_x;

                        if
(num_d_care_x == 2)

                        {



            fout
<<"x_"<<q/2<<sum_x;



            fout << " <= not out_";

                                    for
(j = 0; j<4; j++)

                                    {

            if (line[j] == '1')

            {

                        fout << 7-j;

            }


            else if (line[j] == '0')

            {

                        fout << 7-j <<
"z";

            }
                                    }



            fout << " ;"<<endl;

            fout << " "<<endl;
                        }

                        else if
(num_d_care_x == 1)

                        {

                        fout
<<"x_"<<q/2<<sum_x;



            fout << " <= out_";

                                    int
count = 0;

                                    for
(j = 0; j<4; j++)

                                    {

            if (line[j] == '1')

            {

                        fout << 7-j;

                        count++;

            }


            else if (line[j] == '0')

            {

                        fout << 7-j <<
"z";

                        count++;

            }


            if (count == 2)

            {

                        break;
                        }

                        else if
(num_d_care_x == 0)

                        {

                        fout
<<"x_"<<q/2<<sum_x;

                        }

            fout << " nor input";

                                    for
(j = 3; j>1; j--)

                                    {

            if (line[j] == '1')

            {

                        fout << 7-j <<
"z";

                        break;

            }


            else if (line[j] == '0')

            {

                        fout <<"("<<
7-j<<")";

                        break;

            }
                                    }


            fout << " ;"<<endl;

            fout << " "<<endl;

                        }
```

```
                fout << " <= out_";

                        for
(j = 0; j<2; j++)

                        {

        if (line[j] == '1')

        {

                fout << 7-j;

        }

        else if (line[j] == '0')

        {

                fout << 7-j <<
"z";

        }

                        }

        fout << " nor out_";

                        for
(j = 2; j<4; j++)

                        {

        if (line[j] == '1')

        {

                fout << 7-j;

        }

        else if (line[j] == '0')

        {

                fout << 7-j <<
"z";

        }

                fout << " ;"<<endl;

                fout << " "<<endl;

                        }

                else if
(num_d_care_x == 3)

                        {

                                for
(int j = 0; j<4; j++)

                                {

        if (line[j] == '1')

        {

                fout <<
"x_"<<q/2<<sum_x;

                fout << " <=
input(" << 7-j <<");" << endl;

        }

        else if (line[j] == '0')

        {

                fout <<
"x_"<<q/2<<sum_x;

                fout << " <=
input" << 7-j <<"z;" << endl;

        }

                }

        fout << endl;

                }

                sum_x = 0;

        }

        else

        {

                if
((exist_x[sum_x]/1000)!=(q/2))

                {

        fout <<
"x_"<<q/2<<sum_x<<" <= "
<<"x_"<<
exist_x[sum_x]/1000<<exist_x[su
m_x]%1000 <<";"<< endl;

        fout <<endl;

                }

        sum_x = 0;
        }
                }

        else
                {

        sum_x = 0;
                }

        ////////////////////////////////////
////////////////////////////////////////////////////
////////////

                if
(exist_y[sum_y] < 0)
                {

        exist_y[sum_y] = base
+ i;

        if (q%2 == 1)

        {

        exist_y[sum_y] =
exist_y[sum_y] + 52;

        }
```

```
        else
        {
                true_term =
number_term;
        }

        if (num_d_care_y == 2)
        {
                fout
<<"y_"<<q/2;


                if (q%2 == 1)
                {

        fout << i + 52;
                }

                else
                {

        fout << i;

                }


                fout << " <=
not out_";

                for (j = 4; j<8;
j++)
                {
                        if
(line[j] == '1')
                        {

        fout << 7-j;
                        }


        else if (line[j] == '0')
                        {

        fout << 7-j << "z";
                        }
                }

                fout <<
" ;"<<endl;

                fout << "
"<<endl;
        }

        else if (num_d_care_y
== 1)
        {
                fout <<
"y_"<<q/2;


                if (q%2 == 1)
                {

        fout << i + 52;
                }

                else
                {

        fout << i;

                }


                fout << "  <=
out_";

                int count = 0;

                for (j = 4; j<8;
j++)
                {
                        if
(line[j] == '1')
                        {

        fout << 7-j;

        count++;
                        }
                }

        else if (line[j] == '0')
                        {

        fout << 7-j << "z";

        count++;
                        }

                        if
(count == 2)
                        {

        break;
                        }
                }

                fout << " nor
input";

                for (j = 7; j>5;
j--)
                {
                        if
(line[j] == '1')
                        {

        fout << 7-j << "z";

        break;
                        }

        else if (line[j] == '0')
                        {

        fout <<"("<< 7-j<<")";

        break;
                        }
                }

                fout <<
```

```
" ;"<<endl;
                fout << "
"<<endl;
        }
        else if (num_d_care_y
== 0)
        {
                fout
<<"y_"<<q/2;

                if (q%2 == 1)
                {
        fout << i + 52;
                }
                else
                {
        fout << i;
                }
                fout << " <=
out_";
                for (j = 4; j<6;
j++)
                {
                        if
(line[j] == '1')
                        {
        fout << 7-j;
                        }
        else if (line[j] == '0')
                        {
        fout << 7-j << "z";
                        }
                }
                fout << " nor
out_";
                for (j = 6; j<8;
j++)
                {
                        if
(line[j] == '1')
                        {
                fout << 7-j;
                        }
        else if (line[j] == '0')
                        {
        fout << 7-j << "z";
                        }
                }
                fout <<
" ;"<<endl;
                fout << "
"<<endl;
        }
        else if (num_d_care_y
== 3)
        {
                for (int j = 4;
j<8; j++)
                {
                        if
(line[j] == '1')
                        {
        fout << "y_"<<q/2;

        if (q%2 == 1)
        {
                fout << i + 52;
        }
        else
        {
                fout << i;
        }
        fout << " <=  input(" <<
7-j <<");" << endl;
                        }
        else if (line[j] == '0')
                        {
        fout << "y_"<<q/2;

        if (q%2 == 1)
        {
                fout << i + 52;
        }
        else
        {
                fout << i;
        }
        fout << " <=  input" <<
7-j <<"z;" << endl;
                        }
                }
```

```
                fout << endl;


        }


        sum_y = 0;
                }


        else
                {

        if (q%2 == 1)

        {

                fout
<<"y_"<<q/2<<i+52<<" <= " <<
"y_"<<exist_y[sum_y]/1000
<<exist_y[sum_y]%1000 <<";"<<
endl;

                fout <<endl;

        }


        else

        {

                fout
<<"y_"<<q/2<<i<<" <= "
<<"y_"<<exist_y[sum_y]/1000<<e
xist_y[sum_y]%1000 <<";"<< endl;

                fout <<endl;

        }


        sum_y = 0;
                }

        ////////////////////////////////////
////////////////////////////////////////////////////
////////

        num_d_care_x = 0;

        num_d_care_y = 0;


                }

                fin.close();

               if (q % 2 == 1)
                {
                        for
(int zz = 0; zz<77; zz++)
                {

        current[zz] = 0;
                }


       fout << "ao" << q/2 << ":
" << "and_or" <<q/2<<endl;
                fout << "port map (';


                if(q/2 == 0)
                        {

                fin.open("tbits0r.out");

                fin2.open("tbits0zr.out");
                        }


                else if (q/2 == 1)
                        {

                fin.open("tbits1r.out");

                fin2.open("tbits1zr.out");
                        }


                else if (q/2 == 2)
                        {

                fin.open("tbits2r.out");

                fin2.open("tbits2zr.out");
                        }


                else if (q/2 == 3)
                        {

                fin.open("tbits3r.out");

                fin2.open("tbits3zr.out");
                        }


                else if (q/2 == 4)
                        {

                fin.open("tbits4r.out");

                fin2.open("tbits4zr.out");
                        }


                else if (q/2 == 5)
                        {

                fin.open("tbits5r.out");

                fin2.open("tbits5zr.out");
                        }


                else if (q/2 == 6)
                        {

                fin.open("tbits6r.out");

                fin2.open("tbits6zr.out");
                        }


                else if (q/2 == 7)
                        {

                fin.open("tbits7r.out");

                fin2.open("tbits7zr.out");
                        }


                getline(fin, line);

                getline(fin, line);

                getline(fin, line);

                number_term =
atoi(&line[3]);

                        for
(int k = 0; k < number_term; k++)
                        {

                getline(fin, line);


                for (int j = 0; j<4; j++)

                {

                        if (line[j] ==
'1')

                        {

                sum = sum + bit;

                        }


                        else if (line[j]
== '-')

                        {

                sum = sum + 2*bit;

                        }


                        bit = bit/3;

                }


                exist_t[sum]++;

                sum = 0;

                bit = 27;
                        }


                fin.close();

                getline(fin2, line);

                getline(fin2, line);
```

```
            getline(fin2, line);

            number_term2 =
atoi(&line[3]);

                                    for
(k = 0; k < number_term2; k++)
                                    {
            getline(fin2, line);


            for (int j = 0; j<4; j++)

            {
                        if (line[j] ==
'1')
                        {


            sum = sum + bit;

                        }



                        else if (line[j]
== '-')
                        {


            sum = sum + 2*bit;

                        }


                        bit = bit/3;
                        }


                        exist_f[sum]++;
                        sum = 0;

                        bit = 27;
                                    }

                        fin2.close();
                                    for
(int f =0; f<77;f++)
                                    {
                        if (((exist_t[f] >
0)&&(exist_t[f] < 90)) || ((exist_f[f]
> 0)&&(exist_f[f] < 90)))

                        {
                                    fout <<
"x_"<<q/2<<f<<", ";

                        }
                                    }

                        fout << " "<<endl;
                                    for
(f = 0; f < number_term; f++)
                                    {
                        fout << "y_" << q/2 << f
<< ", " ;
                                    }
                                    for
(f=0; f< number_term2; f++)
                                    {
                        fout << "y_" << q/2 <<
f+52 << ", " ;
                                    }


                        fout << " clk_1, output("
<< q/2 << "), outputz(" << q/2 <<
") );"<<endl;

                        fout << endl;

                                    for
(f=0; f<77; f++)
                                    {
                        exist_t[f] = 0;
                                    }

                                    for
(f=0; f<77; f++)
                                    {
                        exist_f[f] = 0;
                                    }
                        }

            }

            fout << ""<< endl;

            fout << "end structure;"
<< endl;

            fout.close();

            return 0;
}
```

C++ code for *encryption_ao_domino*:

```
#include <fstream>
#include <string>
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

using namespace std;

int main()
{
            ofstream fout;
            ifstream fin;
            ifstream fin2;

            string line;

            int exist_t [77] = {0};
            int exist_f [77] = {0};
            int sum = 0;
            int bit = 27;
            int number_term = 0;
            int xpin = 2;
            int ypin = 1;
            int x = 0;
            int y = 0;
            int go_right = 1;
            int go_left = 1;
            int shift = 0;
            int yt = 0;
            int yb = 52;
            int right = 0;
            int left = 0;
            int done = 0;

            for (int i = 0; i <8; i++)
            {
                        if ( i == 0)
                        {

            fin.open("tbits0r.out");

            fin2.open("tbits0zr.out");

            fout.open("and_or0.il");
                        }
                        else if (i == 1)
                        {

            fin.open("tbits1r.out");

            fin2.open("tbits1zr.out");

            fout.open("and_or1.il");
                        }

                        else if (i == 2)
                        {

            fin.open("tbits2r.out");

            fin2.open("tbits2zr.out");

            fout.open("and_or2.il");
                        }

                        else if (i == 3)
                        {

            fin.open("tbits3r.out");

            fin2.open("tbits3zr.out");

