Test Vehicle for a Wafer Scale Field Programmable Gate Array

by

Benoit Dufort

B.A.Sc., Université Laval, 1993

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
IN THE SCHOOL OF ENGINEERING SCIENCE

Simon Fraser University

July 1995

Approval

Name: Benoit Dufort

Degree: Master of Applied Science

Title of Thesis: **Test Vehicle for a Wafer Scale Field Programmable Gate Array**

Examining Commitee:

Dr. M. Jamal Deen, Chairman

Dr. Glenn H. Chapman, Senior Supervisor

Dr. Richard F. Hobson, Supervisor

Dr. Colombo R. Bolognesi, Examiner

Date Approved: July 13'95

PARTIAL COPYRIGHT LICENSE

Title of Thesis/Project/Extended Essay

"Test Vehicle for a Wafer Scale Field Programmable Gate Array"

Author:

(signature)

Benoit DUFORT
(name)

July 11, 95
(date)

Abstract

Field Programmable Gate Arrays are growing steadily in use and have already change the way designers build digital circuits. With their low cost and very fast turnaround time, they are especially well suited for prototyping new designs. However, the general nature of FPGAs implies a circuit density much lower than custom designs. This currently limits the size of the circuits that can be implemented on a single FPGA to 40000 equivalent gates . Boards of FPGAs are used, but their speed remains slow, because of the large capacitance of the inter-chip routing.

This thesis investigates the use of Wafer Scale Technology to expand the size of FPGAs to 3 million gates for a 200mm wafer. The defect avoidance proposed uses the laser link technology to restructure the circuit in a square array. Two different techniques, the row-column substitution and the combination of cell by cell and column substitution, are analyzed. The first one is proposed to increase the yield of small FPGAs while the second one is designed to restructure wafer scale chips. Simulations to show the effect of the restructuring on the chip yield are presented.

The proposed design is described and the defect avoidance structures explained in detail. A new kind of device, called the testable laser link, has been designed and tested. Its application in the wafer scale FPGA is presented, both in the power distribution and the reconfiguration. Two chip sized test vehicles incorporating the restructuring devices described in the thesis have been successfully fabricated and the results of different tests of cells and signal routing are analyzed. These indicate that a wafer scale FPGA would be feasible with the described techniques.

Acknowledgments

I wish to thank my senior supervisor, Dr. Glenn Chapman, for his help and guidance during this work. I also wish to thank my family and friends for their constant support during my time at Simon Fraser University.

I would like to thank the Canadian Microelectronic Corporation for fabricating the chips presented in this thesis.

This work was supported in part by the Natural Science and Engineering Research Council of Canada, the SFU Center for System Science and the British Columbia Advanced System Institute.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General

Field Programmable Gate Arrays (FPGAs) have progressed rapidly since their introduction in 1985, and are now widely employed by designers, especially as a cheap and fast means to implement new designs. An FPGA is basically an array of uncommitted programmable logic blocks that can perform different digital functions. Those blocks can be interconnected in different ways by use of a programmable routing structure. Figure 1.1 gives a block diagram of a typical FPGA. With their very low development cost and turnaround time for implementing thousands of logic gates, FPGAs provide a new capability which has changed the future of digital design. The largest FPGAs have an equivalent gate count of approximately 40,000 gates [1]. With the large amount of routing involved in an FPGA design, however, usually around 70%-90% [2], it is difficult to increase the cell count and, therefore, the design complexity of a single chip. Large FPGAs are also very expensive, mainly because of their low yields. One way to increase the gate count of a single FPGA is to use a denser technology, but still the amount of routing is an obstacle to very high gate count FPGAs. Arrays of FPGA chips on a board are used as a prototype platform [3], however the delay between the chips remains large compared to the delay within the chip.

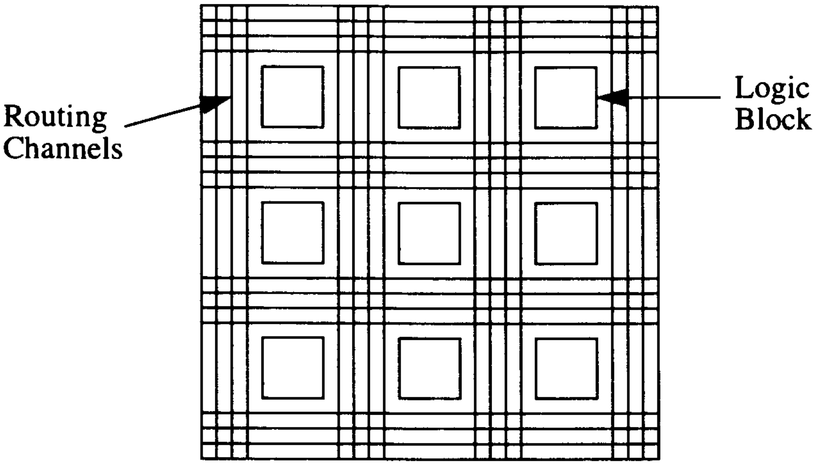


Figure 1.1 FPGA Block Diagram

While seldom considered, one way to increase the gate count of FPGAs is to employ the technique known as Wafer Scale Integration. The chip size of a standard design must be kept small in order to achieve reasonable yield, because of the defects inherent in any microelectronic fabrication process. One way to counter this problem is to use redundancy and defect avoidance. By harvesting and using only the working parts of a circuit, it is possible to increase the size of a chip, ultimately to an entire wafer.

The restructuring technique employed at Simon Fraser University (SFU) is the laser link technology, developed at MIT Lincoln Laboratory [4]. By using the power of a laser, connections can be made between two metal layers of a microelectronic process and the same laser may serve to cut lines, allowing the restructuring of the design.

This thesis investigates the use of this technique to produce FPGAs of large area and very high gate count. The idea of a wafer scale FPGA has already been proposed in another paper [5]. A different approach is proposed where the defect avoidance is invisible to the user. The focus of this thesis is to solve the interconnection and defect avoidance aspects of wafer scale systems. The FPGA cells employed are simple structures which would be replaced by more complex cells in a full system. Reasonable estimates indicate that in a final system, with a 0.5µm CMOS technology, it would be possible to implement an FPGA of approximately 1.5 million equivalent gates on a 150mm wafer, and close to 3 million on a 200mm wafer, given a yield of 75% for the cells. The same restructuring technique can also serve to build smaller FPGAs, in the order of 120 000 equivalent gates with an approximate size of 3cm x 3cm.

The restructuring can also serve to increase the yield of standard FPGAs, by providing one or two extra rows in case there is a defect, in the same way dynamic RAM chips are reconfigured today. Without increasing the gate count, this technique would be useful to reduce dramatically the cost of large FPGAs as in the case of RAM, where the number of working chips is increased by a factor of 5 with laser restructuring [12], and also provide a means to produce devices of larger areas.

FPGA designs are very well appropriated to wafer scale implementation. First, since FPGAs are arrays of identical cells, they are easier to test and reconfigure than large custom circuits; secondly, the FPGA being a reconfigurable system in itself, some of the reconfiguration circuitry is already available in the standard design and less overhead is needed to allow for reconfiguration. Finally, there is a very good potential market for large FPGAs, much better than other wafer scale projects which are very specialized.

1.2 Applications

The first application that comes to mind for a wafer scale FPGA is a prototype emulator. With their current capacities, standard devices are limited in the designs they can implement. Very large devices, such as microprocessors, require a very high gate count and therefore very complex and expensive emulators. A wafer scale FPGA would provide a cheaper and faster way to simulate those very large designs.

Another interesting application is for self healing circuits. Not only the circuit but also the testing and reconfiguration circuitry could be implemented on the same FPGA. This could prove very useful in hard to reach areas or in applications where the hardware has to be fault tolerant. FPGA is the technology of choice for a new type of computers where instead of programming instructions in a standard hardware, the hardware itself is reconfigured to suit the computing requirements. Once again, very large FPGAs would be very useful and perform better than a large number of small FPGAs.

An interesting alternative is to use the defect avoidance techniques of the large systems and apply them to moderate size FPGAs to allow much better yield. This technique is already used in all the dynamic memory chip and could greatly reduce the price of actual high-end FPGAs.

1.3 Thesis Objectives

The main objective of this thesis is to show that it is possible to apply the different techniques of Wafer Scale Integration to an FPGA design. Those techniques include power considerations, redundancy, restructuring, testing and clock distribution. A new kind of device to facilitate power testing and distribution is also presented. Different defect avoidance techniques are analyzed and simulated to find the best way to restructure FPGAs. Different types of redundancy are also analyzed.

The object of the work is not to build a complete wafer scale system, but rather to solve the problems of wafer scale on smaller dimension devices that are easier to work with and less expensive. Once the problems have been solved on the smaller devices, the increase in size should be relatively straightforward.

The work presented concentrates on designing a test vehicle to prove the concepts and apply them to a wafer scale design. There is a section describing the software requirements of a wafer scale FPGA but no extensive work has been done in this area. No attempts to optimize the logic nor the routing of FPGAs has been done. Instead, the restructuring method developed is general and can be used on different FPGA technologies and thus can be optimized by using state of the art logic and routing.

1.4 Thesis Organization

Chapter two is a theoretical review of both the Wafer Scale and the FPGA technologies. A description of the concepts essential to the understanding of large area FPGA systems is presented.

In chapter three, experiments on the laser link restructuring technique in the Mitel 1.5μm technology are presented. Work done during the early part of the master on another wafer scale test vehicle, the thermal scene simulator, are discussed, with an emphasis on the experimental work done with the chips.

Chapter four addresses the concepts of defect avoidance in FPGAs. Simulations performed to find the best restructuring method are analyzed. The design considerations involved with building a wafer scale FPGA are studied. The chapter ends with an overview of the software needed once a wafer scale system is build, both for testing of the hardware and programming of the device.

Chapter five emphasizes on the experimental work done on the test vehicle. The design is presented with each part explained in detail and the experiments on the defect avoidance methods exposed. The power distribution and the new device called the Testable Power Link are tested and their performance analyzed. The clock time delay, a critical parameter for FPGA users, is studied in detail and comparisons between HSPICE simulation and the experiments are shown. A ring oscillator was mapped on the test vehicle and its performance for different types of restructuring is presented.

The last chapter concludes by analyzing the feasibility, both technical and economical, of the Wafer Scale FPGA. A section on future work is also presented.

Chapter 2

Theory of Wafer Scale Integration and Field Programmable Gate Arrays

This chapter deals with the theory background used in conceiving a wafer scale field programmable gate array. The first section treats of the wafer scale integration technology in general. The second section deals with the theory of the FPGAS, their applications and the commercially available products.

2.1 Wafer Scale Integration

The main limitation of microelectronic fabrication is presence of production defects in the circuits. Only one defect on a chip makes it impossible to use. As the technologies get more mature, the defect density decreases but the chips must be kept relatively small to ensure sufficient yield. To build a large area chip is virtually impossible if there is no way to avoid the defects in the circuit.

The process of building large chips with the capacity to avoid defective areas is called Wafer Scale Integration [6]. The basic idea is that instead of fabricating small chips and retaining only those without defects, a very large chip can be built if there is a way to bypass the circuitry affected by defective areas. One way to do this is to use redundancy: when a defective cell is identified, a spare cell is used to replace it. The challenge is to build a circuitry to perform the reconfiguration. This circuitry must be as small as possible and have very little influence on the operation of the rest of the circuit.

So it is possible with this technique to increase significantly the size of microelectronic circuits. Because of the large amount of transistors on such a large device, the technology of choice is CMOS, due to its low power dissipation. But power still remains an important issue of wafer scale integration. The distribution of the signal throughout a very large device also becomes an issue, especially for the power rails and the clock lines. Testing of the different parts of the circuits may also become a problem and a circuit allowing the testing of hard to reach cells must be designed. Defect avoidance algorithms must be designed to make the best use of the area and maximize the speed of the circuits. Those are all aspects that the wafer scale designer must take into account.

There are two different approaches for the reconfiguration circuitry: active switches and permanent switches [6].

2.1.1 Active Switches

Active switches are basically pass transistors or transmission gates. The signals to different parts of the circuit can be rerouted by programming those switches. They have the advantage to be easily programmable and reconfigured many times. They have however many drawbacks. First, they use more space than permanent switches, especially the programming circuitry [7]; they are also more resistive, thus imposing a longer delay on the lines. Because of their large area overhead, they are also more sensitive to defects, and the switches themselves can be defective, making the circuit impossible to reconfigure.

2.1.2 Permanent Switches

Under this classification are different types of switches, such as EPROMs, EEPROMs, Laser Programmable Switches and Anti-Fuses. They all have the drawback that they are programmable only once (except EEPROMs). But they require less area and they offer much better electrical characteristics than active switching. Permanent switches are well suited for defect avoidance because once the defects are known, the circuit is reconfigured only once. But they do not allow the possibility of self healing. They are also much better candidates for the power distribution circuitry, since smaller resistances can be achieved with permanent switches.

2.1.3 Laser Link

The type of switch used here at SFU is called the Laser Link and has been developed at MIT Lincoln Laboratories in the mid-eighties as part of the Restructurable VLSI program [4]. The idea is to employ the power of a laser to make connections between two metal layers. To this effect, a special structure called the Laser Link is needed. It is basically a gateless transistor (see Figure 2.1). In unconnected form the laser link has the high impedance of two back to back diodes. A connection is formed by an Argon laser focused in the gap between the implant regions. By melting the silicon in the gap with a 2 W, 50μs laser pulse focused to 1.2 μm radius spot between the two heavily doped regions, the dopant flows across the gap, forming a low resistance connection (~100Ω) between the two metal lines. Typically two such "zap" points are made per link.

The main advantage of this type of structure is that it can be implemented in standard CMOS technology since it does not require any additional steps or materials. Of course it requires the use of a laser table that can be precisely aligned to allow the laser spot to be focused between the active regions.

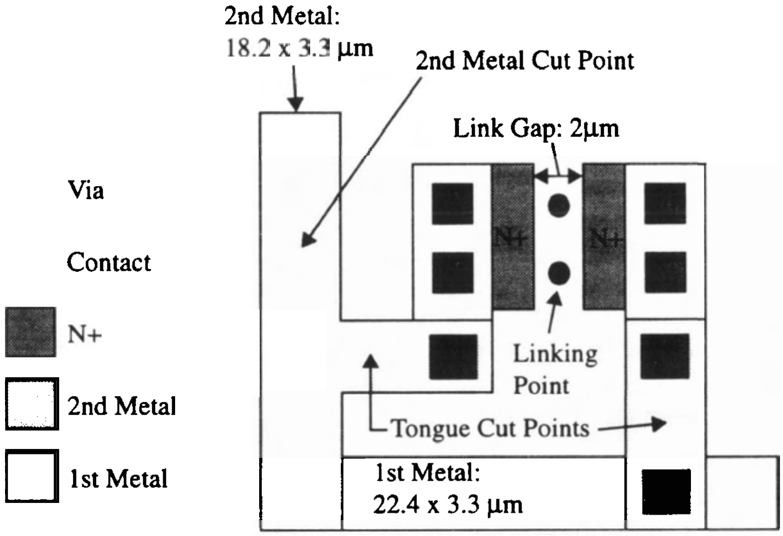


Figure 2.1 Mitel 1.5µm CMOS Laser Link

To successfully reconfigure a design, cuts are made to disconnect certain lines in the circuit. This is done by shining the laser on top of the metal line and melting it. To start a design, it is necessary to know the different parameters such as laser power and pulse duration in order to make a suitable connection in a given technology. The next chapter explains the experimental procedure used to extract those parameters for the Mitel 1.5µm CMOS technology and gives an example of a wafer scale circuit experiment done here at SFU.