            fout.open("and_or3.il");
                        }

                        else if (i == 4)
                        {
```

```
            fin.open("tbits4r.out");
            fin2.open("tbits4zr.out");
            fout.open("and_or4.il");
                }
            else if (i == 5)
                {
            fin.open("tbits5r.out");
            fin2.open("tbits5zr.out");
            fout.open("and_or5.il");
                }
            else if (i == 6)
                {
            fin.open("tbits6r.out");
            fin2.open("tbits6zr.out");
            fout.open("and_or6.il");
                }
            else if (i == 7)
                {
            fin.open("tbits7r.out");
            fin2.open("tbits7zr.out");
            fout.open("and_or7.il");
                }
                fout <<
"library=\"sfulib3\"" << endl;
                fout << endl;
                fout <<
"procedure(and_or"<<i<<"( )"<<e
ndl;
                fout <<
"let( ( cvw cnm x y x1 y1 l0 l1
rod1 rod2 pt1 pt2 )" << endl;
                fout << endl;
                fout << "
sprintf(cnm \"and_or" << i <<
"\" )"<<endl;
                fout << endl;
                fout << "
cell=dbOpenCellViewByType(libr
ary cnm \"layout\" \"maskLayout\"
\"w\")" << endl;
                fout << "
printf(\"Cellname: %s\" cnm)" <<
endl;
                fout << endl;
                fout << "x =
0" << endl;
                fout << "y =
0" << endl;
                fout << endl;
                getline(fin,
line);
                getline(fin,
line);
                getline(fin,
line);
                number_term
= atoi(&line[3]);
                for (int k = 0;
k < number_term; k++)
                {
            getline(fin, line);
                        for
(int j = 0; j<4; j++)
                        {
            if (line[j] == '1')
            {
                        sum = sum +
bit;
            }
            else if (line[j] == '-')
            {
                        sum = sum +
2*bit;
            }
            bit = bit/3;
                        }
            exist_t[sum]++;
            sum = 0;
                        bit
= 27;
                }
                getline(fin2,
line);
                getline(fin2,
line);
                getline(fin2,
line);
                number_term
= atoi(&line[3]);
                for (k = 0; k <
number_term; k++)
                {
            getline(fin2, line);
                        for
(int j = 0; j<4; j++)
                        {
            if (line[j] == '1')
            {
                        sum = sum +
bit;
            }
            else if (line[j] == '-')
            {
                        sum = sum +
2*bit;
            }
            bit = bit/3;
                        }
            exist_f[sum]++;
            sum = 0;
                        bit
= 27;
                }
                //top
                for (int h=0;
h<77; h++)
                {
            if((exist_t[h] > 1) &&
(exist_t[h] < 90))
                        {
            for (int d = 0;
d<exist_t[h] - 1; d++)
            {
                        if (exist_t[h-1]
== 0)
                        {
            exist_t[h-1] = 100 + h;
            done++;
                        }
                        else if
(exist_t[h+1] == 0)
                        {
            exist_t[h+1] = 100 + h;
            done++;
                        }
```

```
                else if
((exist_t[h-1]==100+h) &&
(exist_t[h-2] == 0))

                {


        exist_t[h-2] = 100 + h;


        done++;

                }

                else if
((exist_t[h+1]==100+h) &&
(exist_t[h+2] == 0))

                {


        exist_t[h+2] = 100 + h;


        done++;

                }

        }


        exist_t[h] = exist_t[h] -
done;

        done = 0;
                        }
                }

                //bot
                for ( h=0;
h<77; h++)
                {

        if((exist_f[h] > 1) &&
(exist_f[h] < 90))
                        {

        for (int d = 0;
d<exist_f[h] - 1; d++)

        {

                if (exist_f[h-1]
== 0)

                {


        exist_f[h-1] = 100 + h;


        done++;

                }


                else if
(exist_f[h+1] == 0)

                {


        exist_f[h+1] = 100 + h;


        done++;

                }

                else if
((exist_f[h-1]==100+h) &&
(exist_f[h-2] == 0))

                {


        exist_f[h-2] = 100 + h;


        done++;

                }

                else if
((exist_f[h+1]==100+h) &&
(exist_f[h+2] == 0))

                {


        exist_f[h+2] = 100 + h;


        done++;

                }

        }

        exist_f[h] = exist_f[h] -
done;

        done = 0;
                        }
                }

                //top2
                for ( h=0;
h<77; h++)
                {

        if((exist_t[h] > 1) &&
(exist_t[h] < 90))
                        {

        for (int d = 0;
d<exist_t[h] - 1; d++)

        {

                for (int v = 4;
v < 20; v++)

                {


        shift = v/2;


                if
((exist_t[h-shift] > 1) &&
(exist_t[h-shift] < 90))

                {


        go_left = 0;


                }


                if
((exist_t[h+shift] > 1) &&
(exist_t[h+shift] < 90))

                {


        go_right = 0;


                }


                if
((go_left == 1) && (exist_t[h-shift]
== 0))

                {


        exist_t[h-shift] = 200 +
h;


        break;


                }


        else if ((go_right == 1)
&& (exist_t[h+shift] == 0))

                {


        exist_t[h+shift] = 200 +
h;


        break;


                }

                }

        }

        go_right = 1;

        go_left = 1;

                }
                }

                //bot2
```

```
                    for (h=0;
h<77; h++)
                    {

          if((exist_f[h] > 1) &&
(exist_f[h] < 90))
                              {

          for (int d = 0;
d<exist_f[h] - 1; d++)

          {

                    for (int v = 4;
v < 20; v++)
                    {


          shift = v/2;




                              //if
(((h-shift) == 0) && (exist_f[h-shift]
> 0))

                              if
((exist_f[h-shift] > 1) &&
(exist_f[h-shift] < 90))

                              {


          go_left = 0;

                              }


                              if
((exist_f[h+shift] > 1) &&
(exist_f[h+shift] < 90))

                              {


          go_right = 0;

                              }



                              if
((go_left == 1) && (exist_f[h-shift]
== 0))

                              {


          exist_f[h-shift] = 200 +
h;


          break;

                              }



          else if ((go_right == 1)
&& (exist_f[h+shift] == 0))

                              {


          exist_f[h+shift] = 200 +
h;


          break;

                              }

                    }

          }

          go_right = 1;

          go_left = 1;

                              }
                    }

                    for (h = 0; h <
77; h++)
                    {

          cout << exist_t[h] << " ";
                    }

                    cout << " " <<
endl;

                    for (h = 0; h <
77; h++)
                    {

          cout << exist_f[h] << " ";
                    }

                    cout << " " <<
endl;

                    //TOP
                    if(exist_t[0]
== 0)
                    {

          fout << "cvw =
dbOpenCellViewByType( library
\"feTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;

          fout << " "<< endl;

          fout << "I0 =
dbCreateInst(cell cvw \"I0\"
list("<< " R->Xsp 0 ) \"R0\" )" <<
endl;

          fout << " " << endl;

          fout << "
dbClose( cvw )" << endl;

          fout << " " << endl;
                    }
                    else if
((exist_t[0] > 90) && (exist_t[0] <
190))
                    {

          fout << "cvw =
dbOpenCellViewByType( library
\"fsTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;

          fout << " "<< endl;

          fout << "I0 =
dbCreateInst(cell cvw \"I0\"
list("<< " R->Xsp 0 ) \"R0\" )" <<
endl;

          fout << " " << endl;

          fout << "
dbClose( cvw )" << endl;

          fout << " " << endl;
                    }
                    else if
(exist_t[0] > 190)
                    {

          fout << "cvw =
dbOpenCellViewByType( library
\"fsTop2\" \"layout\"
\"maskLayout\" \"r\" )" << endl;

          fout << " "<< endl;

          fout << "I0 =
dbCreateInst(cell cvw \"I0\"
list("<< " R->Xsp 0 ) \"R0\" )" <<
endl;

          fout << " " << endl;

          fout << "
dbClose( cvw )" << endl;

          fout << " " << endl;
                    }
                    else
                    {

          fout << "cvw =
dbOpenCellViewByType( library
\"fTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;

          fout << " "<< endl;

          fout << "I0 =
dbCreateInst(cell cvw \"I0\"
list("<< " R->Xsp 0 ) \"R0\" )" <<
endl;

          fout << " " << endl;

          fout << "
dbClose( cvw )" << endl;

          fout << " " << endl;
                    }

                    x = x+2;
```

```
                    fout << "cvw
= dbOpenCellViewByType( library
\"Top\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                    fout <<" "<<
endl;

                    for (k = 1; k <
14; k++)
                    {
                              x =
k*2;

                              if
((exist_t[k] > 0)&&(exist_t[k] < 90))
                              {

          if (k%2 == 1)

          {

                    x = x + 2;

          }


          fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

          if (k%2 == 1)

          {

                    fout <<
"\"MY\" )" << endl;

          }

          else

          {

                    fout <<
"\"R0\" )" << endl;

          }
                              }
                    }

                    fout << " " <<
endl;
                    fout << "
dbClose( cvw )" << endl;
                    fout << " " <<
endl;

                    fout << "cvw
= dbOpenCellViewByType( library
\"eTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                    fout <<" "<<
endl;

                    for (k = 1; k <
14; k++)
                    {
                              x =
k*2;

                              if
(exist_t[k] == 0)
                              {

          fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) \"R0\" )" << endl;
                              }
                    }

                    fout << " " <<
endl;
                    fout << "
dbClose( cvw )" << endl;
                    fout << " " <<
endl;

                    fout << "cvw
= dbOpenCellViewByType( library
\"sTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                    fout <<" "<<
endl;

                    for (k = 1; k <
14; k++)
                    {
                              x =
k*2;

                              if
((exist_t[k] > 90) && (exist_t[k] <
190))
                              {

          if (k%2 == 1)

          {

                    x = x + 2;

          }


          fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

          if (k%2 == 1)

          {

                    fout <<
"\"MY\" )" << endl;

          }

          else

          {

                    fout <<
"\"R0\" )" << endl;

          }
                              }
                    }

                    fout << " " <<
endl;
                    fout << "
dbClose( cvw )" << endl;
                    fout << " " <<
endl;

                    fout << "cvw
= dbOpenCellViewByType( library
\"sTop2\" \"layout\"
\"maskLayout\" \"r\" )" << endl;
                    fout <<" "<<
endl;

                    for (k = 1; k <
14; k++)
                    {
                              x =
k*2;

                              if
(exist_t[k] > 190)
                              {

          if (k%2 == 1)

          {

                    x = x + 2;

          }


          fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

          if (k%2 == 1)

          {

                    fout <<
"\"MY\" )" << endl;

          }

          else

          {

                    fout <<
"\"R0\" )" << endl;

          }
                              }
                    }

                    fout << " " <<
endl;
                    fout << "
dbClose( cvw )" << endl;
                    fout << " " <<
endl;

                    fout << "cvw
= dbOpenCellViewByType( library
\"pSub\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
```

```
                        fout << " "<<
endl;
                        fout << "I198
=  dbCreateInst(cell cvw \"I198\"
list(" << " 28*R->Xsp+R->Xsp 0 )
\"R0\" )" << endl;
                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"Top\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 14; k
< 42; k++)
                        {
                                x =
(k+2)*2;

                                if
((exist_t[k] > 0)&&(exist_t[k] < 90))
                                {

        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"eTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 14; k
< 42; k++)
                        {
                                x =
(k+2)*2;

                                if
(exist_t[k] == 0)
                                {

        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) \"R0\" )" << endl;
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"sTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 14; k
< 42; k++)
                        {
                                x =
(k+2)*2;

                                if
((exist_t[k] > 90) && (exist_t[k] <
190))
                                {

        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"sTop2\" \"layout\"
\"maskLayout\" \"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 14; k
< 42; k++)
                        {
                                x =
(k+2)*2;

                                if
(exist_t[k] > 190)
                                {

        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
```

```
                        }
                }


                fout << " " <<
endl;
                fout << "
dbClose( cvw )" << endl;
                fout << " " <<
endl;

                fout << "cvw
= dbOpenCellViewByType( library
\"pSub\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                fout << " "<<
endl;
                fout << "I199
=  dbCreateInst(cell cvw \"I199\"
list(" << " 88*R->Xsp+R->Xsp 0 )
\"R0\" )" << endl;
                fout << " " <<
endl;
                fout << "
dbClose( cvw )" << endl;
                fout << " " <<
endl;

                fout << "cvw
= dbOpenCellViewByType( library
\"Top\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                fout <<" "<<
endl;

                for (k = 42; k
< 70; k++)
                {
                        x =
(k+4)*2;

                        if
((exist_t[k] > 0)&&(exist_t[k] < 90))
                        {
        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else
        {

                fout <<
"\"R0\" )" << endl;

        }

                        }
                }

                fout << " " <<
endl;
                fout << "
dbClose( cvw )" << endl;
                fout << " " <<
endl;

                fout << "cvw
= dbOpenCellViewByType( library
\"eTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                fout <<" "<<
endl;

                for (k = 42; k
< 70; k++)
                {
                        x =
(k+4)*2;

                        if
(exist_t[k] == 0)
                        {

        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) \"R0\" )" << endl;
                        }
                }

                fout << " " <<
endl;
                fout << "
dbClose( cvw )" << endl;
                fout << " " <<
endl;

                fout << "cvw
= dbOpenCellViewByType( library
\"sTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                fout <<" "<<
endl;

                for (k = 42; k
< 70; k++)
                {
                        x =
(k+4)*2;

                        if
((exist_t[k] > 90) && (exist_t[k] <
190))
                        {

        if (k%2 == 1)

        {

                x = x + 2;

        }

        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }

                        }
                }

                fout << " " <<
endl;
                fout << "
dbClose( cvw )" << endl;
                fout << " " <<
endl;

                fout << "cvw
= dbOpenCellViewByType( library
\"sTop2\" \"layout\"
\"maskLayout\" \"r\" )" << endl;
                fout <<" "<<
endl;

                for (k = 42; k
< 70; k++)
                {
                        x =
(k+4)*2;

                        if
(exist_t[k] > 190)
                        {

        if (k%2 == 1)

        {

                x = x + 2;

        }

        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

        if (k%2 == 1)

        {
```

```
                        fout <<
"\"MY\" )" << endl;

                }

                else

                {

                        fout <<
"\"R0\" )" << endl;

                }
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"pSub\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout << " "<<
endl;
                        fout << "I200
=  dbCreateInst(cell cvw \"I200\"
list(" << " 148*R->Xsp+R->Xsp 0 )
\"R0\" )" << endl;
                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"Top\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 70; k
< 77; k++)
                        {
                                x =
(k+6)*2;

                                if
((exist_t[k] > 0)&&(exist_t[k] < 90))
                                {

                if (k%2 == 1)

                {

                        x = x + 2;

                }


                fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

                if (k%2 == 1)

                {

                        fout <<
"\"MY\" )" << endl;

                }

                else

                {

                        fout <<
"\"R0\" )" << endl;

                }
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"eTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 70; k
< 77; k++)
                        {
                                x =
(k+6)*2;

                                if
(exist_t[k] == 0)
                                {

                fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) \"R0\" )" << endl;
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"sTop\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 70; k
< 77; k++)
                        {
                                x =
(k+6)*2;

                                if
((exist_t[k] > 90) && (exist_t[k] <
190))
                                {

                if (k%2 == 1)

                {

                        x = x + 2;

                }


                fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

                if (k%2 == 1)

                {

                        fout <<
"\"MY\" )" << endl;

                }

                else

                {

                        fout <<
"\"R0\" )" << endl;

                }
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"sTop2\" \"layout\"
\"maskLayout\" \"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 70; k
< 77; k++)
                        {
                                x =
(k+6)*2;

                                if
(exist_t[k] > 190)
                                {

                if (k%2 == 1)

                {
```

```
                x = x + 2;

        }


        fout << "I" << k << " =
dbCreateInst(cell cvw \"I" << k <<
"\" list("<< x << "*R->Xsp+R->Xsp
0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                                }
                }

                fout << " " <<
endl;
                fout << "
dbClose( cvw )" << endl;
                fout << " " <<
endl;

                //Bot
                if(exist_f[0]
== 0)
                {

        fout << "cvw =
dbOpenCellViewByType( library
\"feBot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;

        fout << " "<< endl;

        fout << "I100 =
dbCreateInst(cell cvw \"I100\"
list("<< " R->Xsp 0 ) \"R0\" )" <<
endl;

        fout << " " << endl;

        fout << "
dbClose( cvw )" << endl;

        fout << " " << endl;
                }
                else if
((exist_f[0] > 90) && (exist_f[0] <
190))
                {

        fout << "cvw =
dbOpenCellViewByType( library
\"fsBot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;

        fout << " "<< endl;

        fout << "I100 =
dbCreateInst(cell cvw \"I100\"
list("<< " R->Xsp 0 ) \"R0\" )" <<
endl;

        fout << " " << endl;

        fout << "
dbClose( cvw )" << endl;

        fout << " " << endl;
                }
                else if
(exist_f[0] > 190)
                {

        fout << "cvw =
dbOpenCellViewByType( library
\"fsBot2\" \"layout\"
\"maskLayout\" \"r\" )" << endl;

        fout << " "<< endl;

        fout << "I100 =
dbCreateInst(cell cvw \"I100\"
list("<< " R->Xsp 0 ) \"R0\" )" <<
endl;

        fout << " " << endl;

        fout << "
dbClose( cvw )" << endl;

        fout << " " << endl;
                }
                else
                {

        fout << "cvw =
dbOpenCellViewByType( library
\"fBot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;

        fout << " "<< endl;

        fout << "I100 =
dbCreateInst(cell cvw \"I100\"
list("<< " R->Xsp 0 ) \"R0\" )" <<
endl;

        fout << " " << endl;

        fout << "
dbClose( cvw )" << endl;

        fout << " " << endl;
                }

                x = x+2;

                fout << "cvw
= dbOpenCellViewByType( library
\"Bot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                fout <<" "<<
endl;

                for (k = 1; k <
14; k++)
                {
                        x =
k*2;

                        if
((exist_f[k] > 0) && (exist_f[k] <
90))
                        {
        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k <<
" =  dbCreateInst(cell cvw \"I" <<
100+k << "\" list("<< x << "*R-
>Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                        }
                }

                fout << " " <<
endl;
                fout << "
dbClose( cvw )" << endl;
                fout << " " <<
endl;

                fout << "cvw
= dbOpenCellViewByType( library
\"eBot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                fout <<" "<<
endl;

                for (k = 1; k <
14; k++)
                {
                        x =
k*2;

                        if
(exist_f[k] == 0)
                        {

        fout << "I" << 100+k <<
```

```
" =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R->Xsp+R->Xsp 0 ) \"R0\" )" << endl;
                                        }
                                }

                                fout << " " << endl;
                                fout << "dbClose( cvw )" << endl;
                                fout << " " << endl;

                                fout << "cvw = dbOpenCellViewByType( library \"sBot\" \"layout\" \"maskLayout\" \"r\" )" << endl;
                                fout <<" "<< endl;

                                for (k = 1; k < 14; k++)
                                {
                                        x = k*2;

                                        if ((exist_f[k] > 90) && (exist_f[k] < 190))
                                        {
                        if (k%2 == 1)

                        {

                                x = x + 2;

                        }


                        fout << "I" << 100+k << " =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R->Xsp+R->Xsp 0 ) ";

                        if (k%2 == 1)

                        {

                                fout << "\"MY\" )" << endl;

                        }

                        else

                        {

                                fout << "\"R0\" )" << endl;

                        }
                                        }
                                }

                                fout << " " << endl;
                                fout << "dbClose( cvw )" << endl;
                                fout << " " << endl;

                                fout << "cvw = dbOpenCellViewByType( library \"sBot2\" \"layout\" \"maskLayout\" \"r\" )" << endl;
                                fout <<" "<< endl;

                                for (k = 1; k < 14; k++)
                                {
                                        x = k*2;

                                        if (exist_f[k] > 190)
                                        {
                        if (k%2 == 1)

                        {

                                x = x + 2;

                        }


                        fout << "I" << 100+k << " =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R->Xsp+R->Xsp 0 ) ";

                        if (k%2 == 1)

                        {

                                fout << "\"MY\" )" << endl;

                        }

                        else

                        {

                                fout << "\"R0\" )" << endl;

                        }
                                        }
                                }

                                fout << " " << endl;
                                fout << "dbClose( cvw )" << endl;
                                fout << " " << endl;

                                fout << "cvw = dbOpenCellViewByType( library \"Bot\" \"layout\" \"maskLayout\" \"r\" )" << endl;
                                fout <<" "<< endl;

                                for (k = 14; k < 42; k++)
                                {
                                        x = (k+2)*2;

                                        if ((exist_f[k] > 0) && (exist_f[k] < 90))
                                        {
                        if (k%2 == 1)

                        {

                                x = x + 2;

                        }


                        fout << "I" << 100+k << " =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R->Xsp+R->Xsp 0 ) ";

                        if (k%2 == 1)

                        {

                                fout << "\"MY\" )" << endl;

                        }

                        else

                        {

                                fout << "\"R0\" )" << endl;

                        }
                                        }
                                }

                                fout << " " << endl;
                                fout << "dbClose( cvw )" << endl;
                                fout << " " << endl;

                                fout << "cvw = dbOpenCellViewByType( library \"eBot\" \"layout\" \"maskLayout\" \"r\" )" << endl;
                                fout <<" "<< endl;

                                for (k = 14; k < 42; k++)
                                {
                                        x = (k+2)*2;

                                        if (exist_f[k] == 0)
                                        {

                        fout << "I" << 100+k << " =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R-
```

```
>Xsp+R->Xsp 0 ) \"R0\" )" <<
endl;
                                        }
                                }

                                fout << " " <<
endl;
                                fout << "
dbClose( cvw )" << endl;
                                fout << " " <<
endl;

                                fout << "cvw
= dbOpenCellViewByType( library
\"sBot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                                fout <<" "<<
endl;

                                for (k = 14; k
< 42; k++)
                                {
                                        x =
(k+2)*2;

                                        if
((exist_f[k] > 90) && (exist_f[k] <
190))
                                        {

        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k <<
" =  dbCreateInst(cell cvw \"I" <<
100+k << "\" list("<< x << "*R-
>Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                                        }
                                }

                                fout << " " <<
endl;
                                fout << "
dbClose( cvw )" << endl;
                                fout << " " <<
endl;

                                fout << "cvw
= dbOpenCellViewByType( library
\"sBot2\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                                fout <<" "<<
endl;

                                for (k = 14; k
< 42; k++)
                                {
                                        x =
(k+2)*2;

                                        if
(exist_f[k] > 190)
                                        {

        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k <<
" =  dbCreateInst(cell cvw \"I" <<
100+k << "\" list("<< x << "*R-
>Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                                        }
                                }

                                fout << " " <<
endl;
                                fout << "
dbClose( cvw )" << endl;
                                fout << " " <<
endl;

                                fout << "cvw
= dbOpenCellViewByType( library
\"Bot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                                fout <<" "<<
endl;

                                for (k = 42; k
< 70; k++)
                                {
                                        x =
(k+4)*2;

                                        if
((exist_f[k] > 0) && (exist_f[k] <
90))
                                        {

        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k <<
" =  dbCreateInst(cell cvw \"I" <<
100+k << "\" list("<< x << "*R-
>Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                                        }
                                }

                                fout << " " <<
endl;
                                fout << "
dbClose( cvw )" << endl;
                                fout << " " <<
endl;

                                fout << "cvw
= dbOpenCellViewByType( library
\"eBot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                                fout <<" "<<
endl;

                                for (k = 42; k
< 70; k++)
                                {
                                        x =
(k+4)*2;

                                        if
(exist_f[k] == 0)
                                        {

        fout << "I" << 100+k <<
" =  dbCreateInst(cell cvw \"I" <<
100+k << "\" list("<< x << "*R-
>Xsp+R->Xsp 0 ) \"R0\" )" <<
endl;
```

```
                        }
                }

                fout << " " << endl;
                fout << "dbClose( cvw )" << endl;
                fout << " " << endl;

                fout << "cvw = dbOpenCellViewByType( library \"sBot\" \"layout\" \"maskLayout\" \"r\" )" << endl;
                fout <<" "<< endl;

                for (k = 42; k < 70; k++)
                {
                        x = (k+4)*2;

                        if ((exist_f[k] > 90) && (exist_f[k] < 190))
                        {
        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k << " =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R->Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout << "\"MY\" )" << endl;

        }

        else

        {

                fout << "\"R0\" )" << endl;

        }
                        }
                }

                fout << " " << endl;
                fout << "dbClose( cvw )" << endl;
                fout << " " << endl;

                fout << "cvw = dbOpenCellViewByType( library \"sBot2\" \"layout\" \"maskLayout\" \"r\" )" << endl;
                fout <<" "<< endl;

                for (k = 42; k < 70; k++)
                {
                        x = (k+4)*2;

                        if (exist_f[k] > 190)
                        {
        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k << " =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R->Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout << "\"MY\" )" << endl;

        }

        else

        {

                fout << "\"R0\" )" << endl;

        }
                        }
                }

                fout << " " << endl;
                fout << "dbClose( cvw )" << endl;
                fout << " " << endl;

                fout << "cvw = dbOpenCellViewByType( library \"Bot\" \"layout\" \"maskLayout\" \"r\" )" << endl;
                fout <<" "<< endl;

                for (k = 70; k < 77; k++)
                {
                        x = (k+6)*2;

                        if ((exist_f[k] > 0) && (exist_f[k] < 90))
                        {
        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k << " =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R->Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout << "\"MY\" )" << endl;

        }

        else

        {

                fout << "\"R0\" )" << endl;