2.2 Field Programmable Gate Arrays

With current technology, it is possible to build large custom designs at relatively low cost. However, because of the extensive manufacturing effort, the cost is high for each unit unless large volumes are produced. So it becomes really hard and expensive to build a prototype. Field Programmable Gate Arrays have emerged as the ultimate solution for low cost and fast turnaround prototyping. An FPGA based prototype can be manufactured in only minutes and their cost is in the order of $100 for low gate counts [2]. This is the reason why FPGAs have evolved so rapidly from a tiny market four years ago to a very large business today. It is predicted that almost 1 billion dollars worth of FPGAs will be sold each year by 1996 [2].

2.2.1 What is an FPGA?

The Field Programmable Gate Array is basically an array of elements capable of performing logic functions that can be interconnected in a general way. Both the logic functions and the interconnections are user programmable. A general FPGA is composed of three parts, as seen in Figure 2.2.

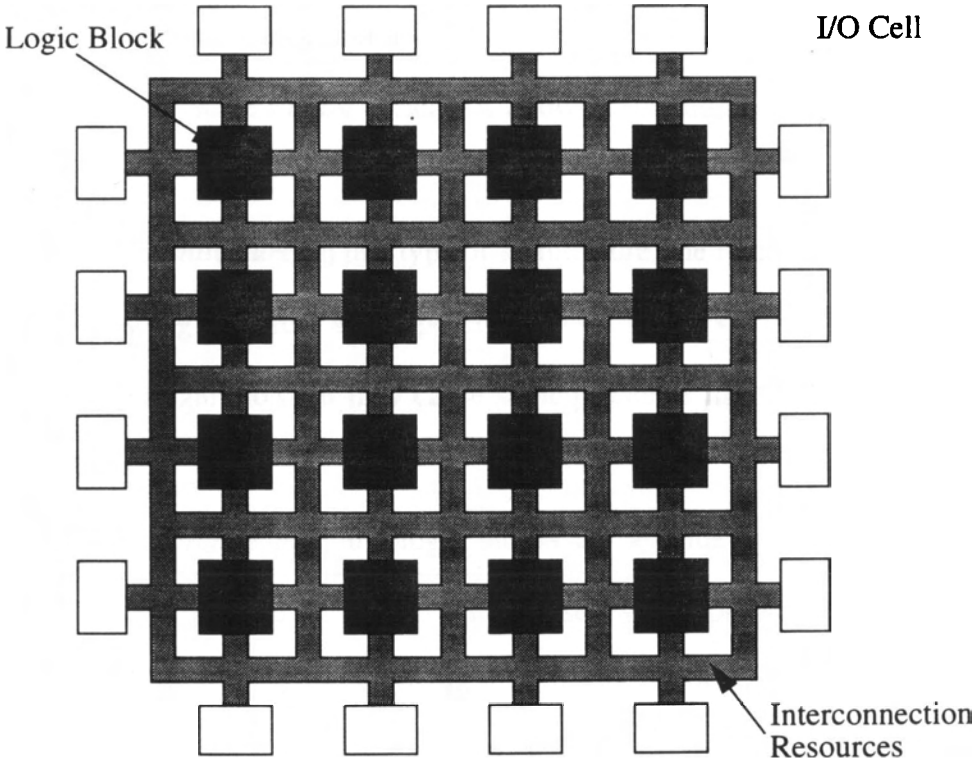


Figure 2.2 Conceptual Simple FPGA

The Logic Block contains the logic to implement different functions. It can be as simple as a two-input nand gate or be quite complicated, such as look-up tables and flip-flops. The interconnection resources are composed of wire segments and programmable switches that allow the signals to propagate between the logic blocks and to go outside the chips via the I/O Cells. These cells are usually composed of multiplexers and buffers to connect the pads to the wire segments. There are several ways to program the logic functions and the switches to route the signal, including: RAM cells controlling pass transistors, anti-fuses, EPROM and EEPROM transistors.

2.2.2 FPGA Architectures

In this section, the different architectures used in FPGA design are presented, with some comments to their applicability to wafer scale designs.

Symmetrical architecture: this is the most commonly used, where the logic blocks are surrounded by vertical and horizontal channels of routing. This is a very good architecture for wafer scale FPGA because it allows bypassing of single cells or entire rows.

Row based architecture: in this type of architecture, the logic blocks are organized in rows and the routing resources are disposed between the rows. This architecture is well suited for row reconfiguration but may cause some problems in reconfiguring very large designs.

Sea of gates architecture: the logic blocks are all side by side and the routing resources are placed on top of them. This causes some problems in most of the reconfiguration techniques and thus this architecture is not well suited for wafer scale applications.

Hierarchical PLDs architecture: this is an architecture where instead of having a large number of simple logic blocks, there is a small number of programmable logic devices(PLDs), which are composed of different logic blocks. This could be an interesting architecture to explore for wafer scale integration: for example, memory cells consume many gates in some designs in a simple FPGA. Significant gains could be obtained by placing blocks of memory throughout the system. It is simpler however to have a repetition of the same cell for the reconfiguration.

2.2.3 FPGA Applications

FPGAs can be used in all applications that can be performed now by other sorts of programmable logic devices. Their ability to be reconfigured on site also gives rise to new technologies. Here are some examples of FPGA applications:

Application-Specific Integrated Circuits (ASICs); being a completely general medium for digital logic implementation, FPGAs are particularly well suited for the design of ASICs. Some examples include controllers, graphics engines and many telecommunication applications.

Random logic implementation; since the FPGAs have a higher density than PALs (Programmable Array Logic), they are a good choice for implementing random logic in circuits where speed is not critical. One FPGA can replace ten to twenty PALs and perform the same function. FPGAs can also replace advantageously many SSI chips that require a lot of area on circuit boards, for "glue" logic.

Prototyping; FPGAs are almost ideal for prototyping applications. Their low cost and the extremely fast turnaround time they offer give them tremendous advantages over traditional prototyping methods. This is an area where a very large FPGA would be very useful, since the more gate equivalent an FPGA can offer, the larger the circuit it can implement.

FPGA-based Compute engines; this is an all new class of computers where instead of fetching instructions in a known hardware, it is the hardware itself which is actually reconfigured to perform the task. This increases the performance in the order of 100 times. Presently, boards of FPGAs are used for those kinds of computers; Wafer Scale FPGAs would increase the performance and capacity of such devices.

On site reconfiguration of hardware; this is particularly useful for applications that may require hardware reconfiguration and repair in hard to reach locations, such as satellites. Once again, many FPGAs could be replaced by a wafer scale design.

2.2.4 Implementation Process

In order to successfully implement a circuit on an FPGA, an efficient CAD system musı be used; this system must be able to perform the tasks shown in Figure 2.3.

The first step is to enter the design. This can be done by any schematic design tool, VHDL description or any acceptable format for the CAD tool. Then, the FPGA CAD tools have to perform the logic optimization, consisting of modifying the logic expressions either for speed or area density. The next step is to perform the technology mapping: it consists of dividing the circuit into logic functions that can be realized by the logic block of the FPGA used; for example, if the logic block used is a two input nand gate, the whole circuit has to be transformed into nand gates. Once again there are two ways to do this: either the mapper can optimize the number of logic blocks used or optimize the circuit for speed and use more logic blocks. The next step is Placement, where the logic blocks are placed to minimize the interconnection delays. Finally, the Routing, which assigns the wire segments and switches to connect the logic blocks together. The two final steps of the CAD tool may be iterative and it can be necessary to redo the placement if the router is unable to successfully route all the connections. These steps can also be repeated to optimize the design for speed.

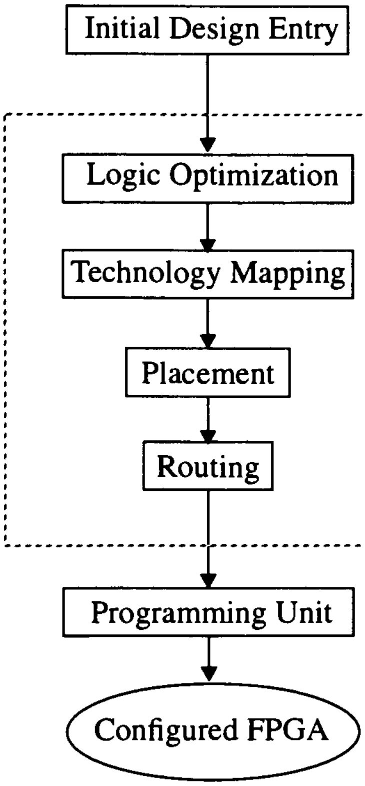


Figure 2.3 FPGA Implementation Process

The last step in the implementation process is the Programming of the FPGA. It depends on the programming technology of the FPGA used. For a RAM programmable FPGA, only a bit pattern fetched out of a separate memory is sufficient. For other technologies, such as anti fuses or EPROMs, an appropriate programming unit must be used.

2.2.5 Commercially Available FPGAs

Several combinations of architecture, logic block type and programming technologies are available on the market [2].

The most important is the Xilinx FPGA. The latest generation of Xilinx FPGAs uses a RAM programmable symmetrical architecture with look-up table based logic blocks.

Actel offers a row-based design with anti-fuse programming and a multiplexer based logic block. Compared to Xilinx FPGAs, the Actel design has a smaller logic block.

Altera uses the hierarchical approach with EPROM programming while Plessey offers sea-of-nand-gates static RAM programmable FPGAs.

There are other companies that offer different types and technologies. The choice of an FPGA depends on the particular application and the speed needed. The CAD tools available should also be taken into account when choosing a type of FPGA to use. Each company offers its own software but a specific software must be used for each type of FPGA and the user can not make a separate choice between the hardware and the programming tool.

2.3 Summary

The present chapter described the basics of Wafer Scale Integration and FPGAs. These are two very wide fields but only succinct information necessary to the understanding of the next chapters has been presented. The section on Wafer Scale Integration described the types of switches used and presented the type used here, the laser link. The section on FPGAs furnished explanations of the different architectures and presented designs that are commercially available.

Chapter 3

Laser Linking Wafer Scale Integration

This chapter presents experiments done to extract parameters for laser link devices using the Mitel technology as a wafer scale medium; the knowledge of those parameters is crucial before any design work can be undertaken. It furnishes also explanations of the experiments done on laser linking with the thermal pixel scene simulator, a wafer scale test vehicle developed here at SFU.

3.1 Mitel 1.5μm Technology Parameter Extraction

3.1.1 Laser Table Setup

To make the laser links and cuts, a special setup is needed. The first part of this setup is the laser. The laser used here at SFU is a 5.0 W Argon laser. Because of the very small dimensions of today's microelectronic structures, a very precise table is needed to correctly aligned the structure to be processed with the laser. The table uses laser interferometry to allow a 0.1 μm precision in both the horizontal and vertical axes. In order to correctly melt the silicon, a short duration (approximately 100μs) laser pulse is needed. This is achieved by passing the laser beam through an electro-optic shutter. A z-axis micropositioner is also used for the remote focusing of the chip. All the equipment is controlled via a Windows based software developed here at SFU. This software can be used to zap single points or a script file can be used to do batch work. A photograph of the laser table setup is shown in Figure 3.1.

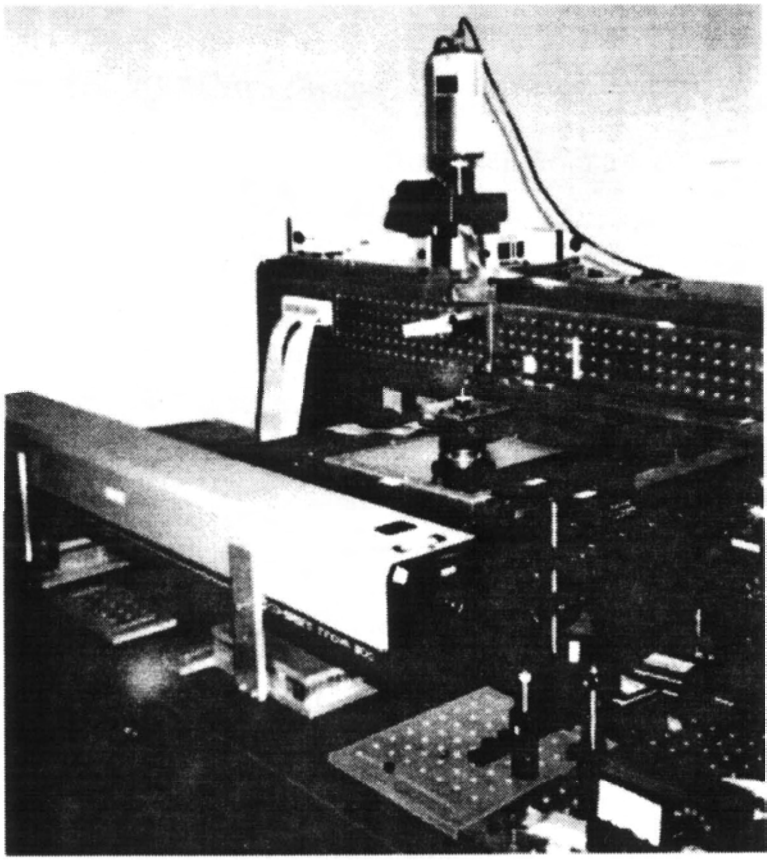


Figure 3.1 Laser Table Setup

3.1.2 Laser Link Power Calculations

Figure 3.2 shows a cross section view of the laser link in the Mitel technology. This is in fact a simplified model used to calculate the power required to form the melt pool.

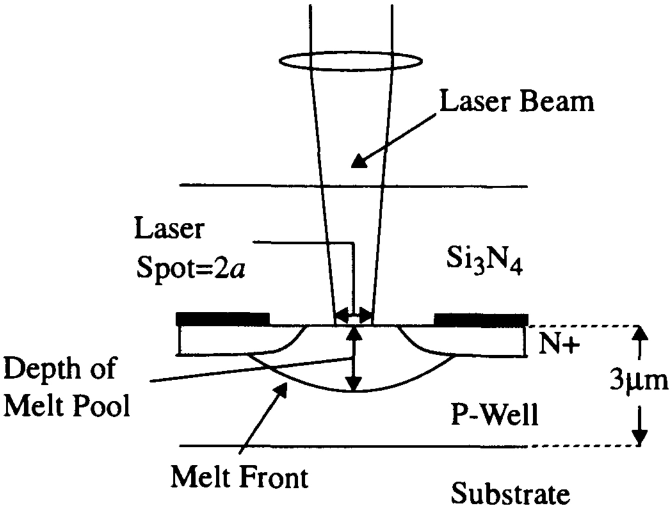


Figure 3.2 Cross Section of the Linking Process

First of all, the vertical temperature distribution from a focused laser spot can be approximated by using the formula ([8] page 171):

$$\Delta T = \frac{2H}{k}\sqrt{\alpha_T t}\left[ierf\left(\frac{z}{2\sqrt{\alpha_T t}}\right) - ierf\left(\frac{\sqrt{z^2+a^2}}{2\sqrt{\alpha_T t}}\right)\right] \qquad (3.1)$$

Where H is the power density, t the time of the pulse, z the depth of penetration, α_T the thermal diffusivity and a the radius of the pulse. This formula assumes a constant α_T with temperature, which is not really true, but a useful first approximation.

For silicon ([8] page 174):

$$\alpha_T = 103 \cdot 10^{-6}\frac{m^2}{s}$$

$$k = 170\frac{W}{m \cdot K}$$

$$T_m = 1680K$$

This formula also assumes the light is absorbed at the surface, which is a good approximation as the green (514nm) Argon light is absorbed by a depth of about 0.3μm. In order to calculate the power needed from the laser, the reflectivity at the silicon-silicon nitride interface must be calculated, by using:

$$R = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2 \tag{3.2}$$

n_{Si}=4.2 and n_{Si3N4}=2 [9], then R=0.126. The effect of the oxide between the silicon nitride and the silicon is neglected in the calculation. The reflection coefficient of the air-Si interface is also needed. This coefficient is R=0.11. The power absorbed by the Si_3N_4 is given by the Beer-Lambert law ([10] page 165):

$$P(z) = P_0 e^{-2\alpha_l z} \tag{3.3}$$

Where α_l is the light absorption coefficient for silicon nitride. For argon laser light (514nm), α_l=300 m^{-1} [11]. Thus the power density at the surface of the silicon is approximately:

$$H = \frac{P_0 e^{-2\alpha_l z}}{\pi a^2} \tag{3.4}$$

where P_0=Laser power.

Instead of calculating one value only for a specific depth, a graph of the power required from the laser pulse in function of the depth of the melt pool is given in Figure 3.3. This is obtained by solving (3.1) for ΔT=1680K, the silicon melting point, for laser power densities from (3.4) using the typical values for this research of:

$$z_{Si3N4} = 3\mu m \qquad t_{pulse} = 100\mu s \qquad a_{spot} = 0.5\mu m$$

The thickness of the passivation layer is an approximation since the real value is unknown.

The graph shows the power needed is in the order of 2.0 W to create a melt pool sufficient to allow the dopants to form a bridge between the two N+ regions, knowing the distance between the active region is 2µm and assuming the melt front propagates with the same velocity in the vertical and horizontal directions. Experiments show this assumption is reasonable. The graph also tells that for a power of about 5.8 W, the melt front reaches the substrate below the P-well. Such a high power must be avoided because a connection to the substrate results in a non usable link.

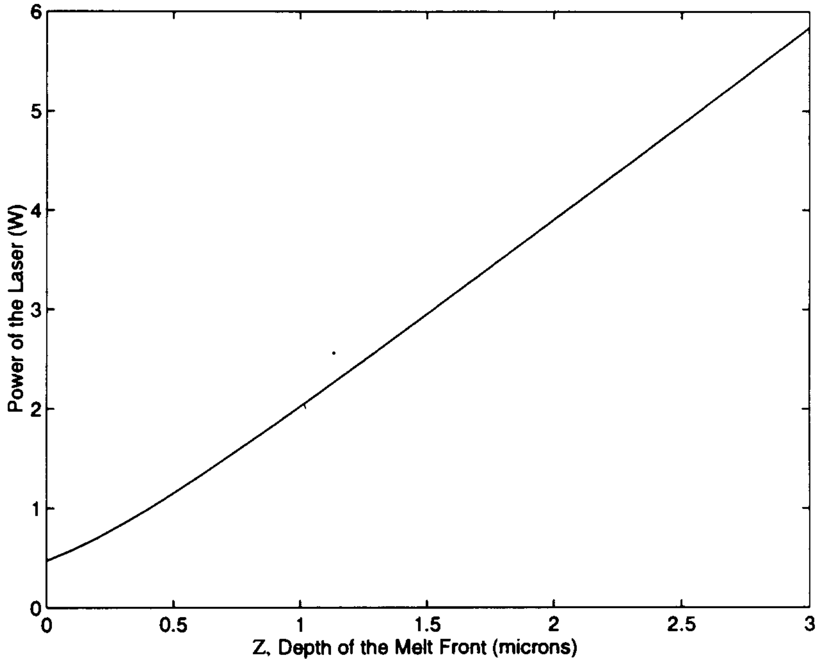


Figure 3.3 Graph of the Power vs. Depth of the Melt Front

3.1.3 Laser Power Experiments

The first set of experiment was done to find the power required to have a low resistance connection between the active regions. Table 3.1 gives the obtained results. The time of the laser pulse is 100 µs; the resistance of five separate connections was averaged.

After 2.5 W, the resistance starts to saturate. While the resistance is lower at 2.75W, the damage is greater and can break the vias of the laser link. The safest power to use is 2.50W. If the link is long enough, the laser can be zapped at two separate points to reduce the resistance of the connection.

Power(W)	R (Ω)	σ
2.00	260.3	20.9
2.25	183.1	15.7
2.50	149.2	11.2
2.75	134.1	6.1

Table 3.1: Resistance of one Zap (Mitel 1.5μm link)

If the second zap is too close to the first one, or if the second zap is at the same location, there is no decrease in resistance. Experiment shows the spots should be at least 6μm apart to have a significant decrease in resistance. At 2.50 W, a second zap decreased the resistance of the link to 109±2Ω . The experiments show the resistance of two zaps follows a curve vs. the power similar to the curve for the resistance of one zap.

3.1.4 Laser Link Experiment

The next experiment consists in doing many links and test the resistance of each of them. The zapping pattern can be seen in Figure 3.4. The width of the active regions is 9.9 μm and the gap between is 2μm (the minimum allowed separation in Mitel 1.5μm technology). The first zap is made 1.65 μm from the top and the second at 6.6 μm from the first zap. The laser power is 2.50W and the pulse duration is 100μs.

A third spot in the middle does not decrease the resistance and is thus useless. This is because the two melt pools created by the zaps are touching and no gain is made by adding an extra zap. The results for 10 links give $R_{average} = 109 \pm 5\Omega$.

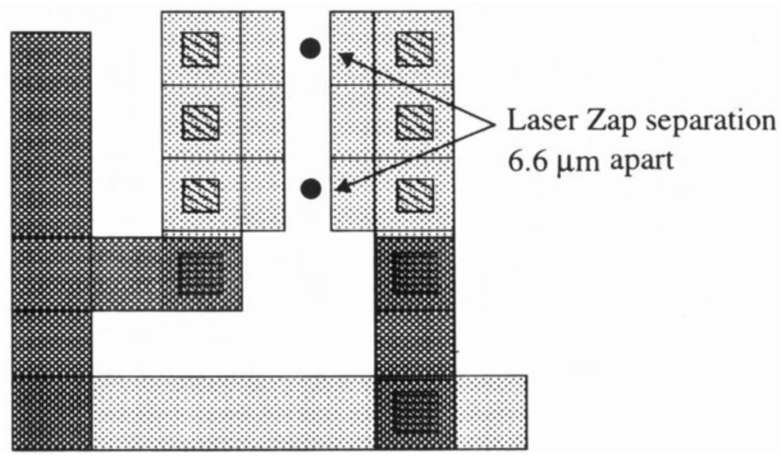


Figure 3.4 Position of the Laser Zaps

A rough rule of thumb to estimate the resistance of links can be deduced from experimental data: one zap produces a resistance of 100Ω; a 50Ω constant resistance from the contact cuts to the N+ region and the implant region is added to the zap resistances. A two-zap link will have 50Ω + (100Ω || 100Ω) giving a total of 100Ω. A three-zap link has a resistance of about 83Ω . This shows the third zap, which has little measurable effect, is not really creating an additional parallel resistive path. Figure 3.5 is a photograph of two laser links, the one on the left has been linked with the method explained above. The laser cut on a second metal line can also be seen.

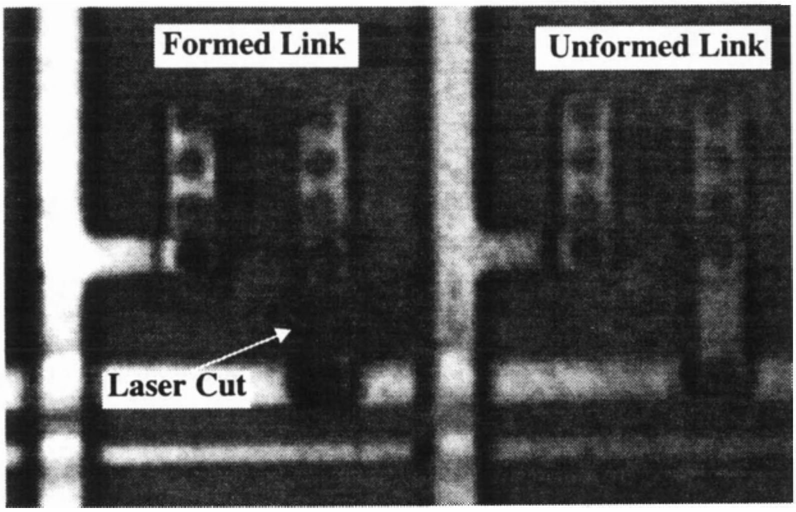
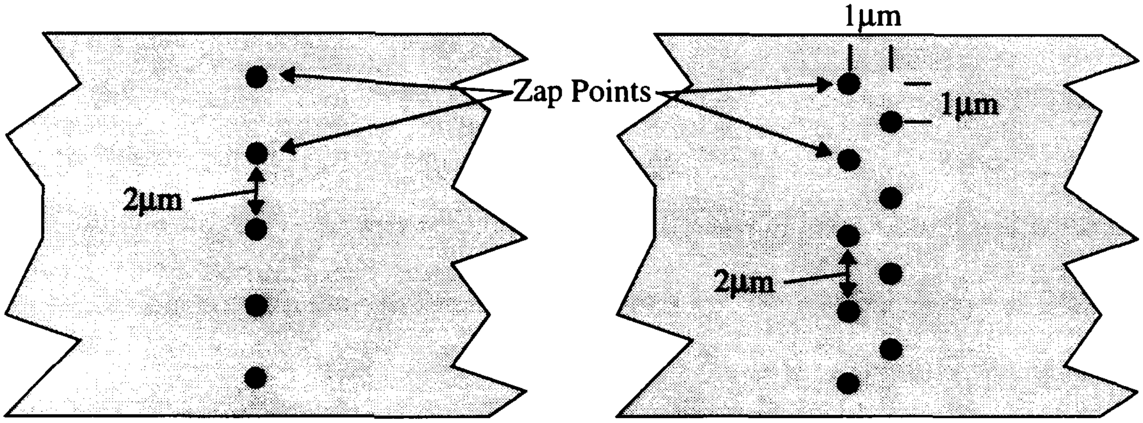


Figure 3.5 Photograph of the Links

3.1.5 Laser Cut Experiments

The power required to cut the aluminum lines must be found. Experiments show there is no problem in cutting the 3.3μm width lines with a laser power higher than 2.5W, by zapping the line in the middle with a pulse of 100μs and a spot size of 1.2μm FWHM (Full Width Half Maximum). Out of a total of approximately 100 cuts made this way, all of them showed a resistance higher than 10 MΩ. So the best way seems to use the same parameters for the cuts and the links. This cutting behavior applies to metal1 and metal2 lines. In order to cut wider lines, such as power lines, a larger number of zaps is needed. The effect of each zap is reduced because of the greater loss of energy due to heat flow. Due to the lack of proper test structures, it was hard to evaluate if the large line was really cut, but by visual inspection, two cutting patterns were developed. They are shown in Figure 3.6 for a 10μm wide metal1 line.



Straight Line Cutting

Zig-Zag Cutting

Figure 3.6 Two Methods for Cutting Large Metal Lines

The straight line method consists of a first zap at 1 μm from the edge of the line and a zap each 2μm afterwards, until a distance of less than 1 μm from the other edge is reached. In the Zig-Zag method, a first zap is made 1μm from the edge, then 1μm away in each direction. The Zig-Zag method seems more reliable (electrical test should be performed to confirm this) than the straight line method but takes a longer time due to the higher number of pulses required.

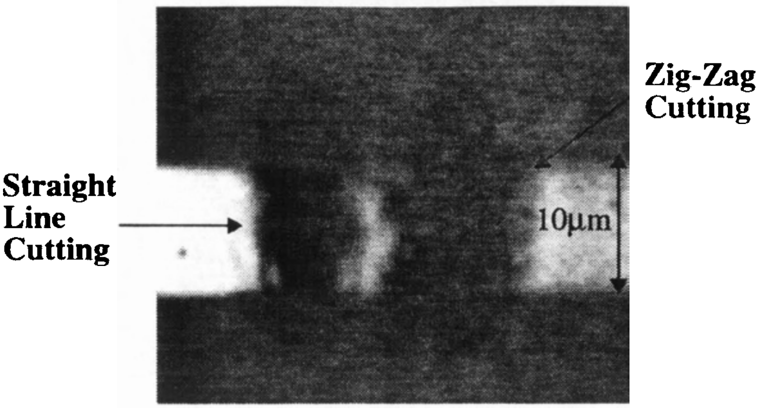


Figure 3.7 Photograph of the Cutting Methods

The time taken to cut a 10μm wide line is 4.27s for the straight line method and 13.45s for the Zig-Zag. The Zig-Zag method also takes up more space. The choice should be made in function of the time and the area available. Figure 3.7 is a photograph of the two methods; the larger area taken by the zig-zag method is clearly seen.

3.1.6 Damage to Silicon Nitride

Silicon nitride can be very sensitive to low intensities of laser light. It will fluoresce at about 10mW of power for a 1.2μm spot, and it has a low damage threshold that depends on the exact composition of the nitride. In the photographs, lots of damage surrounding the links and cuts can be seen. This is due to the behavior of the silicon nitride under the laser pulse. Such damage is not seen in the Northern Telecom 3μm process which uses glass instead of nitride. Figure 3.8 shows the radius of damage for different laser powers. The large damage at 4.5W is probably due to a defect present in the silicon nitride layer. Its diameter is close to 10μm. These defects can have a significant impact because the layer seems much more opaque when shot once with the laser and there is no or little effect when a pulse is applied on top of a damaged area. These damages increase the minimum spacing allowable between links and lines to cut.