        }
                        }
                }

                fout << " " << endl;
                fout << "dbClose( cvw )" << endl;
                fout << " " << endl;

                fout << "cvw = dbOpenCellViewByType( library \"eBot\" \"layout\" \"maskLayout\" \"r\" )" << endl;
                fout <<" "<< endl;

                for (k = 70; k < 77; k++)
                {
                        x = (k+6)*2;

                        if (exist_f[k] == 0)
                        {
        fout << "I" << 100+k << " =  dbCreateInst(cell cvw \"I" << 100+k << "\" list("<< x << "*R->Xsp+R->Xsp 0 ) \"R0\" )" << endl;
                        }
                }
```

```
                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"sBot\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 70; k
< 77; k++)
                        {
                                x =
(k+6)*2;

                                if
((exist_f[k] > 90) && (exist_f[k] <
190))
                                {

        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k <<
" =  dbCreateInst(cell cvw \"I" <<
100+k << "\" list("<< x << "*R-
>Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout << "cvw
= dbOpenCellViewByType( library
\"sBot2\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout <<" "<<
endl;

                        for (k = 70; k
< 77; k++)
                        {
                                x =
(k+6)*2;

                                if
(exist_f[k] > 190)
                                {

        if (k%2 == 1)

        {

                x = x + 2;

        }


        fout << "I" << 100+k <<
" =  dbCreateInst(cell cvw \"I" <<
100+k << "\" list("<< x << "*R-
>Xsp+R->Xsp 0 ) ";

        if (k%2 == 1)

        {

                fout <<
"\"MY\" )" << endl;

        }

        else

        {

                fout <<
"\"R0\" )" << endl;

        }
                                }
                        }

                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        //DFF
                        fout << "cvw
= dbOpenCellViewByType( library
\"Dff\" \"layout\" \"maskLayout\"
\"r\" )" << endl;
                        fout << " "<<
endl;
                        fout << "I201
=  dbCreateInst(cell cvw \"I201\"
list(" << " 166*R->Xsp+R->Xsp 0 )
\"R0\" )" << endl;
                        fout << " " <<
endl;
                        fout << "
dbClose( cvw )" << endl;
                        fout << " " <<
endl;

                        fout <<
"mkRodPin(cell \"equ"<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<"169*R->Xsp+R->Xsp2
4*R->Ysp))"<<endl;
                        fout <<
"mkRodPin(cell \"out_t"<<"\"
\"metal1\" \"output\" list( \"none\" )
list("<<"172*R->Xsp+R->Xsp2
5*R->Ysp))"<<endl;
                        fout <<
"mkRodPin(cell \"out_z"<<"\"
\"metal1\" \"output\" list( \"none\" )
list("<<"175*R->Xsp+R->Xsp2
5*R->Ysp))"<<endl;
                        fout <<
"mkPWR(cell \"VSS\" \"metal1\"
\"inputOutput\" list(80*R->Xsp R-
>HNsub) list(0:-0.2 177*R-
>Xsp:R->HNsub+R-
>MaIcv) )"<<endl;
                        fout <<
"mkPWR(cell \"VDD\" \"metal1\"
\"inputOutput\" list(80*R->Xsp R-
>HPsub) list(0:R->HPsub-R-
>MaIcv 177*R->Xsp:11*R-
>Ysp+ovlp) )"<<endl;
                        fout << " "
<<endl;

                        for (h = 0; h <
77; h++)
                        {

                cout << exist_t[h] << " ";
                        }

                        cout << " " <<
endl;

                        for (h = 0; h <
77; h++)
                        {

                cout << exist_f[h] << " ";
                        }


                        cout << " " <<
endl;


                        for (int z=0;
z<14; z++)
                        {
                                if
(z%2 == 0)
                                {


                if ( ((exist_t[z] > 0) &&
(exist_t[z] < 90)) || ((exist_f[z] > 0)
&& (exist_f[z] < 90)) )

                {

                        fout <<
```

```
"mkRodPin(cell \"x"<<z<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<xpin<<"*R->Xsp+R->Xsp2
5*R->Ysp))"<<endl;

                fout << " " <<
endl;

        }

                        }

        else
                        {

        if ( ((exist_t[z] > 0) &&
(exist_t[z] < 90)) || ((exist_f[z] > 0)
&& (exist_f[z] < 90)) )

        {

                fout <<
"mkRodPin(cell \"x"<<z<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<xpin-1<<"*R->Xsp+R-
>Xsp2 5*R->Ysp))"<<endl;

                fout << " " <<
endl;

        }
                        }

        xpin = xpin+2;
                }

               xpin = xpin+4;

                for (z=14;
z<42; z++)
                {
                        if
(z%2 == 0)
                        {


        if ( ((exist_t[z] > 0) &&
(exist_t[z] < 90)) || ((exist_f[z] > 0)
&& (exist_f[z] < 90)) )

        {

                fout <<
"mkRodPin(cell \"x"<<z<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<xpin<<"*R->Xsp+R->Xsp2
5*R->Ysp))"<<endl;

                fout << " " <<
endl;

        }
                        }


        else
                        {

        if ( ((exist_t[z] > 0) &&
(exist_t[z] < 90)) || ((exist_f[z] > 0)
&& (exist_f[z] < 90)) )

        {

                        fout <<
"mkRodPin(cell \"x"<<z<<"\"
\"metal1\" \"input\" list( \"none\")
list("<<xpin-1<<"*R->Xsp+R-
>Xsp2 5*R->Ysp))"<<endl;

                        fout << " " <<
endl;

        }
                        }

        xpin = xpin+2;
                }

               xpin = xpin+4;

                for (z=42;
z<70; z++)
                {
                        if
(z%2 == 0)
                        {


        if ( ((exist_t[z] > 0) &&
(exist_t[z] < 90)) || ((exist_f[z] > 0)
&& (exist_f[z] < 90)) )

        {

                        fout <<
"mkRodPin(cell \"x"<<z<<"\"
\"metal1\" \"input\" list( \"none\")
list("<<xpin<<"*R->Xsp+R->Xsp2
5*R->Ysp))"<<endl;

                        fout << " " <<
endl;

        }
                        }

        else
                        {

        if ( ((exist_t[z] > 0) &&
(exist_t[z] < 90)) || ((exist_f[z] > 0)
&& (exist_f[z] < 90)) )

        {

                        fout <<
"mkRodPin(cell \"x"<<z<<"\"
\"metal1\" \"input\" list( \"none\")
list("<<xpin-1<<"*R->Xsp+R-
>Xsp2 5*R->Ysp))"<<endl;

                        fout << " " <<
endl;

        }
                        }

        xpin = xpin+2;
                }

               xpin = xpin+4;

                for (z=70;
z<77; z++)
                {
                        if
(z%2 == 0)
                        {


        if ( ((exist_t[z] > 0) &&
(exist_t[z] < 90)) || ((exist_f[z] > 0)
&& (exist_f[z] < 90)) )

        {

                        fout <<
"mkRodPin(cell \"x"<<z<<"\"
\"metal1\" \"input\" list( \"none\")
list("<<xpin<<"*R->Xsp+R->Xsp2
5*R->Ysp))"<<endl;

                        fout << " " <<
endl;

        }
                        }

        else
                        {

        if ( ((exist_t[z] > 0) &&
(exist_t[z] < 90)) || ((exist_f[z] > 0)
&& (exist_f[z] < 90)) )

        {

                        fout <<
"mkRodPin(cell \"x"<<z<<"\"
\"metal1\" \"input\" list( \"none\")
list("<<xpin-1<<"*R->Xsp+R-
>Xsp2 5*R->Ysp))"<<endl;

                        fout << " " <<
endl;

        }
                        }

        xpin = xpin+2;
                }

                fout <<" "
<<endl;

                for (z=0;
z<14; z++)
                {
                        if
(z%2 == 0)
                        {

        if (exist_t[z] > 0)

        {

                        fout <<
"mkRodPin(cell \"y"<<yt<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin<<"*R->Xsp+R->Xsp2
```

```
7*R->Ysp))"<<endl;
                        yt++;
                }

                if (exist_f[z] > 0)

                {

                        fout <<
"mkRodPin(cell \"y"<<yb<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin<<"*R->Xsp+R->Xsp2
3*R->Ysp))"<<endl;
                        yb++;
                }
                                }

                else
                                {
                if (exist_t[z] > 0)

                {

                        fout <<
"mkRodPin(cell \"y"<<yt<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin+1<<"*R->Xsp+R-
>Xsp2 7*R->Ysp))"<<endl;
                        yt++;
                }


                if (exist_f[z] > 0)

                {

                        fout <<
"mkRodPin(cell \"y"<<yb<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin+1<<"*R->Xsp+R-
>Xsp2 3*R->Ysp))"<<endl;
                        yb++;
                }

                                }


                ypin = ypin+2;
                        }

                        ypin = ypin+4;

                        for (z=14;
z<42; z++)
                        {
                                if
(z%2 == 0)
                                {
                if (exist_t[z] > 0)
                {
                        fout <<
"mkRodPin(cell \"y"<<yt<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin<<"*R->Xsp+R->Xsp2
7*R->Ysp))"<<endl;
                        yt++;
                }


                if (exist_f[z] > 0)

                {
                        fout <<
"mkRodPin(cell \"y"<<yb<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin<<"*R->Xsp+R->Xsp2
3*R->Ysp))"<<endl;
                        yb++;
                }
                                }

                else
                                {
                if (exist_t[z] > 0)

                {
                        fout <<
"mkRodPin(cell \"y"<<yt<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin+1<<"*R->Xsp+R-
>Xsp2 7*R->Ysp))"<<endl;
                        yt++;
                }


                if (exist_f[z] > 0)

                {
                        fout <<
"mkRodPin(cell \"y"<<yb<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin+1<<"*R->Xsp+R-
>Xsp2 3*R->Ysp))"<<endl;
                        yb++;
                }

                                }

                ypin = ypin+2;
                        }

                        ypin = ypin+4;

                        for (z=42;
z<70; z++)
                        {
                                if
(z%2 == 0)
                                {
                if (exist_t[z] > 0)

                {
                        fout <<
"mkRodPin(cell \"y"<<yt<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin<<"*R->Xsp+R->Xsp2
7*R->Ysp))"<<endl;
                        yt++;
                }


                if (exist_f[z] > 0)

                {
                        fout <<
"mkRodPin(cell \"y"<<yb<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin<<"*R->Xsp+R->Xsp2
3*R->Ysp))"<<endl;
                        yb++;
                }
                                }

                else
                                {
                if (exist_t[z] > 0)

                {
                        fout <<
"mkRodPin(cell \"y"<<yt<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin+1<<"*R->Xsp+R-
>Xsp2 7*R->Ysp))"<<endl;
                        yt++;
                }


                if (exist_f[z] > 0)

                {
                        fout <<
"mkRodPin(cell \"y"<<yb<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin+1<<"*R->Xsp+R-
>Xsp2 3*R->Ysp))"<<endl;
                        yb++;
                }

                                }

                ypin = ypin+2;
```

```
                }

                ypin = ypin+4;

                for (z=70;
z<77; z++)
                {
                        if
(z%2 == 0)
                        {
        if (exist_t[z] > 0)

        {

                fout <<
"mkRodPin(cell \"y"<<yt<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin<<"*R->Xsp+R->Xsp2
7*R->Ysp))"<<endl;

                yt++;

        }


        if (exist_f[z] > 0)

        {

                fout <<
"mkRodPin(cell \"y"<<yb<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin<<"*R->Xsp+R->Xsp2
3*R->Ysp))"<<endl;

                yb++;

        }
                        }

        else
                        {
        if (exist_t[z] > 0)

        {

                fout <<
"mkRodPin(cell \"y"<<yt<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin+1<<"*R->Xsp+R-
>Xsp2 7*R->Ysp))"<<endl;

                yt++;

        }


        if (exist_f[z] > 0)

        {

                fout <<
"mkRodPin(cell \"y"<<yb<<"\"
\"metal1\" \"input\" list( \"none\" )
list("<<ypin+1<<"*R->Xsp+R-
>Xsp2 3*R->Ysp))"<<endl;

                yb++;
        }
                ypin = ypin+2;
        }


                int final_right
= 0;
                int final_left =
0;

                fout << " " <<
endl;

                //internal
route
                for (int z = 0;
z < 77; z++)
                {
                        if
(exist_t[z] > 190)
                        {
        if ((exist_t[z] - 200) > z)

        {

                right =
exist_t[z]-200;

                left = z;


                if(right < 14)

                {


        final_right = right*2+2;

                }

                else if ((right
>= 14)&&(right<42))

                {


        final_right = right*2 + 6;