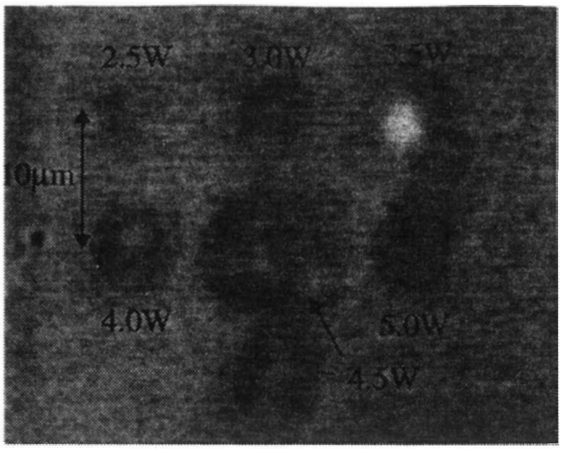


Figure 3.8 Photograph of the Damage in the Silicon Nitride

3.1.7 Batch Linking and Cutting

In this section the behavior of many links and cuts done in parallel with a script file is discussed. First the chip has to be very well aligned, especially if the links or cuts are far apart on the table. Due to the 'wiggling' (side motion) effects of the z axis motor, the focus was not changed during the linking process. The results were as follows: for five links in parallel, the resistance was 41.2±0.4 Ω and the total time to do the connections was 10.58s. This gives an average link resistance of 206Ω. The individual links have a measured resistance of about 100Ω with this setup. This means the links have a slightly higher resistance when done in batch. An explanation is that the alignment is not as precise as when the links are done individually, therefore some of them, especially the last ones, show a higher resistance. The average time to do each link is 2.12 s, which is rather long. Faster control systems of the table will be needed for a very large number of links.

For the cuts, it took 3.7s to do 5 cuts of 3.3μm wide lines in a row. The resistance was high, 26MΩ, proving the cutting was successful. The average time for each cut was 0.74s. The alignment seems less critical with the cuts. A different focus is used for cuts and links. The difference is about 3μm. To do batch linking and cutting with the same file, or over a large area, the focus has to be adjusted and therefore a Z axis controller which is very precise and stable is needed.

3.1.8 Linking Summary

The goal of these experiments was to extract the parameters needed to use the laser linking and cutting with the Mitel 1.5μm technology. Results have shown that the Mitel technology can be used efficiently for this purpose. With proper table settings and careful alignment, links resistances in the order of 100Ω can be achieved. The cutting of thin lines (less than 3.3μm) seems very reliable, more than with other technologies used before, like the 3μm CMOS from Northern Telecom. The cutting of large lines seems to be efficient, but in this experiment only visual inspection was used because of the lack of test structures.

The links seem to be less reproducible and their resistance is influenced by the parameters. The minimum width of the links seems to be around 10μm if a resistance in the order of 100Ω is wanted. This is because the distance between two zaps must be at least ~6μm to be effective. If a second zap is too close to the first one, there is no effect. A careful alignment of the electro-optic shutter is needed to achieve maximum throughput and effective use of the laser. The shutter's closed condition should block the light so there is no permanent effect on the chip when the table is moved.

The designer should be careful about the extension of the P-well on its design. Even if allowed by the DRC checker, the P-well should not extend less than 3μm from a link. This is to avoid a connection to the substrate; such connections were seen in links closer than 3μm from the P-well, but not in those at a greater distance. It is assumed these shorts occur because the P-well is shallower near its edge. In addition, the power of the laser has to be kept reasonable to avoid a vertical connection to the P-well.

One difficulty with the Mitel technology comes from the silicon nitride passivation layer. The laser produces large damage which can interfere with surrounding structures and block the laser for further processing. The designer should be aware of this and keep a reasonable distance between structures needing repair. The silicon nitride may become conductive when zapped with the laser and interconnection between metal 1 and 2 may be possible [12], although not encountered during these experiments.

The batch linking and cutting is reliable over a small area and by keeping the same focus. The speed is slow for linking, around 2 s per link, and it must be improved if a large number of links have to be made. A design with all the links aligned is faster to zap than a random pattern. The z axis has to be very stable if the focus has to be changed. In the current setup, there is an x and y movement when the focus is changed, causing a misalignment of the coordinates.

These experiments have shown linking and cutting is possible with the Mitel technology but improvements in the laser table control are needed for large batch jobs.

3.2 Practical Example: Test Vehicle for a Wafer Scale Thermal Pixel Scene Simulator

This section describes the laser linking work done on a wafer scale test vehicle designed by M. J. Syrzycki, L. S. Carr, G. H. Chapman and M. Parameswaran: the Thermal Pixel Scene Simulator [13]. It combines micromachining and wafer scale restructuring techniques to build a large array of infrared emitters. The main purpose of this section is to present an example of the restructuring work done with the laser table on the laser links.

3.2.1 Design

Figure 3.9 shows the layout of the basic transducer cell. On the upper right, the thermal pixel, a micromachined device that emits infrared radiation when a current is applied, can be seen. Beneath the device is a pixel driver, to control the current fed to the emitter. The local memory is used to store the value of the pixel. The A/D converter is used to convert the signal from the photodiode. This part of the design was not used during these experiments.

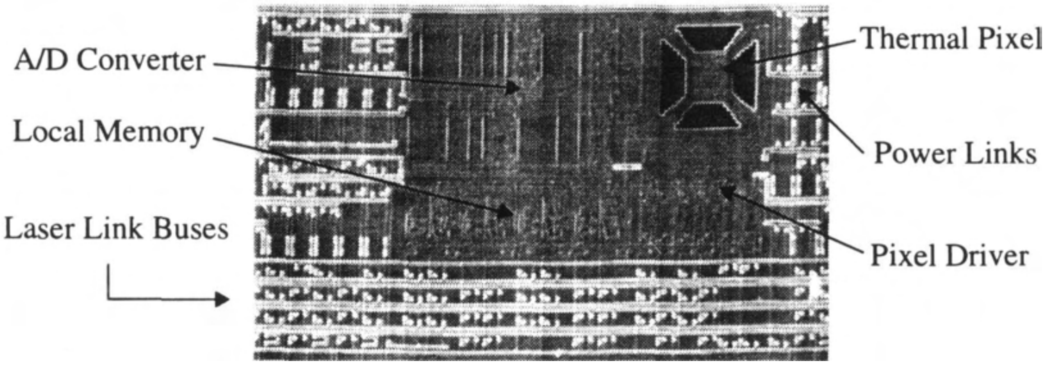


Figure 3.9 SEM Photograph of the Transducer Cell (1260μm x 742μm)

Surrounding the basic circuitry are restructuring laser link buses. On the right, the large laser links are used to hook up the power to the cell and to drive the current in the thermal emitter. The other laser links serve to connect the different signals to the logic part of the design. The laser links are disposed in an alternate up and down fashion to provide a denser bus. The design was manufactured in Northern Telecom 3μm CMOS. The photograph of the test chip is shown in Figure 3.10. The test chip is a 4x2 array of transducer cells with the laser link buses running across the entire chip.

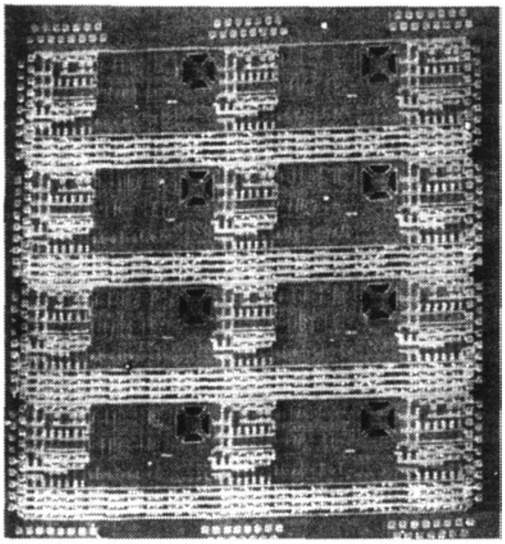


Figure 3.10 SEM Photograph of the Test Chip (7mm x 7mm)

3.2.2 Experimental Procedure

When first powered up, the power rails are disconnected and there is no power consumption. Before making the connection to the power rails, and before any laser link is connected, optical probing [14] is performed. By shining the laser at low power (~3mW) at the junction between the substrate and the active region, electron-hole pairs are created that generate a small photocurrent (around 30μA) between the substrate and the line connected to the link. By measuring the photocurrent, the path resistance can be measured and the signal route verified. This is shown in Figure 3.11.

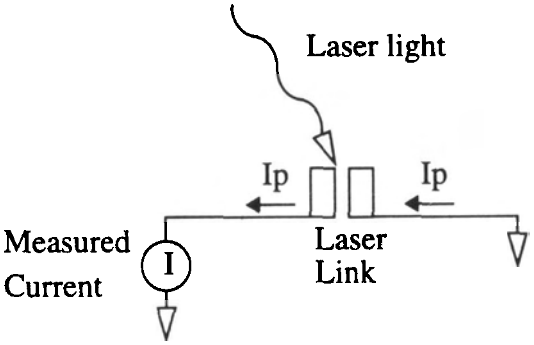


Figure 3.11 Optical Probing

The first step is to connect the power to the cell (Step 1 in Figure 3.12). When done, the power consumption is measured. If the current draw is normal, the interconnection of the signal lines can begin. If there is a high power surge, indicating a short in the cell circuitry, the cell is disconnected. The first step in interconnecting the signal buses is to connect the four signal lines to the driver circuitry and test its operation (Step 2). If this is successful, the hook-up of the latches can be performed (Step 3). To do this, the line to the driver circuit is cut and by connecting two laser links, the cut is bypassed and the signal redirected into a D flip-flop. The final test to the cell is then performed and consists in being able to drive the pixel with a four bit memory, resulting in 16 possible current draws. Typical cell interconnections required 19 links and 5 cuts.

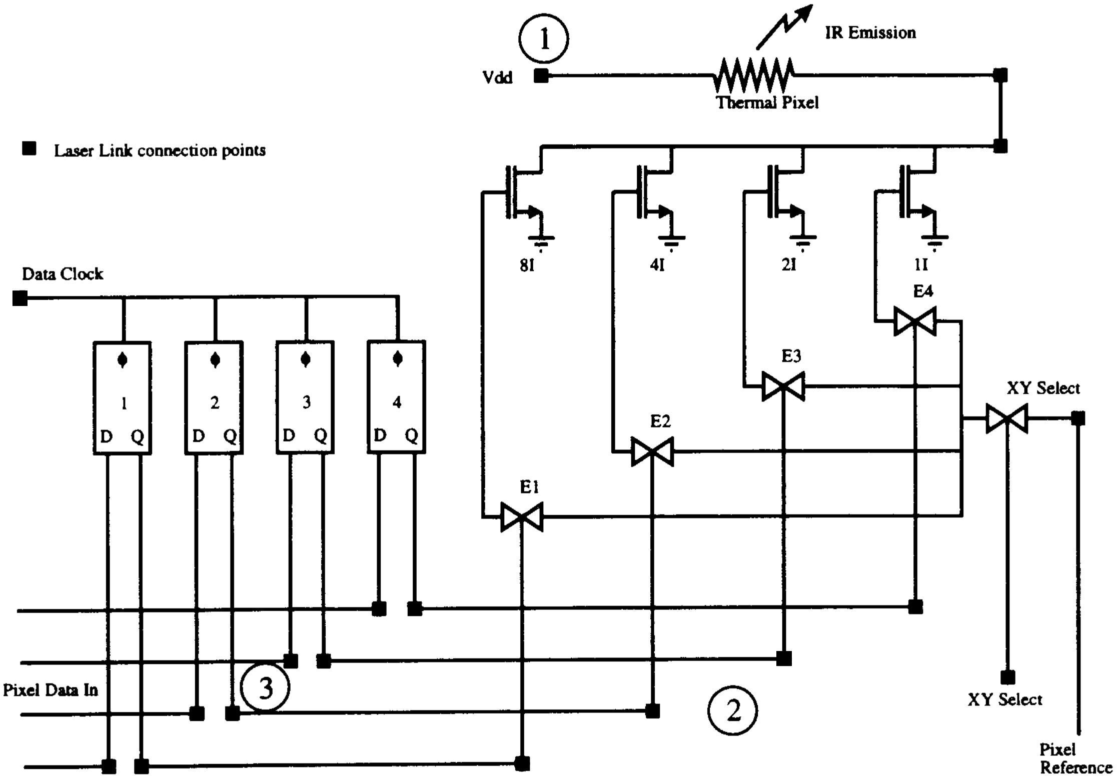


Figure 3.12 Design Schematic

3.2.3 Experimental Results

Two types of chips were tested. The first one was tested as fabricated while the other one was anisotropically etched to form a suspended plate holding the pixel. The parameters for the laser links and cuts were extracted the same way as explained in the previous section for the Mitel technology. There was no difference found in the parameters for both chips. The typical resistance for the standard laser link was 75Ω while the wider power links showed a 25Ω resistance. The first test chip, which was unetched, had seven operating pixels, six of which were latched. The etched chip was fully functional, each one of the eight pixels working with the latching circuitry [15].

This work, performed early in the master program, was very useful in learning the basics of wafer scale integration and also to learn how to use the laser table system. Many of the concepts were later used in the design and test of the FPGA vehicle.

3.3 Summary

This chapter has described the experimental work done with the laser linking wafer scale technology. In the first section, the experimental procedure to extract the linking and cutting parameters was presented while the second section dealt with the work done on another type of test vehicle, the wafer scale thermal pixel scene simulator.

The experiments described above provided useful insights and necessary results in the elaboration of the FPGA test vehicle.

Chapter 4

Defect Avoidance in FPGAs

The key in building a working Wafer Scale field programmable gate array is to design a system to eliminate the different types of defects present on the wafer after fabrication. In this chapter, there is a brief introduction of the types of defects and faults they create. Thereafter a summary of different defect avoidance techniques will be presented and the requirements for restructuring an FPGA will be investigated.

The following section concerns restructuring algorithms and their effects on the harvest of good cells. Simulations are made to test the performance of the algorithms and the architectures. An other section concerns the design requirements for building a wafer scale FPGA while the last section is an overview of the different tools needed to program a large FPGA and how they differ from the commercially available software.

4.1 Defect Avoidance

This section treats of the general defect avoidance techniques and how they can be applied to FPGA restructuring. In all these cases only one type of FPGA cell throughout the wafer is assumed.

4.1.1 Fabrication Defects

There are numerous defect mechanisms in any microelectronic process. The goal of this section is not to explain every type of defects but rather to classify the faults they create to find a proper way to avoid them. Figure 4.1 shows an example of the major categories.

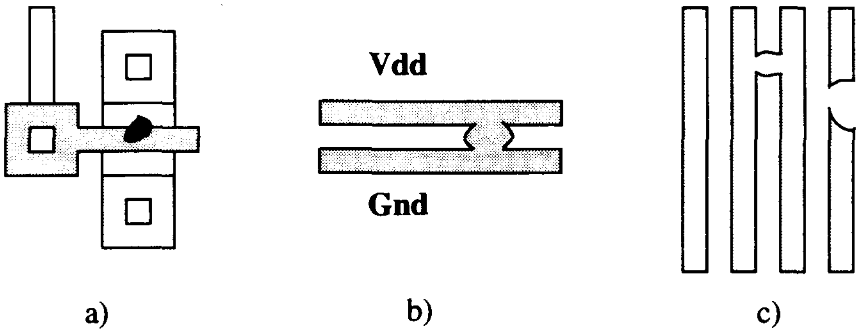


Figure 4.1 Three Categories of Defects: a) Logic Defect: e.g. Gate Oxide Hole; b) Power Defect: e.g. Power Short; c) Routing Defect: e.g. Bus Open Circuit and Bus Short

Logic Defects: all the defects affecting the logic operation of a circuit are grouped under this category. Those defects can be of many types, such as misalignment, pinhole defects, shorts or open circuits; their effect is localized, however, and affects only the logic operation of a certain part of the circuit.

Power Defects: these types of defects can be caused by many defect mechanisms, but the most common outcome is the power bus metal to metal short. This is the most critical kind of defect because if it is not taken into account in the design, just one of these defects can kill an entire wafer even before tests can be performed. For this purpose a special defect avoidance scheme must be employed for this category of defects.

Routing Defects: this category includes all the defects that affect the buses on the wafer, either the signal buses or the reconfiguration buses. They can be very deadly if they are not taken into account because the reconfiguration circuitry can be inoperative, killing the entire wafer.

4.1.2 General Defect Avoidance

Defect avoidance is defined as the different ways to avoid defective parts of a circuit and provide means to employ the working parts to build a larger circuit than achievable with standard microelectronics. One way to obtain this is to divide the circuit into identical parts. They can be rows, running from side to side of the wafer, or they can be cells, a small part of circuitry that can perform a certain function. Defect avoidance is realized by providing spares that can be connected instead of the defective cells or rows. This is called redundancy. The level of redundancy depends on the density of the defects and the desired yield. There are two classes of redundancy: global and local sparing. In global sparing, a spare can replace any of the cells in the circuit; this is very versatile but can lead to long delays if the spare cell is situated far away. There are also some applications where the physical placement of the cell is critical, like large sensor or transducer arrays. The kind of redundancy used then is called local sparing [16], where the spares are physically close to the original cell. Figure 4.2 gives an example of the two redundancy classes. Figure 4.2 a) shows an example of a spare column of cells where the spares are used to replace two defective cells to form a 6x6 array of working cells. In Figure 4.2 b), local sparing is used to produce an array of 3x3 cells. The dashed line encloses the cell and its spares. Only one out of those four cells needs to be working.

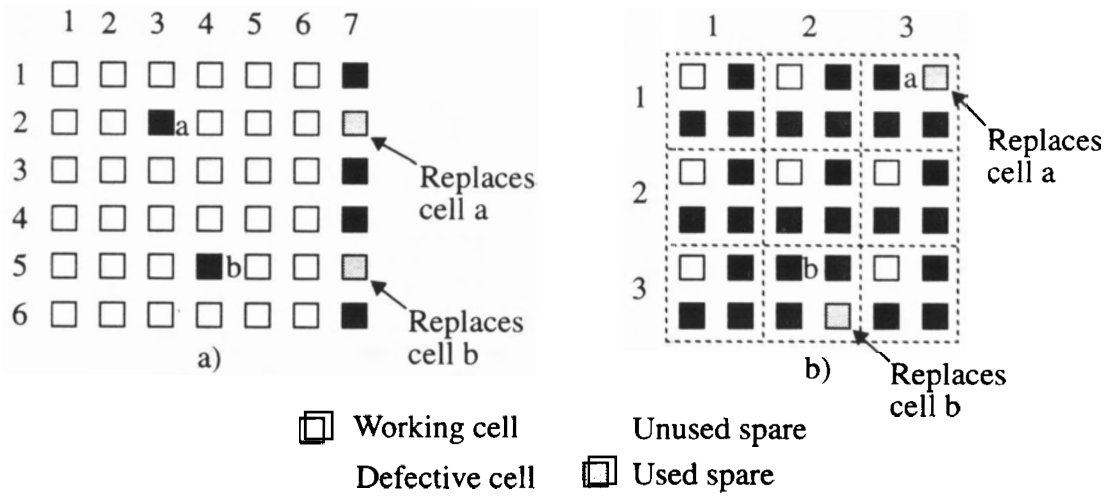


Figure 4.2 Two Redundancy Classes: a) Global Sparing; b) Local Sparing

For FPGAs, the physical placement of the cells is not critical because all cells are identical. The spare cell, however, must be close to the defective cell in order to reduce the time delay between the cells. Local sparing is ideal for that purpose but requires very large overhead. The best way to restructure an array of cells is not to use dedicated spare cells, but rather build the array using the closest available cell as a spare. This means every cell in the array can be a spare.

4.1.3 Making the Defect Avoidance Invisible to the User

The idea of a wafer scale FPGA has been proposed by others[5], but was presented with a different approach to defect avoidance. In this earlier paper, the defect avoidance is performed by the FPGA software itself, using the inherent reconfigurability of the FPGA circuitry. However, the FPGA software has to be aware of which cells in the array are defective to bypass them. It may also cause problems because the FPGA may become no longer symmetrical. Macro circuits already optimized cannot be used because of the defective cells breaking the array. This method is also not tolerant of certain faults such as power shorts. Although this method requires no overhead for restructuring, the above reasons make it hard to use.

The proposed wafer scale FPGA in this thesis does not use the electronic bypass capability of its routing architecture but rather physical restructuring switches. While this technique uses a minimum of overhead because of the small area occupied by the laser links, the major advantage of this method is to make the restructuring invisible to the user. The restructuring consists in harvesting a two dimensional array of working cells from an array containing defective cells and/or buses. By using different techniques, the array appears fault free and the actual map of the defects does not have to be known by the user when programming the FPGA.

The following section explains how to restructure such arrays to provide a restructuring invisible to the user.

4.2 Restructuring of a 2-D Array

Different restructuring techniques are presented. They all have the same goal, i.e. to build the largest 2-D array from a basic array containing defects. Advantages and drawbacks of these techniques are evaluated and their potential for building FPGAs discussed.

4.2.1 Row-Column Substitution

The simplest way of avoiding defects is the row-column substitution. If a defective cell is found during the tests, the entire row or column containing this cell is bypassed; this method is very fast and requires simple algorithms. The bypassing circuitry is kept to a minimum, since the signals only have to go through the cell and reach the adjacent one. Figure 4.3 shows a 2D array of 6 x 6 cells being restructured using the row-column substitution technique. The algorithm must alternate and substitute a row after a column in order to maximize the size of the array. Proceeding that way, the worst array is equal to the size of the original array minus the number of defects divided by 2. The restructuring is however more complex, because of defects occurring in the bypass circuitry. The algorithm must bypass cells with defects in the reconfiguration circuitry first, since they only can be bypassed either by a column or a row.

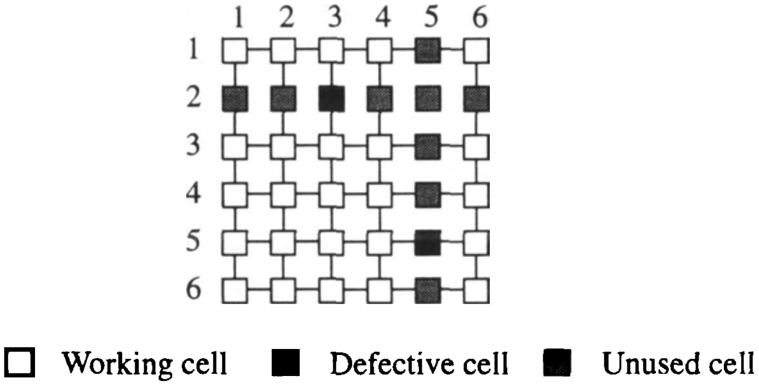


Figure 4.3 Row-Column Substitution

As an example, the bottom defective cell (row 5, column 5) in Figure 4.3 has to bypass the signal from the cell on its left to the cell on its right. This can only be done if the horizontal bypass circuitry is not defective. If this circuitry is defective, then this cell has to be bypassed with a row and the other defective cell would be bypassed by a column to keep the logical array at 5x5 cells. Of course, both the horizontal and vertical bypass circuitry may be faulty; in this case, the algorithm must use both the column and the row substitution, reducing the size of the final array.

While this method is simple and economical in time, it leaves lots of unused cells and, if the defect density is high, the final array will be very small. The size of the array should increase if the defects tend to agglomerate, because many defects can be bypassed with only one column or row substitution. Due to its simplicity, this method is well suited to the restructuring of smaller arrays where the yield is already high. A method similar to dynamic RAM memory column substitution will be investigated for the restructuring of small FPGAs later in this chapter.

4.2.2 Cell by Cell Substitution

The next method is called cell by cell substitution. When a defective cell is encountered, a neighboring cell is used to replace it. Special restructuring buses are placed between the columns of cells. With a combination of switches, the defective cells can be bypassed and an array can be constructed.

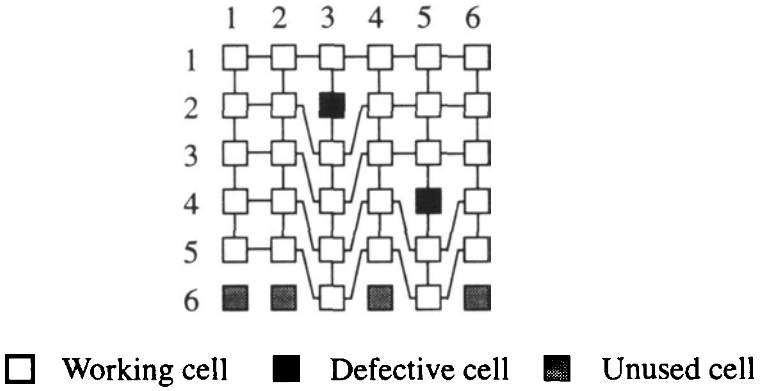


Figure 4.4 Vertical Cell by Cell Substitution

An example is shown in Figure 4.4. There is a defective cell in row 2, column 3. When connecting the rows, the cell in the row below (row 3) is used to replace the defective cell (row 2). So every cell in column 3 must be shifted down in order to complete the rows. An interesting thing happens if another defective cell appears in column 3. Now the cells need to be shifted 2 rows down in order to complete the restructuring. There are two ways to handle this problem: the first is to provide an additional restructuring channel between each column. This is straightforward but requires additional area and the maximum number of defective cells allowed in a row is equal to the number of restructuring channels. The second way is to use pseudo faults: working cells are sacrificed to allow the use of only one restructuring channel while being able to restructure an array with many defective cells in the same area. This is very important because the defects tend to cluster on a wafer. In the vertical cell by cell substitution, the columns are kept straight while the rows are shifted down; this means more rows than columns are needed to restructure a square array. Another way to perform the cell by cell substitution is to use restructuring channels in both the vertical and horizontal direction. This allows efficient harvesting but requires complex algorithms and the major drawback is the high amount of overhead involved in these architectures. Because of the large area already occupied by the FPGA routing, this technique is not investigated in this thesis.

4.2.3 Row-Column and Cell Substitution

The cell by cell substitution is an efficient way to restructure an array, but it assumes that the restructuring circuitry is fault-free. The best way to restructure FPGAs is to use a combination of the two methods presented above. The main restructuring remains the cell by cell substitution, however a set of extra columns is also provided. This extra set has a dual purpose: first, it allows the bypass of an entire column if a restructuring bus is defective; secondly, the extra columns can be used to replace columns containing the most defects and gain extra rows. An example of such a restructuring is shown in Figure 4.5. The defective cell (2, 3) was bypassed using the cell by cell substitution while the cluster of three defective cells in column 5 was bypassed by the column substitution method. The result is a 5x5 array while cell by cell substitution alone would allow only 3 rows.

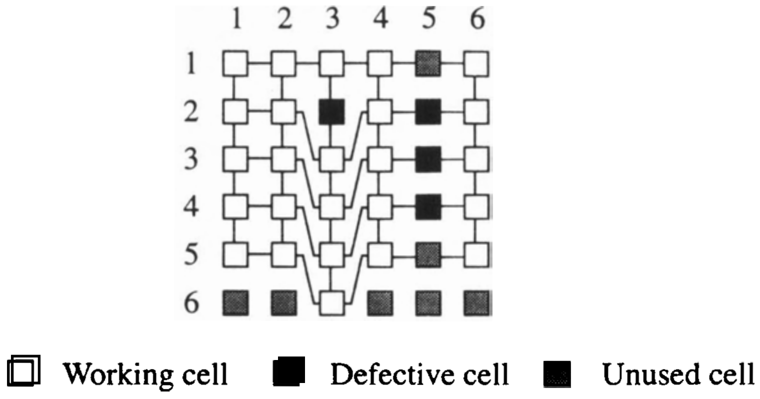


Figure 4.5 Row-Column and Cell Substitution

4.3 Algorithms and Yield Simulations

This section shows the algorithms and yield simulations performed to find the best restructuring technique applicable to FPGAs. The restructuring methods elaborated in the last section are studied and explained in detail, while the algorithms and Monte-Carlo simulation results are set out and discussed. A brief description of the defect distribution model is also presented.

4.3.1 Defect Distribution Simulations

There are many papers dealing with the simulation of the defect distribution of a particular fabrication process [18-23]. In earlier yield models, the defect distribution on the wafer was thought to follow a Poisson distribution:

$$P(X=k) = \frac{e^{-\lambda}\lambda^{k}}{k!} \qquad (4.1)$$

where λ=defect density per unit area, k=number of defects and P=probability of having k defects in the unit area. This distribution means that the probability of a defect appearing in a region of the wafer is completely independent of the defects already present.

Experimental data however shows the probability of a defect appearing in an area is dependant on the number of defects already present in this area. This phenomenon is called defect clustering. The distribution is then better represented by a Negative Binomial Distribution:

$$P(x, S) = \frac{\Gamma(x+\alpha_c)}{x!\Gamma(\alpha_c)} \frac{(\lambda/\alpha_c)^x}{(1+\lambda/\alpha_c)^{x+\alpha_c}} \tag{4.2}$$

where P=probability of having x defects in an area S, λ=defect density and α_C=cluster coefficient. In most models the area used for λ is that of the circuit block or cell. Clusters begin to appear on the wafer, depending on the value of the α_C parameter. A low value for α_C means a high clustering. An infinite α_C parameter means no clustering, α_C=1 is moderate clustering while α_C=0.1 is high clustering. Values of α_C ranging from 0.125 to 4 have been encountered in samples of different products [22].