                }

                else if ((right
>= 42)&&(right<70))

                {


        final_right = right*2 +
10;

                }

                else

                {


        final_right = right*2 +
14;

                }


                if(left < 14)

                {


        final_left = left*2+2;

                }

                else if ((left
>= 14)&&(left<42))

                {


        final_left = left*2 + 6;

                }

                else if ((left
>= 42)&&(left<70))

                {


        final_left = left*2 + 10;

                }

                else

                {


        final_left = left*2 + 14;

                }


                if(right%2 ==
0)

                {


        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;


        fout << "list(" <<
final_right << "*R->Xsp-0.01-R-
>Mbxl2+R->Xsp2-0.033:5*R-
>Ysp " << final_right << "*R-
>Xsp-0.01+R->Mbxl2+R-
>Xsp2:6*R->Ysp+R->Mwd2 ))"
<< endl;


        fout << " " << endl;

                }
```

```
                else

                {


        final_right--;


        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;


        fout << "list(" <<
final_right << "*R->Xsp+0.01-R-
>Mbxl2+R->Xsp2:5*R->Ysp " <<
final_right << "*R->Xsp+0.01+R-
>Mbxl2+R->Xsp2+0.033:6*R-
>Ysp+R->Mwd2 ))" << endl;


        fout << " " << endl;

                }


                if(left%2 == 0)

                {


        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;


        fout << "list(" <<
final_left << "*R->Xsp+0.02-
0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 " << final_left << "*R-
>Xsp+0.02+0.16+R->Xsp2:5*R-
>Ysp+R->Abx+R->Psp+R-
>Mbx2))" << endl;


        fout << " " << endl;


                }

                else

                {


        final_left--;


        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;


        fout << "list(" <<
final_left << "*R->Xsp-0.02-
0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 " << final_left << "*R-
>Xsp-0.02+0.16+R->Xsp2:5*R-
>Ysp+R->Abx+R->Psp+R-
>Mbx2 ))" << endl;


        fout << " " << endl;

                }


                fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;


                fout << "list(" <<
final_left << "*R->Xsp-0.02-
0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 " << final_right << "*R-
>Xsp-0.01+R->Xsp2:6*R-
>Ysp+R->Mwd2 ))" << endl;


                fout << " " <<
endl;


        }


        else
        {
                right = z;

                left =
exist_t[z] -200;


                if(right < 14)

                {


        final_right = right*2+2;


                }

                else if ((right
>= 14)&&(right<42))

                {


        final_right = right*2 + 6;


                }

                else if ((right
>= 42)&&(right<70))

                {


        final_right = right*2 +
10;


                }

                else

                {


        final_right = right*2 +
14;


                }


                if(left < 14)

                {


        final_left = left*2+2;


                }

                else if ((left
>= 14)&&(left<42))

                {


        final_left = left*2 + 6;


                }

                else if ((left
>= 42)&&(left<70))

                {


        final_left = left*2 + 10;


                }

                else

                {


        final_left = left*2 + 14;


                }


                if(right%2 ==
0)

                {


        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;


        fout << "list(" <<
final_right << "*R->Xsp+0.02-
0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 " << final_right << "*R-
>Xsp+0.02+0.16+R->Xsp2:5*R-
>Ysp+R->Abx+R->Psp+R-
>Mbx2))" << endl;


        fout << " " << endl;


                }
```

```
                else

                {

        final_right--;

        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

        fout << "list(" <<
final_right << "*R->Xsp-0.02-
0.16+R->Xsp2:6*R->Ysp-R-
>Mwd2 " << final_right << "*R-
>Xsp-0.02+0.16+R->Xsp2:5*R-
>Ysp+R->Abx+R->Psp+R-
>Mbx2))" << endl;

        fout << " " << endl;

                }

                if(left%2 == 0)

                {

        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

        fout << "list(" <<
final_left << "*R->Xsp-0.01-R-
>Mbxl2+R->Xsp2-0.033:5*R-
>Ysp " << final_left << "*R->Xsp-
0.01+R->Mbxl2+R->Xsp2:6*R-
>Ysp+R->Mwd2 ))" << endl;

        fout << " " << endl;

                }

                else

                {

        final_left--;

        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

        fout << "list(" <<
final_left << "*R->Xsp+0.01-R-
>Mbxl2+R->Xsp2:5*R->Ysp " <<
final_left << "*R->Xsp+0.01+R-
>Mbxl2+R->Xsp2+0.033:6*R-
>Ysp+R->Mwd2 ))" << endl;

        fout << " " << endl;

                }

        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

        fout << "list(" <<
final_left << "*R->Xsp-0.01+R-
>Xsp2:6*R->Ysp-R->Mwd2 " <<
final_right << "*R-
>Xsp+0.02+0.16+R->Xsp2:6*R-
>Ysp+R->Mwd2 ))" << endl;

        fout << " " <<
endl;

        }

                }

        else if ((exist_t[z] > 90)
&& (exist_t[z] < 190))
                {

        if ((exist_t[z] - 100) > z)

        {

                right =
exist_t[z]-100;

                left = z;

                if(right < 14)

                {

        final_right = right*2+2;

                }

                else if ((right
>= 14)&&(right<42))

                {

        final_right = right*2 + 6;

                }

                else if ((right
>= 42)&&(right<70))

                {

        final_right = right*2 +
10;

                }

                else

                {

        final_right = right*2 +
14;

                }

                if(left < 14)

                {

        final_left = left*2+2;

                }

                else if ((left
>= 14)&&(left<42))

                {

        final_left = left*2 + 6;

                }

                else if ((left
>= 42)&&(left<70))

                {

        final_left = left*2 + 10;

                }

                else

                {

        final_left = left*2 + 14;

                }

                if(right%2 ==
0)

                {

                }

                else

                {

        final_right--;

                }

                if(left%2 == 0)

                {
```

```
                }

                else

                {

        final_left--;

                }


                fout <<
"dbCreateRect(cell list(\"poly1\"
\"drawing\")"<<endl;

                fout << "list("
<< final_left << "*R->Xsp+R-
>Xsp2:6*R->Ysp-R->Pwd2 " <<
final_right << "*R->Xsp+R-
>Xsp2:6*R->Ysp+R->Pwd2 ))" <<
endl;

                fout << " " <<
endl;


        }


        else

        {

                right = z;

                left =
exist_t[z] -100;


                if(right < 14)

                {


        final_right = right*2+2;

                }

                else if ((right
>= 14)&&(right<42))

                {


        final_right = right*2 + 6;

                }

                else if ((right
>= 42)&&(right<70))

                {


        final_right = right*2 +
10;

                }

                else

                {


        final_right = right*2 +
14;

                }


                if(left < 14)

                {


        final_left = left*2+2;

                }

                else if ((left
>= 14)&&(left<42))

                {


        final_left = left*2 + 6;

                }

                else if ((left
>= 42)&&(left<70))

                {


        final_left = left*2 + 10;

                }

                else

                {


        final_left = left*2 + 14;

                }


                if(right%2 ==
0)

                {

                }

                else

                {


        final_right--;

                }


                if(left%2 == 0)

                {

                }

                else

                {


        final_left--;

                }


                fout <<
"dbCreateRect(cell list(\"poly1\"
\"drawing\")"<<endl;

                fout << "list("
<< final_left << "*R->Xsp+R-
>Xsp2:6*R->Ysp-R->Pwd2 " <<
final_right << "*R->Xsp+R-
>Xsp2:6*R->Ysp+R->Pwd2 ))" <<
endl;

                fout << " " <<
endl;

        }
                        }
                }

                for (z = 0; z <
77; z++)
                {
                        if
(exist_f[z] > 190)
                        {

        if ((exist_f[z] - 200) > z)

        {


                right =
exist_f[z]-200;

                left = z;


                if(right < 14)

                {


        final_right = right*2+2;

                }

                else if ((right
>= 14)&&(right<42))

                {


        final_right = right*2 + 6;

                }

                else if ((right
>= 42)&&(right<70))
```

```
                    {

          final_right = right*2 +
10;

                    }

                    else

                    {

          final_right = right*2 +
14;

                    }

                    if(left < 14)

                    {

          final_left = left*2+2;

                    }

                    else if ((left
>= 14)&&(left<42))

                    {

          final_left = left*2 + 6;

                    }

                    else if ((left
>= 42)&&(left<70))

                    {

          final_left = left*2 + 10;

                    }

                    else

                    {

          final_left = left*2 + 14;

                    }

                    if(right%2 ==
0)

                    {

          fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                    fout << "list(" <<
final_right << "*R->Xsp-0.01-R-
>Mbxl2+R->Xsp2-0.033:4*R-
>Ysp-R->Mwd2 " << final_right
<< "*R->Xsp-0.01+R->Mbxl2+R-
>Xsp2:5*R->Ysp ))" << endl;

                    fout << " " << endl;

                              }

                              else

                              {

                    final_right--;

                    fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                    fout << "list(" <<
final_right << "*R->Xsp+0.01-R-
>Mbxl2+R->Xsp2:4*R->Ysp-R-
>Mwd2 " << final_right << "*R-
>Xsp+0.01+R->Mbxl2+R-
>Xsp2+0.033:5*R->Ysp ))" <<
endl;

                    fout << " " << endl;

                              }

                              if(left%2 == 0)

                              {

                    fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                    fout << "list(" <<
final_left << "*R->Xsp+0.02-
0.16+R->Xsp2:5*R->Ysp-R-
>Abx-R->Psp-R->Mbx2  " <<
final_left << "*R-
>Xsp+0.02+0.16+R->Xsp2:4*R-
>Ysp+R->Mwd2))" << endl;

                    fout << " " << endl;

                              }

                              else

                              {

                    final_left--;

                    fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                    fout << "list(" <<
final_left << "*R->Xsp-0.02-
0.16+R->Xsp2:5*R->Ysp-R-
>Abx-R->Psp-R->Mbx2 " <<
final_left << "*R->Xsp-
0.02+0.16+R->Xsp2:4*R-
>Ysp+R->Mwd2 ))" << endl;

                    fout << " " << endl;

                              }

                              fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                              fout << "list("
<< final_left << "*R->Xsp-0.02-
0.16+R->Xsp2:4*R->Ysp-R-
>Mwd2 " << final_right << "*R-
>Xsp-0.01+R->Xsp2:4*R-
>Ysp+R->Mwd2 ))" << endl;

                              fout << " " <<
endl;

                    }

                    else

                    {

                              right = z;

                              left =
exist_f[z] -200;

                              if(right < 14)

                              {

                    final_right = right*2+2;

                              }

                              else if ((right
>= 14)&&(right<42))

                              {

                    final_right = right*2 + 6;

                              }

                              else if ((right
>= 42)&&(right<70))
```

```
                {
        final_right = right*2 +
10;
                }
                else
                {
        final_right = right*2 +
14;
                }

                if(left < 14)
                {
        final_left = left*2+2;
                }
                else if ((left
>= 14)&&(left<42))
                {
        final_left = left*2 + 6;
                }
                else if ((left
>= 42)&&(left<70))
                {
        final_left = left*2 + 10;
                }
                else
                {
        final_left = left*2 + 14;
                }

                if(right%2 ==
0)
                {
        fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

        fout << "list(" <<
final_right << "*R->Xsp+0.02-
0.16+R->Xsp2:5*R->Ysp-R-
>Abx-R->Psp-R->Mbx2  " <<
final_right << "*R-
>Xsp+0.02+0.16+R->Xsp2:4*R-
>Ysp+R->Mwd2))" << endl;

                fout << " " << endl;

                        }
                        else
                        {
                final_right--;

                fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                fout << "list(" <<
final_right << "*R->Xsp-0.02-
0.16+R->Xsp2:5*R->Ysp-R-
>Abx-R->Psp-R->Mbx2 " <<
final_right << "*R->Xsp-
0.02+0.16+R->Xsp2:4*R-
>Ysp+R->Mwd2))" << endl;

                fout << " " << endl;

                        }

                        if(left%2 == 0)
                        {
                fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                fout << "list(" <<
final_left << "*R->Xsp-0.01-R-
>Mbxl2+R->Xsp2-0.033:4*R-
>Ysp-R->Mwd2 " << final_left <<
"*R->Xsp-0.01+R->Mbxl2+R-
>Xsp2:5*R->Ysp ))" << endl;

                fout << " " << endl;

                        }
                        else
                        {
                final_left--;

                fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                fout << "list(" <<
final_left << "*R->Xsp+0.01-R-
>Mbxl2+R->Xsp2:4*R->Ysp-R-
>Mwd2 " << final_left << "*R-
>Xsp+0.01+R->Mbxl2+R-
>Xsp2+0.033:5*R->Ysp ))" <<
endl;

                fout << " " << endl;