It is almost impossible to create a model that will perfectly reflect the defect distribution of a known process. Extensive research on the process itself can only give partial knowledge of the defect distribution. A defect distribution Monte Carlo simulation was developed here at SFU. The goal was to distribute defects on a wafer with a distribution that follows the Negative Binomial Distribution. The simulation is based on the model presented by C. H. Stapper in [23]. The program starts with an array of non defective cells; after a time interval Δt, the appearance of a defect in the cell is calculated by comparing a random number generated with an assigned probability for each cell. This probability is a linear function of the number of defects in the cell and in its four nearest neighbors. The weight associated with the number of defects in the cell is higher than the weight for the neighbors. By changing the value of these weights, the α_C parameter can be changed. If the number of defects in the cell itself and the neighboring cells is not taken into account, a Poisson distribution is obtained. The program stops when the desired defect density is obtained. An example of two defective cell maps is shown in Figure 4.6.

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

(a) $\alpha_c=\infty$ (no clustering)

```
0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4
0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 4 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0 0 0 0 2 2 1 0 0 0 0 2 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 3 1 0 0 1 3 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

(b) α_c=0.1 (high clustering)

Figure 4.6 Defect Map Example (λ=0.1)

The map on the left (Figure 4.6 a) is a pure Poisson distribution ($\alpha_c=\infty$) while the one on the right (Figure 4.6 b) has a very small clustering coefficient (α_c=0.1) thus significant clustering. Both maps have the same average number of defects per cell (λ=0.1). Note on the right that certain cells have a very high number of defects and how the defects tend to be grouped in clusters.

An important parameter in any Monte Carlo simulation is the number of wafers simulated to realistically represent the distribution. Different tests have been performed on the model to test the number of iterations required. These tests show that approximately 100 wafers give distributions with an average α_c parameter quite constant between distributions. Thus in the simulations, wafer lots of 100 wafers will be used. Table 4.1 shows the results of the simulations for an array of 100x100 cells. Ten lots with 100 wafers each were simulated with three different cluster parameters. The average simulation time is one minute per wafer lot on a Sparc10. The table shows the average defect density λ and the average α_c, along with their respective standard deviations.

λ		α_c	
$\bar{\lambda}$	σ	$\bar{\alpha}_c$	σ
0.106	$2.4\text{x}10^{-4}$	2.06	0.26
0.107	$4.1\text{x}10^{-4}$	0.60	$8.5\text{x}10^{-3}$
0.103	$2.0\text{x}10^{-4}$	0.23	$1.3\text{x}10^{-2}$

Table 4.1: Distribution of Wafer Lots; target λ=0.1

This method of simulating defects may not represent exactly a particular fabrication process but approximates Negative Binomial Distributions with sufficient accuracy to perform restructuring simulations. The model should be compared to an existing production line to modify the parameters and ensure better accuracy.

4.3.2 Row-Column Restructuring

The purpose of this section is to demonstrate that it is possible to increase the yield of current size FPGAs with a technique similar to the one used to reconfigure dynamic RAM chips. The largest currently available FPGAs have very low yields. By providing a small amount of extra rows and columns, it is possible to restructure the array and therefore increase significantly the yield. Because of the restriction in size, the amount of overhead must be kept to a minimum and the delay added by this overhead must also be small, in order to keep the performance very close to full custom FPGAs. For these reasons, the row-column restructuring is the best way to increase the yield of these devices.

The restructuring algorithm used is very simple; it consists in restructuring first the routing defects with the appropriate bypassing of either a row or a column. Then logic defects are bypassed using a row or a column, depending on the spares available. The C-like pseudo code is presented in table 4.2.

```
/* Restructure routing defects*/
for(i=1; i<row_number; i++) for(j=1; j<col_number; j++)
    { if(defect[i][j]==vertical_routing) bypass_col;
    if(defect[i][j]==horizontal_routing) bypass_row; }
/*Restructure logic defects*/
for(i=1; i<row_number; i++) for(j=1; j<col_number; j++)
    {if(defect[i][j]==logic)
        if(col_sum<row_sum) bypass_col;
            else bypass_row;} /*Use row or column spare depending on availability*/
```

Table 4.2: Row Column Algorithm C-like Pseudo-code

Better algorithms are presented in [17] and [24]. Simulations with this simple algorithm shows that even without the best procedure the yield is increased significantly. For the simulation, batches of 1000 chips containing an array of 25x25 cells are used. The approximate dimensions of the largest currently available FPGA (2cm x 2cm) are used. The defect density is adjusted to obtain approximately a 5% yield in non restructurable arrays (the cell yield is then 99.5%). The defects in the cell can either affect the logic or the routing. As stated in [17], a defect in a cell has a 40% chance of affecting the routing, a fact too often neglected by reconfiguration models. After each chip is simulated, it is restructured. The percentage of chips successfully restructured is then calculated. The results for the yield are shown in Figure 4.7. In this simulation, a Poisson distribution was assumed. A physical array dimension of 26 means there is one extra row and one extra column, a physical array dimension of 27 means two extra rows and two extra columns and so on. The physical array dimension of 25 represents a chip with no restructuring capability.

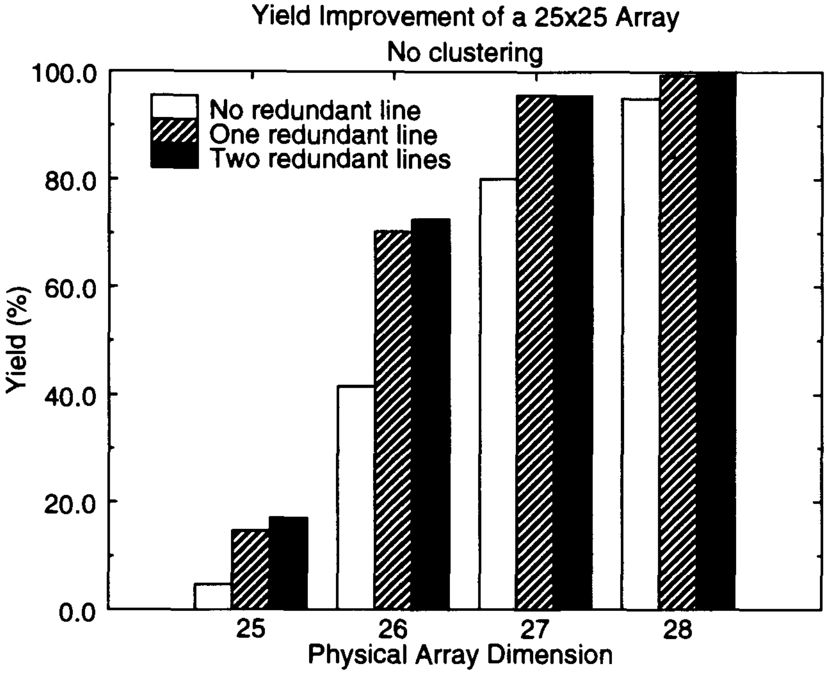


Figure 4.7 Yield Results for a Logical 25x25 Array, no clustering (λ=0.005)

In Figures 4.7 through 4.10, the horizontal axis represents the number of physical rows and columns available to build the 25x25 array. The dark columns are the results when one and two redundant lines are added to the cell, for both the vertical and horizontal routing channel. If the simulation says the defect is in the bus channel, the routing cannot bypass using lines in that area. Adding n extra lines, however, will allow routing to be possible for a number of defects ≤ n in the channel. Even with only one extra row and one extra column, the yield is increased by almost a factor of 9. Yields near 100% can be achieved with 3 extra rows and 3 extra columns. The use of a redundant line increases the yield, but requires extra overhead that may cause additional delays. Yield results with wafer showing high clustering (low α_c) are shown in Figure 4.8. The simulation is said to be high clustering because in small arrays, it is hard to evaluate the α_c parameter because the defect density is very low. The yield after restructuring for wafers with clustering are slightly better than those without clustering. This is due to the higher probability of having defects in the same columns or rows.

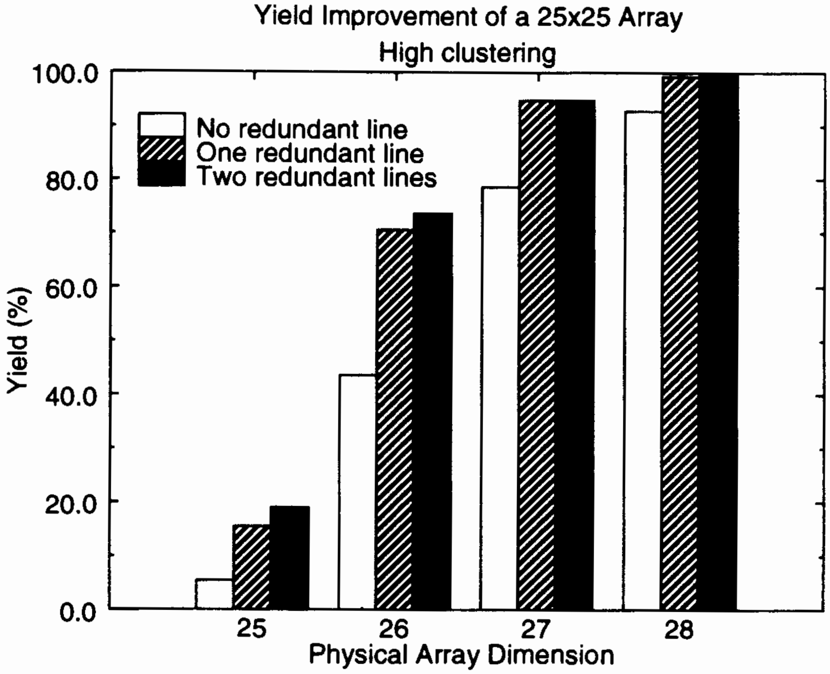


Figure 4.8 Yield Results for a Logical 25x25 Array, high clustering (λ=0.005)

The two simulations are extreme cases and a standard process should fall in between as far as clustering is concerned. The simulations without any clustering give the worst case. These simulations were repeated with a lower yield of 99% for each cell. The results for no clustering are shown in Figure 4.9 while the results for high clustering are shown in Figure 4.10. The yields are lower than the previous simulations, because of the higher defect density. The improvement is nevertheless important, especially with a redundant line. The second redundant line increases the yield more in highly clustered chips. While its effect was too small in the other simulation, this simulation with a lower yield shows that the possibility of adding two extra lines should be taken into account for highly defective chips.

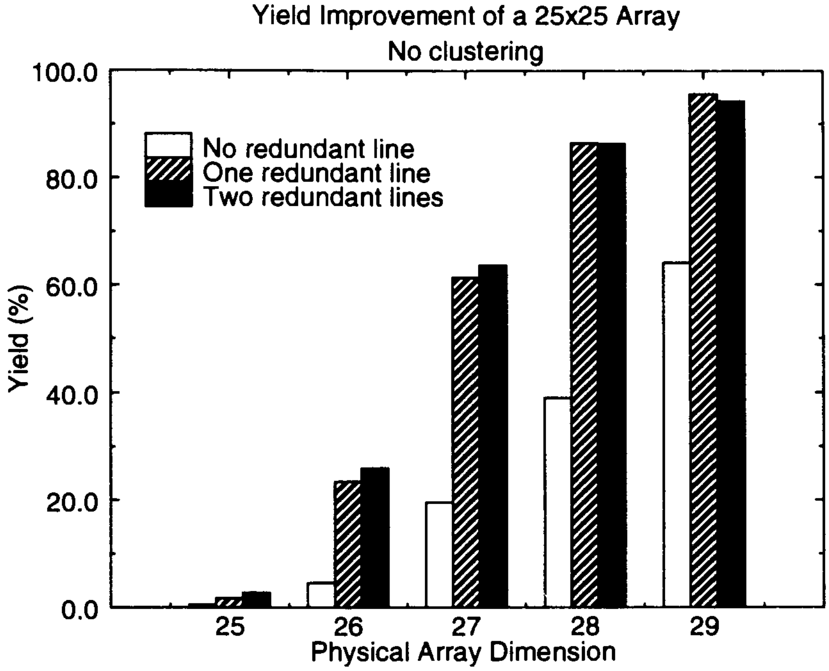


Figure 4.9 Yield Results for a Logical 25x25 Array, no clustering (λ=0.01)

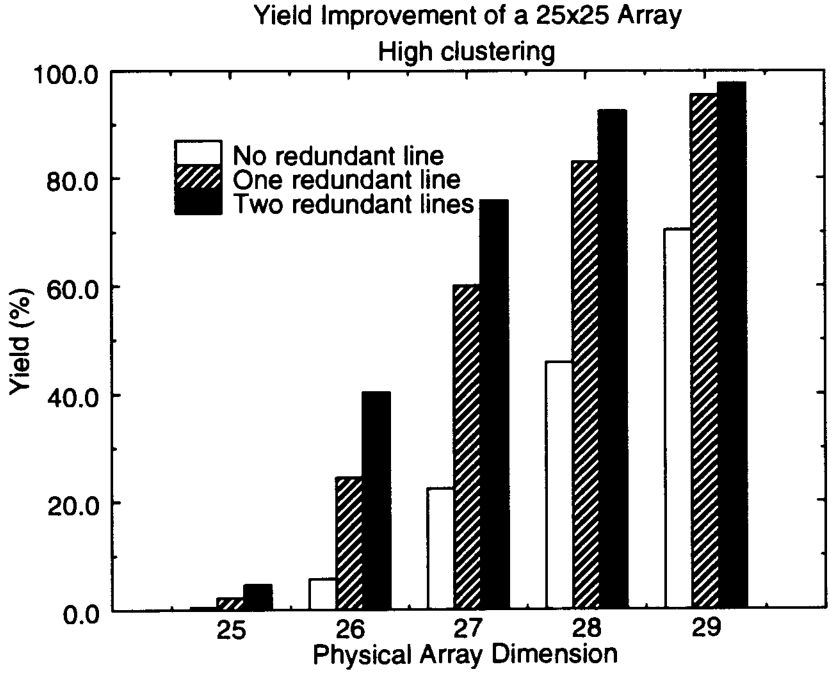


Figure 4.10 Yield Results for a Logical 25x25 Array, high clustering (λ=0.01)

These simulations show it is possible to increase the yield of currently available chips with a simple row/col restructuring and thus reducing the production cost. The rework needed to restructure the arrays is small and could be compared to the rework needed in dynamic RAM chips. There is a delay added to the circuit but as will be shown, the use of the laser link minimizes this delay.

4.3.3 Cell by Cell Restructuring

For the cell by cell restructuring, a different simulation approach is made. Whole wafers with 100x100 cells are simulated and the clustering is identified by the alpha parameters calculated from the lot. To perform this kind of restructuring, a special algorithm called the Gupta Algorithm [17] is needed (shown in Table 4.3). The purpose of this algorithm is to build a logical array of good cells from a physical array containing defective cell. The physical and logical columns are identical, only the logical row numbers are changed. Assuming i is the current physical row index and i' is the logical row index, the algorithm starts with i=1, i'=1. The i'-th logical row is configured by selecting the first available usable cell (i.e. neither faulty nor a pseudo-fault) from the top, in every column. When all the cells have been assigned to the i'-th logical row, the pseudo faults are determined. For two consecutive cells in the i'-th logical row, when cell(e,j), (from the e-th row and j-th column), and cell($f,j+1$) have $e=f$, there are no pseudo-faults between them. If $e<f$, every cell(k,j) for $e<k<f$ will be assumed a pseudo-fault. If $e>f$, every cell($k,j+1$) for $f<k<e$ is declared a pseudo-fault.