                        }

                fout <<
"dbCreateRect(cell list(\"metal1\"
\"drawing\")"<<endl;

                fout << "list("
<< final_left << "*R->Xsp-0.01+R-
>Xsp2:4*R->Ysp-R->Mwd2 " <<
final_right << "*R-
>Xsp+0.02+0.16+R->Xsp2:4*R-
>Ysp+R->Mwd2 ))" << endl;

                fout << " " <<
endl;
        }
                                }
        else if ((exist_f[z] > 90)
&& (exist_f[z] < 190))
                                {
        if ((exist_f[z] - 100) > z)
        {
                right =
exist_f[z]-100;
                left = z;

                if(right < 14)
                {
        final_right = right*2+2;
                }
                else if ((right
>= 14)&&(right<42))
                {
        final_right = right*2 + 6;
```

```
                    }

                    else if ((right
>= 42)&&(right<70))

                    {


          final_right = right*2 +
10;

                    }

                    else

                    {


          final_right = right*2 +
14;

                    }


                    if(left < 14)

                    {


          final_left = left*2+2;

                    }

                    else if ((left
>= 14)&&(left<42))

                    {


          final_left = left*2 + 6;

                    }

                    else if ((left
>= 42)&&(left<70))

                    {


          final_left = left*2 + 10;

                    }

                    else

                    {


          final_left = left*2 + 14;

                    }


                    if(right%2 ==
0)

                    {

                    }
```

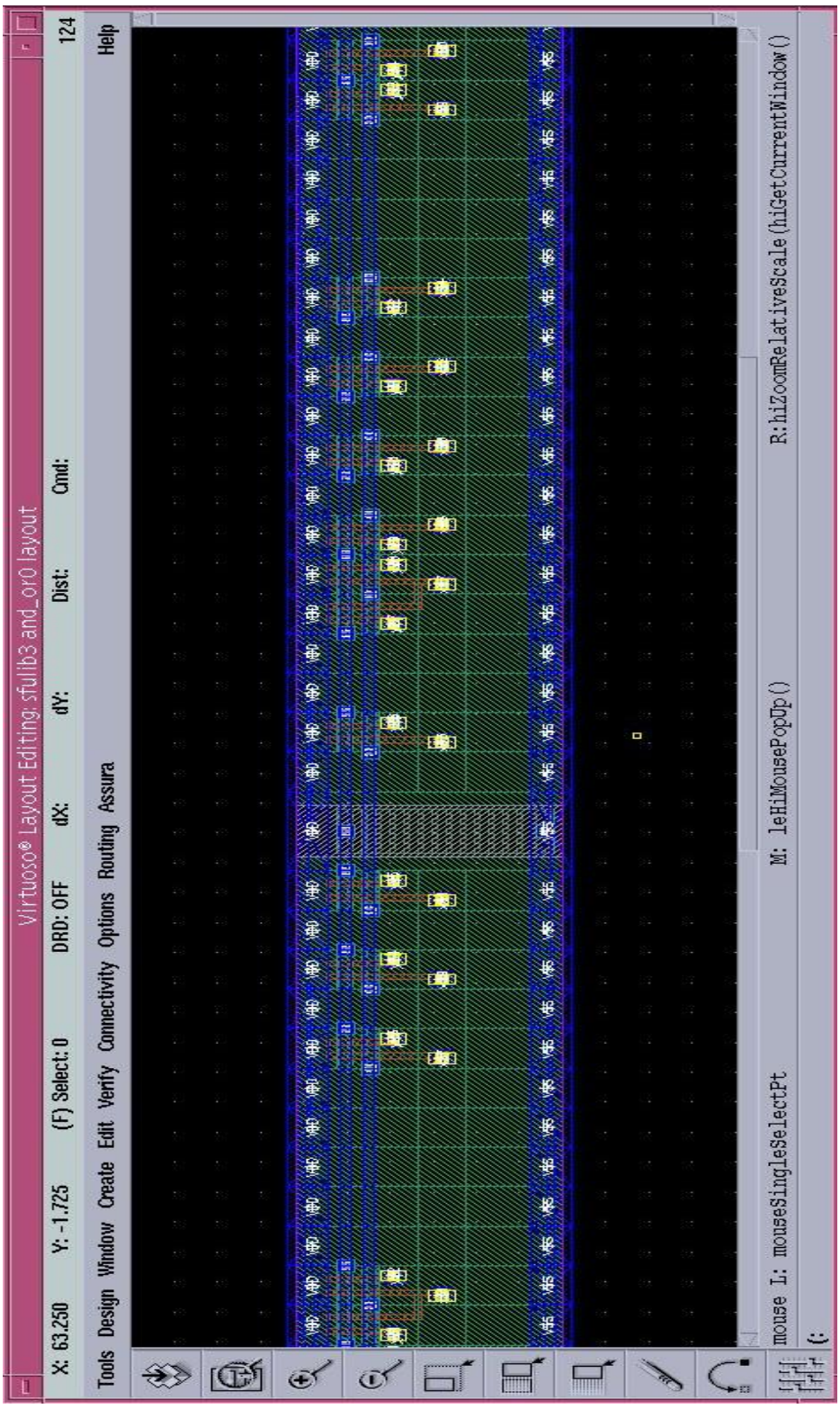
```
                    else

                    {


          final_right--;

                    }

                   if(left%2 == 0)

                    {

                    }

                    else

                    {


          final_left--;

                    }


                    fout <<
"dbCreateRect(cell list(\"poly1\"
\"drawing\")"<<endl;

                    fout << "list("
<< final_left << "*R->Xsp+R-
>Xsp2:4*R->Ysp-R->Pwd2 " <<
final_right << "*R->Xsp+R-
>Xsp2:4*R->Ysp+R->Pwd2 ))" <<
endl;

                    fout << " " <<
endl;


          }


          else

          {

                    right = z;

                    left =
exist_f[z] -100;


                    if(right < 14)

                    {


          final_right = right*2+2;

                    }

                    else if ((right
>= 14)&&(right<42))

                    {
```

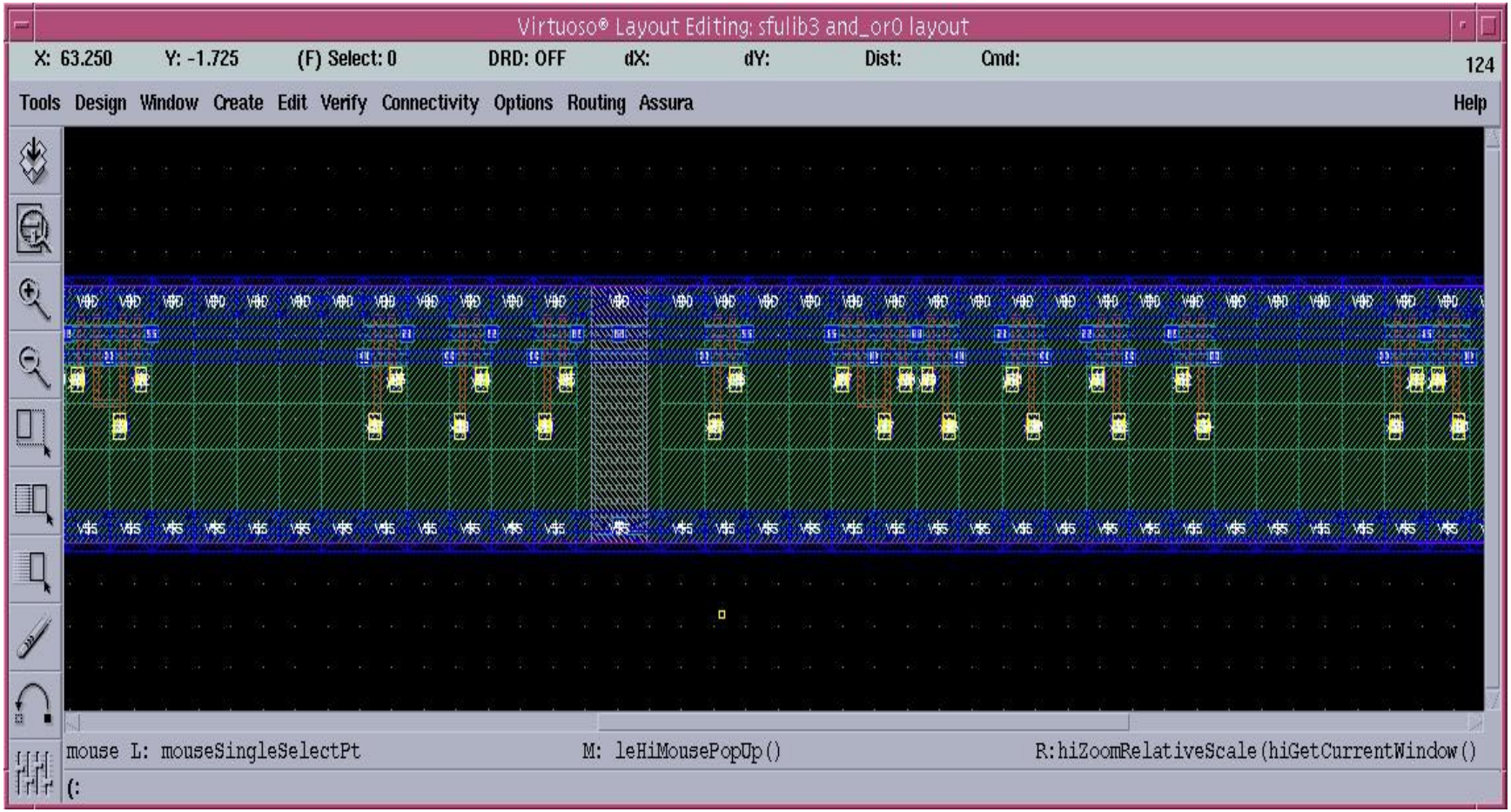
```
          final_right = right*2 + 6;

                    }

                    else if ((right
>= 42)&&(right<70))

                    {


          final_right = right*2 +
10;

                    }

                    else

                    {


          final_right = right*2 +
14;

                    }


                    if(left < 14)

                    {


          final_left = left*2+2;

                    }

                    else if ((left
>= 14)&&(left<42))

                    {


          final_left = left*2 + 6;

                    }

                    else if ((left
>= 42)&&(left<70))

                    {


          final_left = left*2 + 10;

                    }

                    else

                    {


          final_left = left*2 + 14;

                    }


                    if(right%2 ==
0)