```
for(i=1; i<row_number; i++) /*Scan the rows*/
    last_row=i;
    for(j=1; j<col_number; j++) /*Scan the columns*/
    x=0;
    while(cell[i+x][j]=defective) x++; /*Find the first non defective cell*/
    cell[i+x][j]=i; /*Assign the row number to this cell*/
    if(last_row<i+x) /* If the row number is smaller than the last column*/
        for(z=last_row+1; z<i+x; z++) /*Scan the cell in the previous row */
            if(cell[z][j-1]!=defective) cell[z][j-1]=pseudo_fault;
                /* If the cell is not defective, it is declared as a pseudo-fault*/
    if(last_row>i+x) /* If the row number is larger than the last column*/
        for(z=i+x+1; z<last_row; z++)/*Scan the cell in the current row */
            if(cell[z][j-1]!=defective) cell[z][j-1]=pseudo_fault;
                /* If the cell is not defective, it is declared as a pseudo-fault*/
    last_row=i+x;
```

Table 4.3: Cell by Cell Substitution C-like Pseudo-code

Figure 4.11 shows an example of this type of restructuring where a cluster of two cells in the same column (2) is bypassed. In a), the cell in the first physical row (1, 1) is assigned to the first logical row (1', 1'). In b), since the cells in the first and second physical rows are defective, the third cell (3, 2) is assigned to the first logical row (1', 2'). The cell in the column on the left (2, 1) must be declared a pseudo fault. Then finally in c), a cell in the first row (1, 3) is assigned to the logical row (1',3'). The physical index on the left being greater (3 compared to 1), the cell (2, 3) in this column must be declared as a pseudo-fault. Pseudo-faults are considered exactly like defective cells in the algorithm.

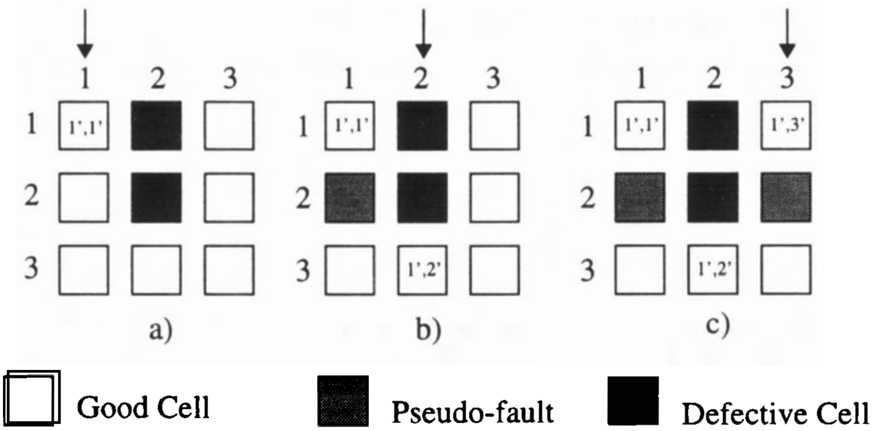


Figure 4.11 Gupta Algorithm Restructuring example

This algorithm assumes a perfect routing channel. As seen, this is not realistic. To this method of restructuring, the row and column substitution must be added in order to circumvent the routing defects. Figure 4.12 shows an example of the restructuring on a 25x25 array. The numbers indicate the index of the logical rows (there is no column bypass in this example).