                    {
```

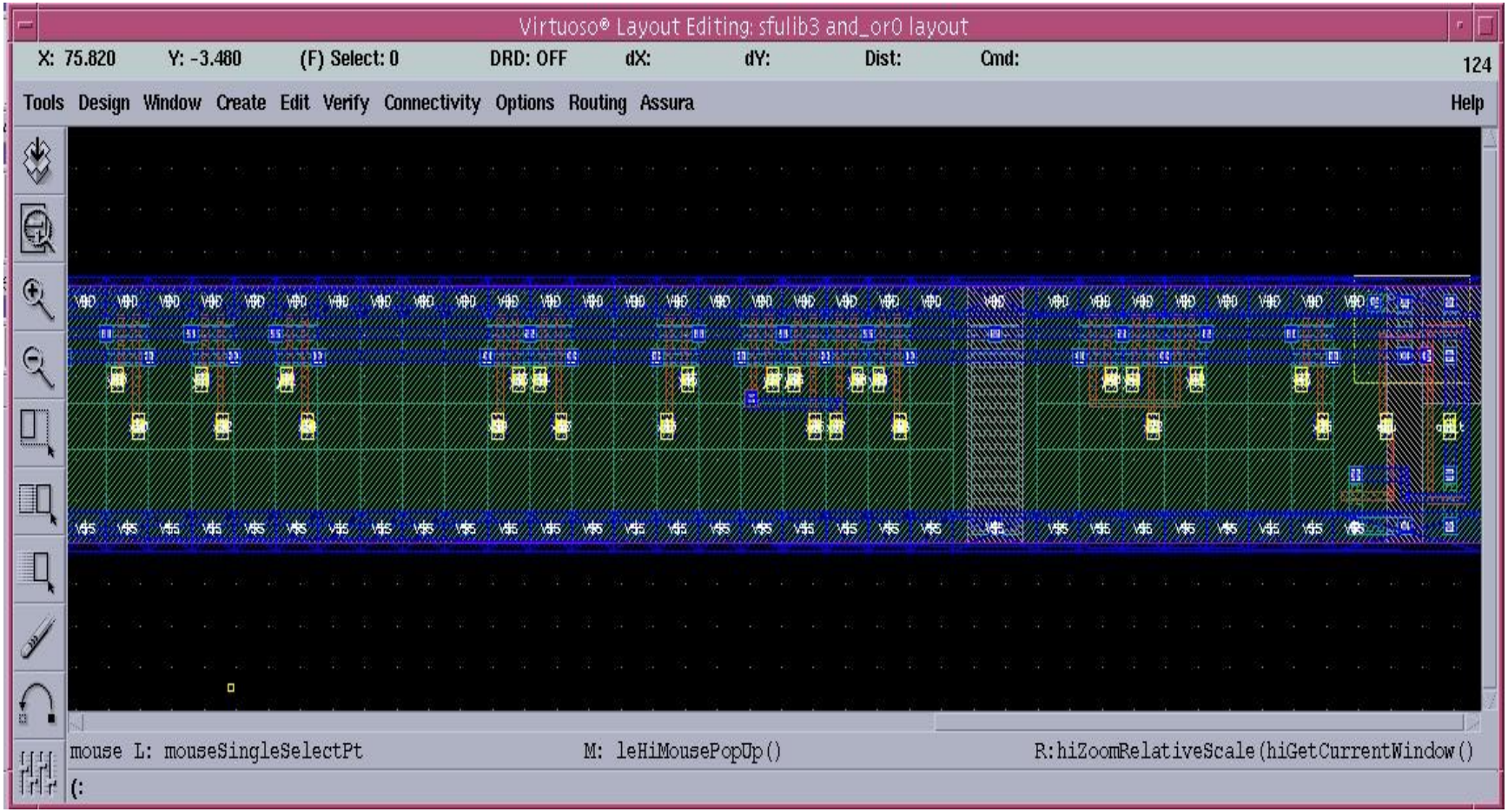
```
                }
                else
                {

        final_right--;
                }

                if(left%2 == 0)
                {
                }
                else
                {

        final_left--;
                }

                fout <<
"dbCreateRect(cell list(\"poly1\"
\"drawing\")"<<endl;
                        fout << "list("
<< final_left << "*R->Xsp+R-
>Xsp2:4*R->Ysp-R->Pwd2 " <<
final_right << "*R->Xsp+R-
>Xsp2:4*R->Ysp+R->Pwd2 ))" <<
endl;

                        fout << " " <<
endl;
                }
                        }
                }

                        fout <<
"dbCreateRect(cell
list(\"prBoundary\"
\"drawing\")"<<endl;
                        fout <<
"list(0:0  177*R->Xsp:11*R-
>Ysp ))" << endl;
                        fout << " " <<
endl;

                        fout << "
dbSave(cell)" << endl;
                        fout << "
dbClose(cell)" << endl;
                        fout << " )"
<< endl;
                return 0;
                        fout << ")" <<
endl;

                        xpin = 2;
                        ypin = 1;
                        x = 0;
                        y = 0;
                        yt = 0;
                        yb = 52;
                        sum = 0;
                        bit = 27;
                        left = 0;
                        right = 0;

                        for (h=0;
h<77; h++)
                        {
                exist_t[h] = 0;
                        }

                        for (h=0;
h<77; h++)
                        {

                exist_f[h] = 0;
                        }

                        fin.close();
                        fin2.close();
                        fout.close();
                }
}
```

Appendix 3: Layout, schematic and abstract view

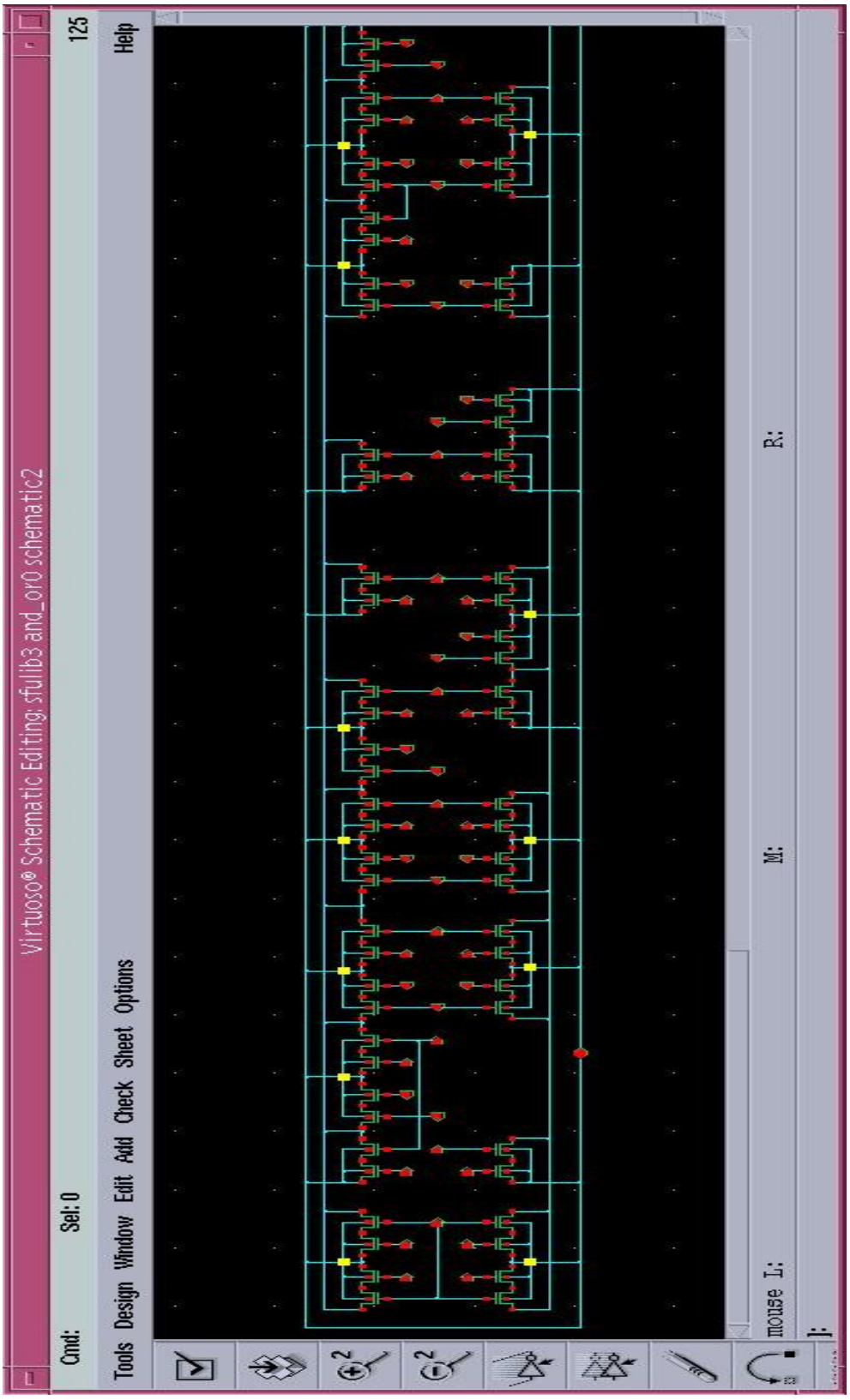
Cadence Layout for and_or0:

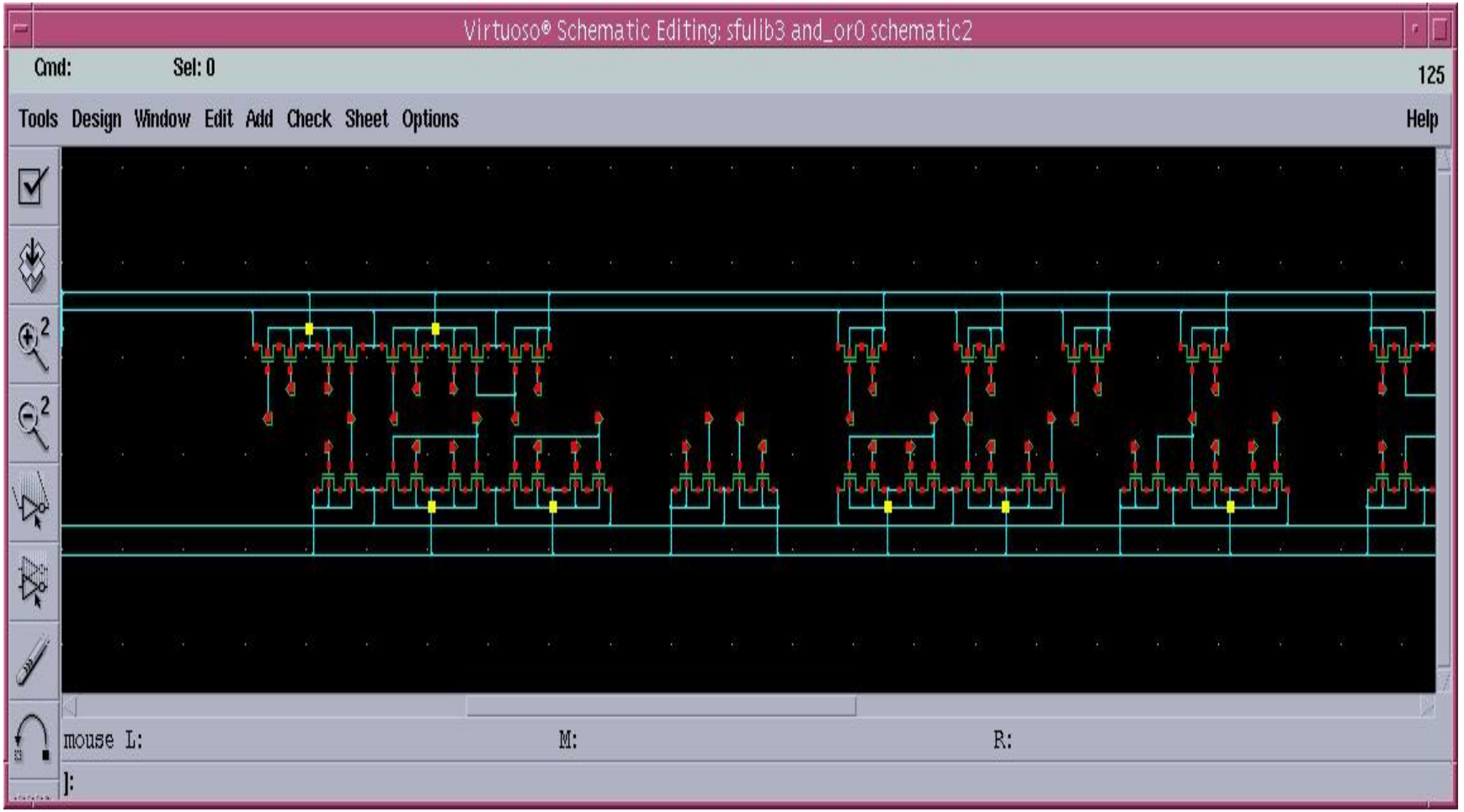


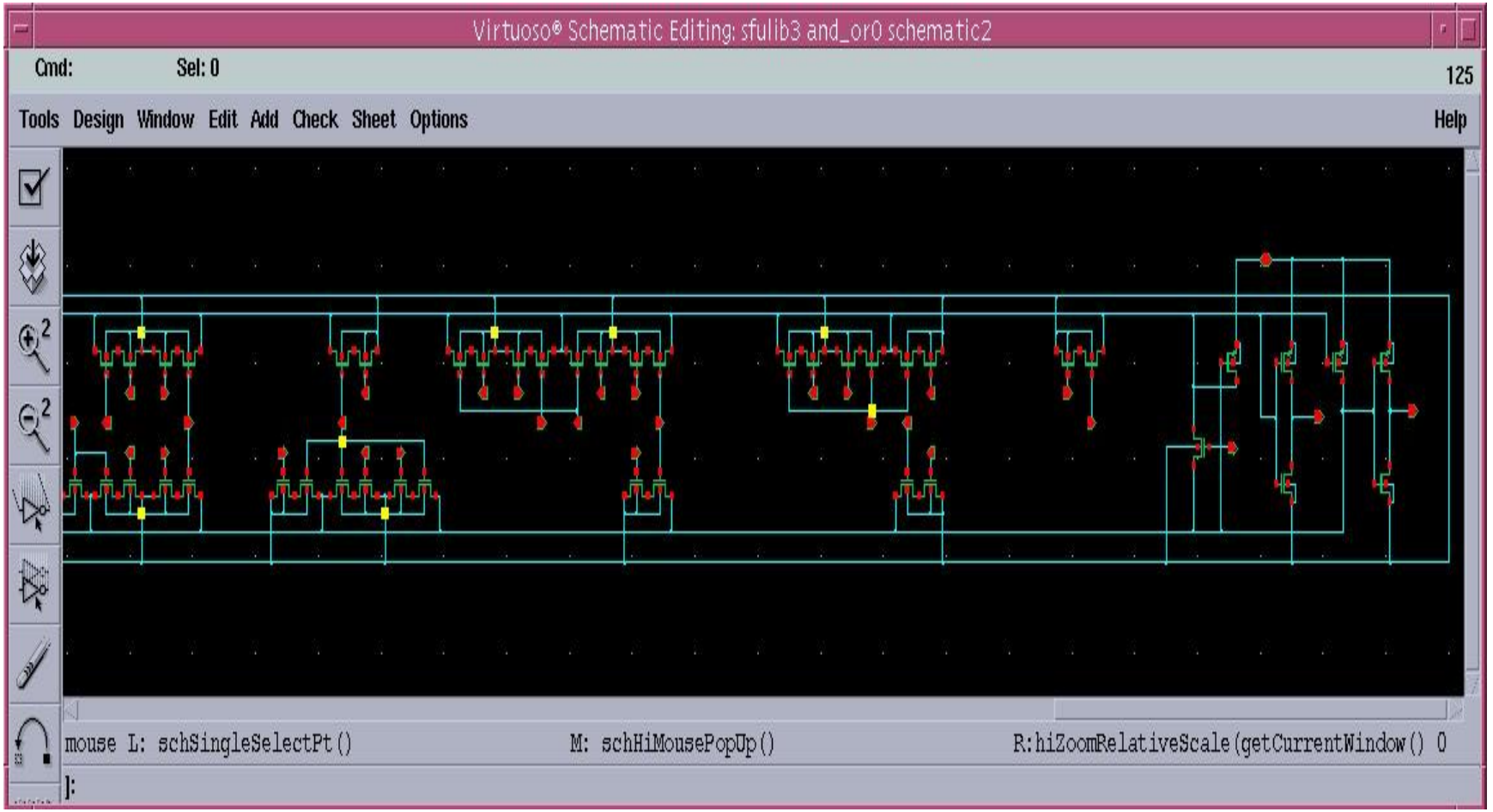

Virtuoso® Layout Editing: sfulib3 and_or0 layout
X: 63.250
Y: -1.725
(F) Select: 0
DRD: OFF
dX:
dY:
Dist:
Cmd:
124
Tools
Design
Window
Create
Edit
Verify
Connectivity
Options
Routing
Assura
Help
mouse L: mouseSingleSelectPt
M: leHiMousePopUp()
R:hiZoomRelativeScale(hiGetCurrentWindow()
(:


Virtuoso® Layout Editing: sfulib3 and_or0 layout
X: 75.820
Y: -3.480
(F) Select: 0
DRD: OFF
dX:
dY:
Dist:
Cmd:
124
Tools
Design
Window
Create
Edit
Verify
Connectivity
Options
Routing
Assura
Help
VDD
VSS
mouse L: mouseSingleSelectPt
M: leHiMousePopUp()
R:hiZoomRelativeScale(hiGetCurrentWindow()
(:

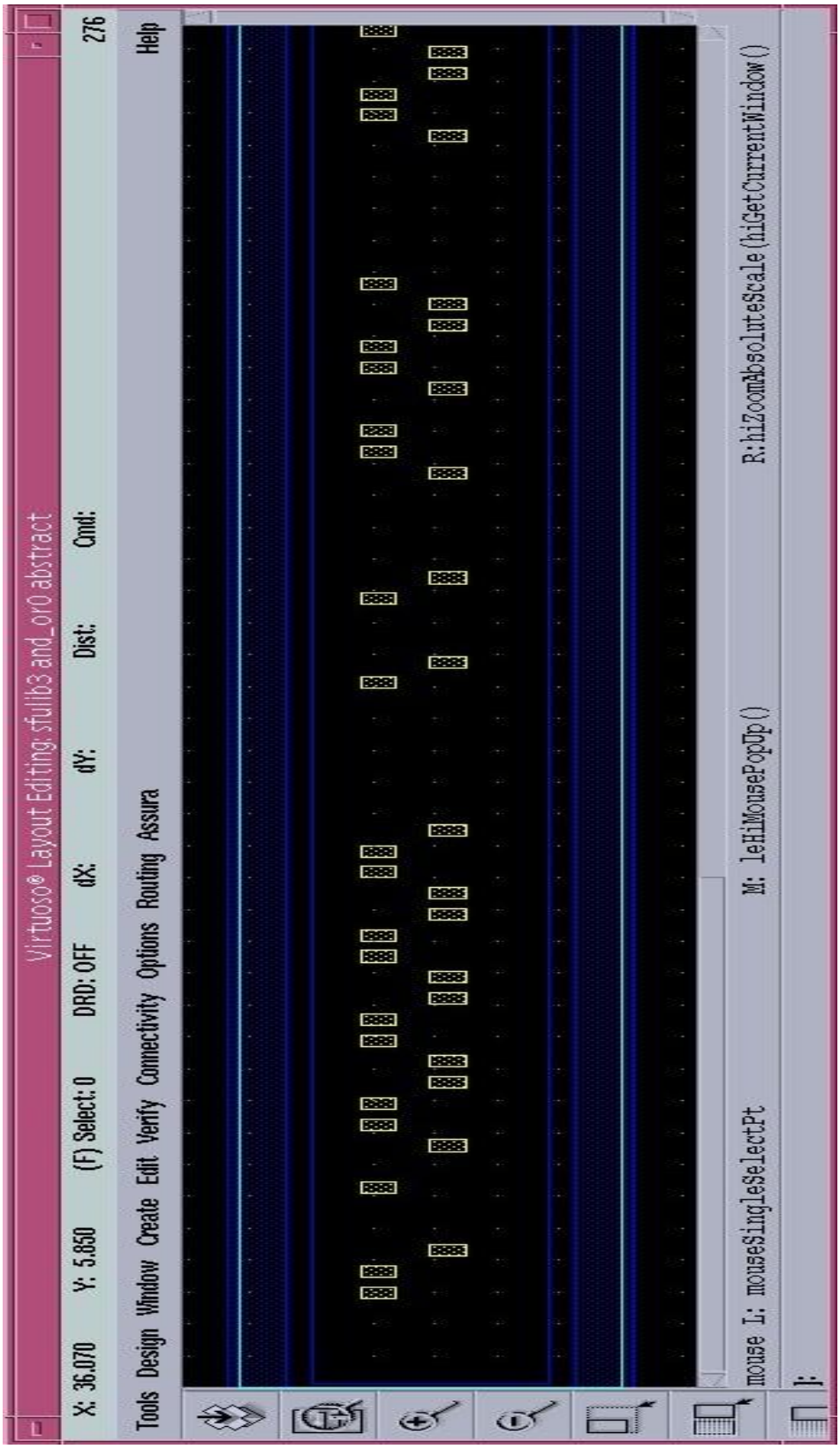
Schematic view for and_or0:




Virtuoso® Schematic Editing: sfulib3 and_or0 schematic2
Cmd:
Sel: 0
125
Tools Design Window Edit Add Check Sheet Options
Help
mouse L:
M:
R:
]:


Virtuoso® Schematic Editing: sfulib3 and_or0 schematic2
Cmd:
Sel: 0
125
Tools Design Window Edit Add Check Sheet Options
Help
mouse L: schSingleSelectPt()
M: schHiMousePopUp()
R:hiZoomRelativeScale(getCurrentWindow() 0
]:

Abstract view for and_or0:



Virtuoso® Layout Editing: sfullb3 and_or0 abstract
X: 53.855
Y: 6.635
(F) Select: 0
DRD: OFF
dX:
dY:
Dist:
Cmd:
276
Tools Design Window Create Edit Verify Connectivity Options Routing Assura
Help
mouse L: mouseSingleSelectPt
M: leHiMousePopUp()
R: hiZoomAbsoluteScale(hiGetCurrentWindow()
I:

Virtuoso® Layout Editing: sfulib3 and_or0 abstract
X: 78.020
Y: -0.955
(F) Select: 0
DRD: OFF
dX:
dY:
Dist:
Cmd:
276
Tools Design Window Create Edit Verify Connectivity Options Routing Assura
Help
mouse L: mouseSingleSelectPt
M: leHiMousePopUp()
R: hiZoomAbsoluteScale(hiGetCurrentWindow())
i:

REFERENCE LIST

1. National Institute of Standard and Technology. 25 October 1999. Data Encryption Standard (DES). *Federal Information Processing Standards Publication, 46-3*.

2. Merkle, R.C. and M.E. Hellman. July 1981. On the Security of Multiple Encryption. *Journal of Communications of the ACM*. Vol. 24:7, 465-467.

3. National Institute of Standard and Technology. 26 November 2001. Announcing the Advanced Encryption Standard (AES). *Federal Information Processing Standards Publication, 197*.

4. Dobbertin, H., L. Knudsen, and M. Robshaw, 2005. The Cryptanalysis of the AES - A Brief Survey. *Proceedings of the 4th International Conference, AES 2004*. Vol. 3373, 1-10.

5. Daemen, J. and V. Rijmen. 2002. *The Design of Rijndael: AES-the advanced encryption standard*. New York: Springer-Verlag.

6. Lenstra, A.K. 2001. Unbelievable Security Matching AES security using public key systems. *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*. Vol. 2248, 67-86.

7. Wakelin, S. 2005. VHDL Implementation of a Security Co-processor. *M.A.Sc. Thesis, Simon Fraser University*, 130.

8. Schenier, B. 1996. *Applied Cryptography*. 2nd ed. New York: John Wiley & Sons.

9. Davis, C. 2001. *IPSec: Securing VPNs.* New York: McGraw-Hill.

10. Menezes, A.J., P.C. van Oorschot, and S.A. Vanstone. 1996. *Handbook of Applied Cryptograph*y. New York: CRC Press.

11. Mali, M., F. Novak and A. Biasizzo. 2005. Hardware Implementation of AES Algorithm. *Journal of Electrical Engineering,* Vol. 56:9-10, 265-269.

12. Holden, J., L. Holden, M. Musa, E. Schaefer, and S. Wedig. September 2008. A simplified AES Algorithm. 25 May 2009 <http://www.rose-hulman.edu/~holden/Preprints/s-aes.pdf>.

13. Chouinard, J.Y. 24 September 2002. Design of Secure Computer Systems CSI4138/CEG4394 Notes on the Advanced Encryption Standard (AES). 25 May 2009. <http://www.site.uottawa.ca/~chouinar/Handout_CSI4138_AES_2002.pdf>.

14. Satoh, A., S. Morioka, K. Takano, and S. Munetoh. 2001. A Compact Rijndael Hardware Architecture with S-BOX Optimization. *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*. Vol. 2248, 239-254.

15. Järvinen , T., P. Salmela, P. Hämäläinen, and J. Takala. 4-8 September 2005. Efficient Byte Permutation Realizations for Compact AES Implementations. *Proceedings of the 13th European Signal Processing Conference (EUSIPCO 2005).*

16. HELION. AES cores. 19 April 2009. <http://www.heliontech.com/aes.htm>.

17. Moriika, S. and A. Satoh. 2002. A 10 Gbps Full-AES Crypto Design with a Twisted-BDD S-Box Architecture. *Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors,* 98-103.

18. Hodjat, A. and I. Verbauwhede. 2004. A 21.54 Gbits/s Fully Pipelined AES Processor on FPGA. *Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines,* 308-309.

19. Zhang, X. and K.K. Parhi. 2002. Implementation Approaches for the Advanced Encryption Standard Algorithm. *IEEE Circuits and Systems Magazine.* Vol. 2:4, 24-46.

20. Hobson, R. and S. Wakelin. 2005. An Area-Efficient High-Speed AES S-Box Method. *Proceedings of the Fifth International Workshop on System-on-Chip for Real-Time Applications,* 376-379.

21. Ng , P., P.T. Balsara, and D.S. June 1996. Performance of CMOS Differential Circuits. *IEEE Journal of Solid-State Circuits*. Vol. 31:6, 841-846.

22. Hodjat, A. and I. Verbauwhede. April 2006. Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/s AES Processors. *IEEE Transactions on Computers*. Vol. 55:4, 366 – 372.