```
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 3  3  3  3  3  3  3  #  3  3  3  3  #  3  3  3  3  3  3  3  3  3  3  3  3
 4  4  4  4  4  4  4  3  4  4  4  4  3  4  #  4  4  4  4  4  4  #  4  4  4
 5  5  5  5  5  5  5  4  5  5  5  5  4  5  4  5  5  5  5  5  5  4  5  5  5
 6  #  #  6  6  6  6  5  6  6  #  6  5  6  5  6  6  6  6  6  6  5  6  6  6
 7  6  6  7  7  7  7  6  7  7  6  7  6  7  6  7  7  7  7  7  7  6  7  7  7
 8  7  7  8  8  8  8  7  8  *  #  #  7  8  7  8  8  8  8  8  8  7  8  8  8
 9  8  8  9  9  9  9  8  9  8  7  8  8  9  8  9  9  9  9  9  #  8  9  9  9
10  9  9 10 10  # 10  9 10  9  8  9  9 10  9 10  # 10 10 10  9  9 10 10 10
11 10 10 11 11 10 11 10 11 10  9 10 10  # 10 11 10 11 11 11 10 10 11 11 11
12 11 11 12 12 11 12 11 12 11 10 11 11 11 11  *  #  * 12 12 11 11 12  # 12
13 12 12 13  *  #  #  #  * 12 11 12 12 12 12 12 11 12 13 13 12 12  # 12 13
14 13 13 14 13 12 13 12 13  # 12 13 13 13 13 13 12 13 14 14 13 13 13 13 14
15 14  #  #  *  #  * 13 14 13 13 14 14 14 14 14 13 14 15 15 14 14 14 14 15
16 15 14 15 14 13 14 14 15 14 14 15 15 15 15 15 14  #  *  *  # 15 15 15 16
17 16 15 16 15 14 15 15 16 15 15 16 16 16 16  *  # 15 16 16 15 16 16 16 17
18 17 16 17 16 15 16 16 17 16 16 17 17 17 17 16 15 16 17 17 16 17 17 17 18
19 18 17 18 17 16 17 17 18 17 17 18 18 18 18 17 16 17 18 18 17 18 18 18 19
20 19 18 19 18 17  # 18 19 18 18 19 19 19  #  # 17 18 19 19 18 19 19 19 20
21 20 19 20 19 18 18 19 20 19 19 20 20 20 19 18 18 19 20 20 19 20 20 20 21
22 21 20 21 20 19 19 20 21 20 20 21 21 21 20 19 19 20 21 21 20 21 21 21 22
 . 22 21 22 21 20 20 21  # 21 21 22 22 22 21 20 20 21 22 22 21 22 22 22  .
 .  . 22  . 22 21 21 22 22 22 22  .  .  . 22 21 21 22  .  *  #  *  .  #  *
 .  .  .  .  . 22 22  .  .  .  .  .  .  .  . 22 22  .  .  . 22  .  .  .  .
```

# Defective cell   * Pseudo-fault   . Unused cell

Figure 4.12 Cell by Cell Restructuring Example

The simulation is done by restructuring 100 wafers and calculating the number of arrays that are successfully restructured, given a certain target array dimension. The physical size of the wafer was 100x100 cells. Figure 4.13 shows the result of the simulation.

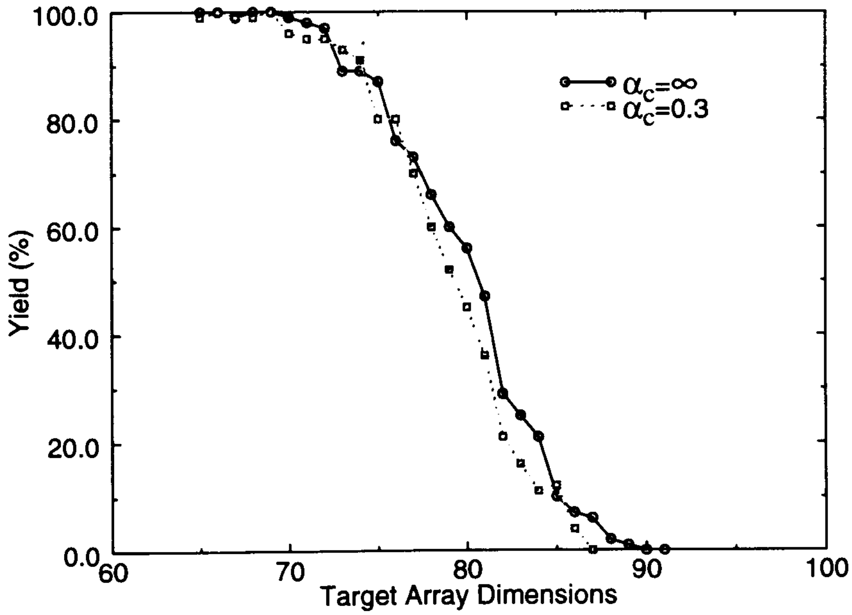


Figure 4.13 Cell by Cell Restructuring Simulation, no extra line, λ=0.01 (100 defects/ wafer)

The bottom axis represents the targeted array dimensions while the vertical axis is the percentage of wafers successfully restructured. From now on, when the term cell by cell restructuring is used, it includes the row-column bypass for the defective routing channels.Two curves are shown, the one on the left with a high clustering and the one on the right with no clustering. Both distributions have the same λ=0.01, which produces 100 defects per wafer. The results show that an array of 80x80 can be restructured with a yield of 50% while almost 100% yield is achieved with a target array of 60x60. The clustering has the effect of reducing the yield slightly. This is due to the fact that column substitution must be used to bypass the cells with a vertical routing defect. The clustering has the effect of grouping the defects together and increasing the probability of a routing defect occurring in one cell.

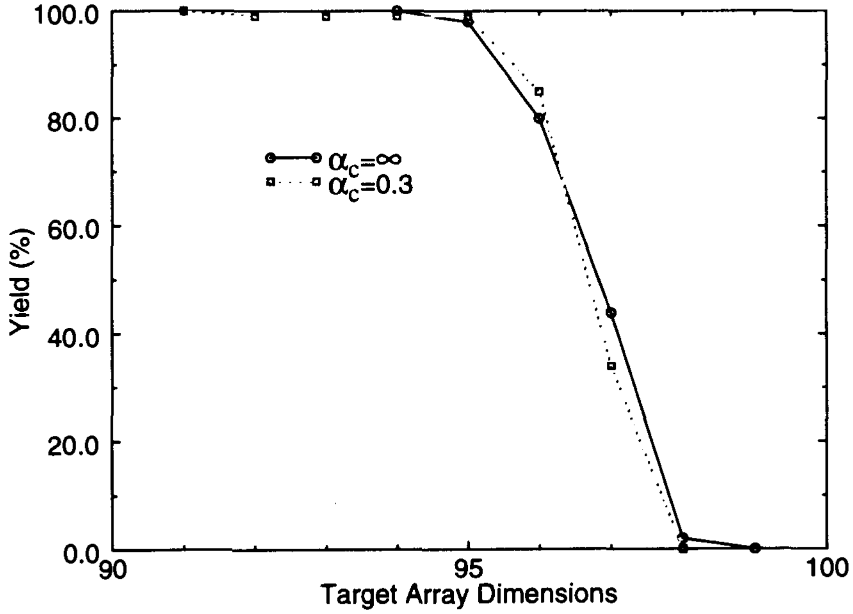


Figure 4.14 Cell by Cell Restructuring Simulation, one extra line, ,λ=0.01 (100 defects/wafer)

The major problem of this technique is the row/column bypassing of the routing defects. The yield can be increased by placing one or more redundant line in the horizontal and vertical routing. Figure 4.14 shows the simulation results. The parameters are the same as in Figure 4.13, except for the addition of an extra line in both the vertical and horizontal channels. With this extra row/col line, the clustered wafers are more efficiently restructured. This is due to the extra line that significantly reduces the number of entire columns or rows being bypassed. The effect of extra lines on clustered wafers is clearly seen in Figure 4.15. The addition of one extra row/col line increases the yield significantly while the addition of a second extra row/col line has no effect. These simulations were done for a high yield process, since λ=0.01 (but still there is 100 defects per wafers). Figure 4.16, Figure 4.17 and Figure 4.18 show the same type of simulations, this time with a process having λ=0.06, or 600 defects per wafer. In Figure 4.16, the yield obtained is very low (~50% for a target array of 35x35). In Figure 4.17, however, when an extra row/col line is added, the yield is much better (50% for a target array of 85x85). The effect of adding an extra row/col line in the channels is clearly seen in Figure 4.18.

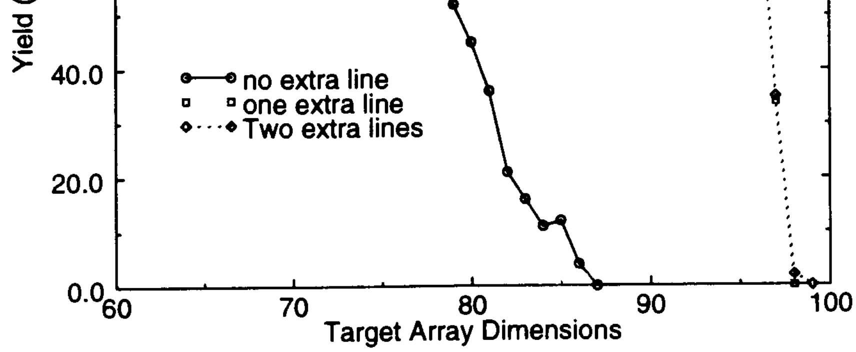


Figure 4.15 Effect of Extra Lines, λ=0.01 (100 defects/wafer), α_c=0.3

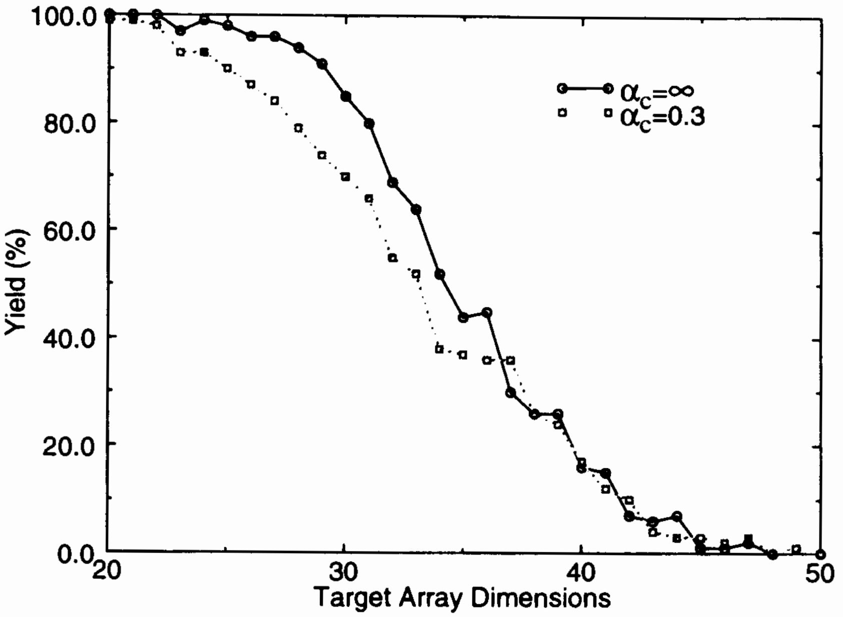


Figure 4.16 Cell by Cell Restructuring Simulation, no extra line, λ=0.06 (600 defects/ wafer)

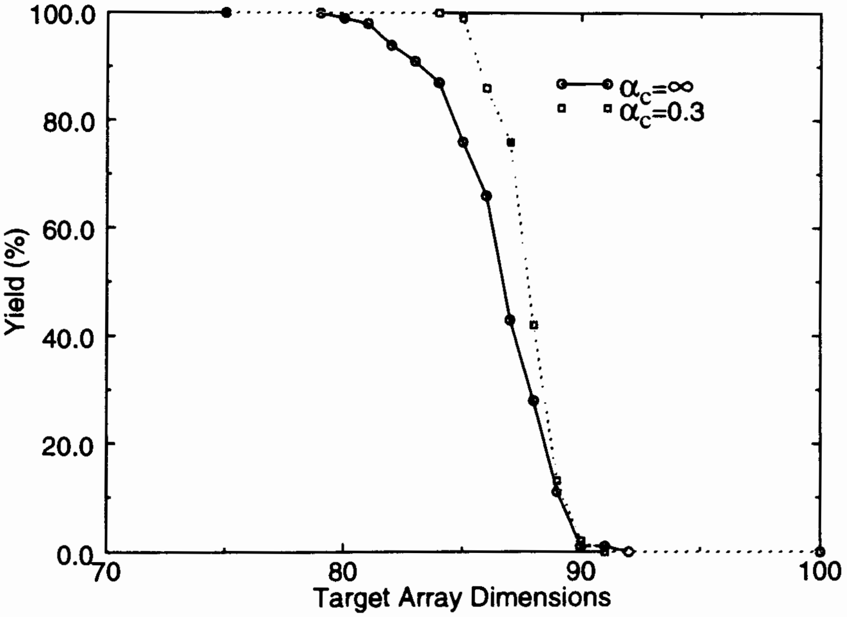


Figure 4.17 Cell by Cell Restructuring Simulation, one extra line, λ=0.06 (600 defects/wafer)

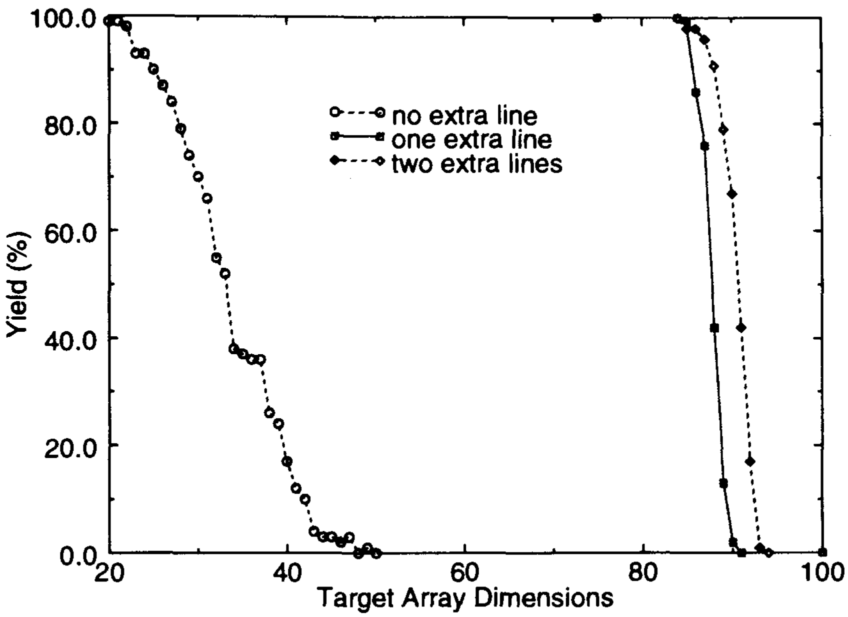


Figure 4.18 Effect of Extra Lines, λ=0.06 (600 defects/wafer), α_c=0.3

The effect of adding two row/col lines is more pronounced in this lower yield simulation. As shown the yield increase is not important enough however to justify the overhead of two extra lines. These high defect density simulations show the effect of the clustering is more pronounced. But once again the use of an extra row/col line improves the yield of the clustered wafers better than the Poisson wafers.

These simulations show the efficiency of the cell by cell restructuring for large and wafer scale FPGAs. Because of the defects occurring in the routing and the reconfiguration resources, however, it is much better to use redundancy of the lines in the cells themselves rather than the cell by cell substitution alone. Rework is also reduced because only one cell has to be linked when an extra line is available. The bypass of an entire row or column requires many links in each cell to be zapped.

The field of defect simulation is very vast. This simulation makes a number of simplifying assumptions and takes into account point defects only. A true process may have defects that are bigger and cover a large area on the wafer, affecting the routing architecture beyond repair. Note however that the approach taken in the simulation is to cluster the point defects together, simulating in a way the larger defects in one cell.

This restructuring approach was chosen because of its simplicity, its low overhead and its ease of use with FPGAs. It will be explained in a later section why the cell by cell substitution with both a vertical and an horizontal restructuring channel is hard to implement on FPGA circuits.

4.4 Design Considerations for Defect Avoidance in FPGAs

The previous section dealt with the different aspects of the restructuring but without any explanation on how to physically implement the circuits. In this current section, different approaches are investigated to design a restructurable FPGA circuit.

4.4.1 Power Routing

The most critical aspect of any wafer scale design is the power routing. A power short on the bus can kill an entire wafer, even before tests can be performed. The way to counter this problem is to design cells disconnected from the power bus and connecting them one by one to test their power consumption and check for shorts. In the WASP project [25], large transistors are used to connect the power to the cells. To test the cell, the transistor is turned on and the power connection made. Testing each separate cell can be done easily and the cells with no problems are kept powered and each one is tested incrementally.

The major drawback of using a transistor is the large resistance placed between the power bus and the power lines in the cell. This causes a voltage drop that can lead to some problems in the electrical performances. A way to counter this problem is the use of a very large transistor which offers a small resistance, but the area taken up then is very large and can become unacceptable when numerous devices are needed.

Instead of large transistors, laser links can be used to hook up the power lines. There is an example of this method in [15], where a thermal pixel cell was powered via a laser link. Experiments performed here at SFU showed no problem in using the laser link to power the cell. Two advantages are the small resistance of the link, around 100Ω for a 6.6μm wide link (down to a few Ohms when very wide links are used), and the small area taken up by the device. The drawback is the time taken to zap the laser link and to cut the power bus in the case of a short in the cell. If a large number of cells have to be tested, this method can become tedious.

A new device, combining the advantages of both methods, has been designed, fabricated and tested here at SFU. Called the Testable Laser Link, it is a combination of a laser link and a small transistor (Figure 4.19).

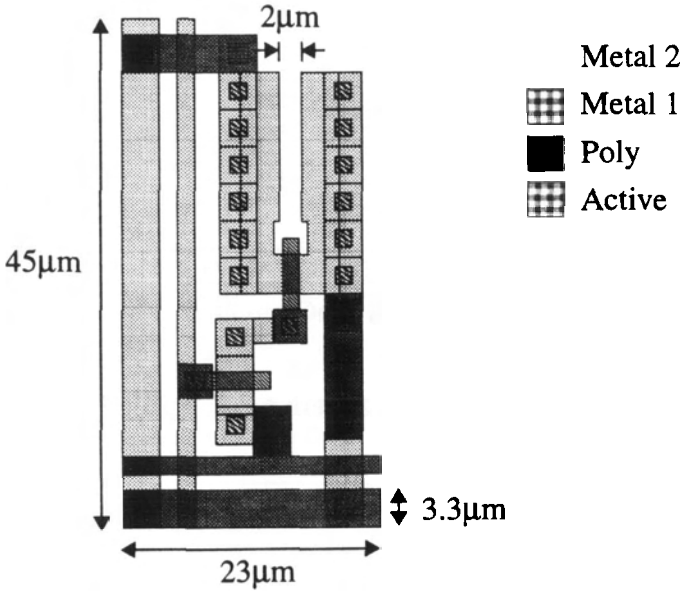


Figure 4.19 Testable Laser Link

The small transistor, when turned on, simulate the effect of zapping the laser link. This becomes very handy because the cells can be tested for shorts without having to zap the laser link, and the structure combines the small area and resistance of a laser link connection with the ease of testability of a transistor.

Electrical tests were performed to show that there were no problems in adding a gate at the end of the laser link. Both the transistor and the laser link were showing the same characteristics when combined in a single structure as they did separately. In Figure 4.20, the graph of the voltage drop for the testable laser link is shown, before and after the zapping. The gate voltage used was 5 volts. The voltage drop is measured across the source and the drain of the transistor.

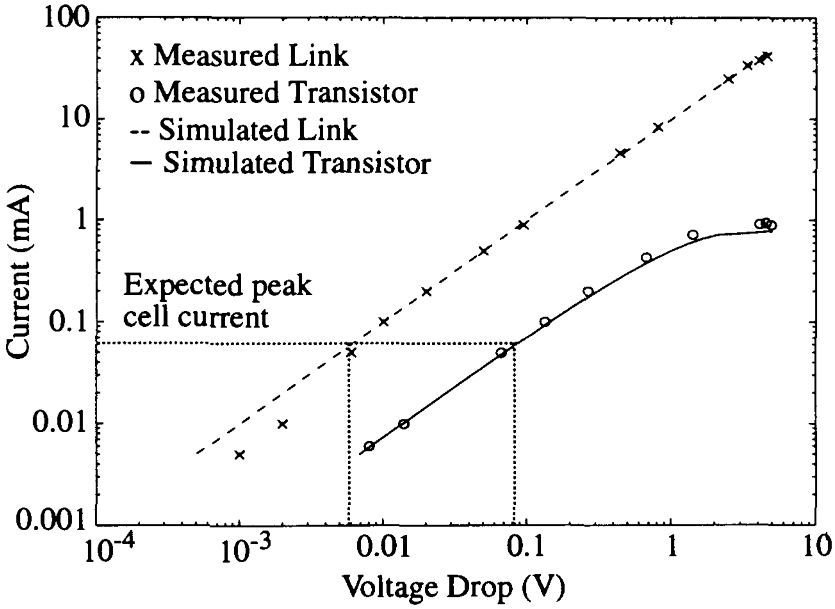


Figure 4.20 Graph of the Voltage Drop across the Testable Power Link

The plot illustrates two advantages of the combined structure: first, the resistance is lower, only 7.5% of the transistor resistance, as shown by the voltage drop. The transistor width is 3.5μm compared to 13.2μm for the laser link, meaning the laser link is 3.5 times less resistive per unit width than the transistor. Secondly, the Laser Link does not saturate in the same way the transistor does, allowing a large current to be consumed by the cell. This structure is very useful in an FPGA design, because of the large number of cells to test. The approach is to incrementally add cells and check the power consumption. A map of defective cells can be produced this way. The cells can be accessed by a row-column circuit, or directly. The best approach depends on the number of cells and also on the number of pads that can be dedicated to the testing. Probe pads can also be used since there is no need to activate the test transistors after the link is zapped.

4.4.2 Clock

The clock signal, or any other signal distributed globally across the chip, must be dealt with care. The distribution of the clock signal on a wafer scale design was studied in many papers. The simplest approach for FPGAs is to use the H-tree architecture [26] to reduce the clock skew between the cells. This strategy is illustrated in Figure 4.21. The length of the clock line is the same for all the cells, thus reducing the clock skew.

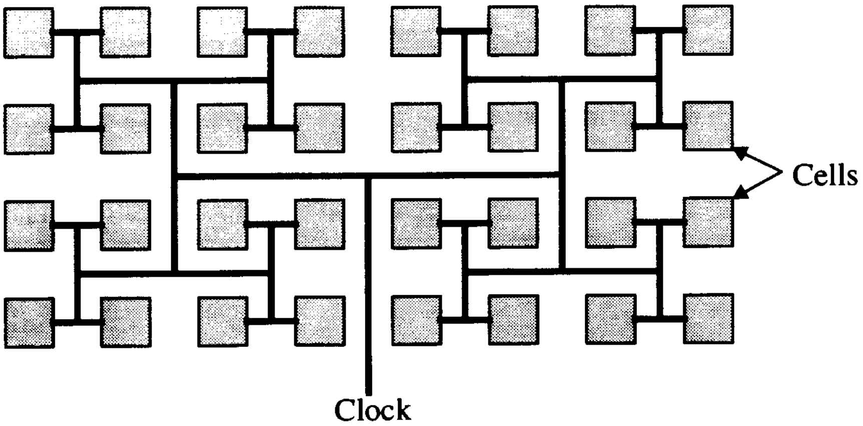


Figure 4.21 H-tree Clock Network

No tests were performed in this thesis to find the best clocking strategy. It depends on the size of the final circuit and on the maximum frequency at which the circuit can be used, depending on the delays between cells. The wafer scale FPGA clocking network could use the clocking strategies under research for wafer scale circuits [27][28].

The clock line must be redundant, however, because of the defects that can occur. The clock line is, like the power, a very critical issue in wafer scale design. The proposed method is to use a redundant clock line in each cell. With laser links, the signal can then be re-routed inside the cell and most of the defects can be avoided in this fashion. The low impedance of the laser link means the clock can be rerouted with very little additional delay.

4.4.3 Routing

As noticed in the previous section, the defect avoidance method chosen is the combination of cell by cell and row-column substitution. In this section are presented the different structures used and developed to restructure FPGAs.

First of all, a way to bypass entire columns of cells is needed. All the signals coming from the cell on the left must pass through the defective cell in order to reach the cell on the right. Figure 4.3 shows this clearly. The easiest way is to provide extra routing and laser link the signal to go through the cell. However, this method takes up a large area. By using the same routing structure as the Xilinx 4000 series[29], a new bypassing method was developed. In the Xilinx routing architecture, there are three different kinds of routing resources: the single length lines, the double length lines and the long lines. A simple routing switch is used which allows each signal to take either three directions (Figure 4.22). The single length lines go through one switch in each cell while the double length lines go through a switch only every other cell. The easiest way to bypass the signal through a dead cell is to permanently connect all the E-W and N-S connections of every switch in the cell.

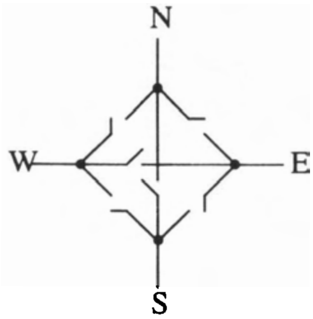


Figure 4.22 Routing Switch

To perform this, a laser link can be connected in parallel with the switch. Once the laser link is zapped, the signal can run freely in the cell. Instead of having two separate structures, a smaller version of the testable laser link is used. Its layout can be seen in Figure 4.23.

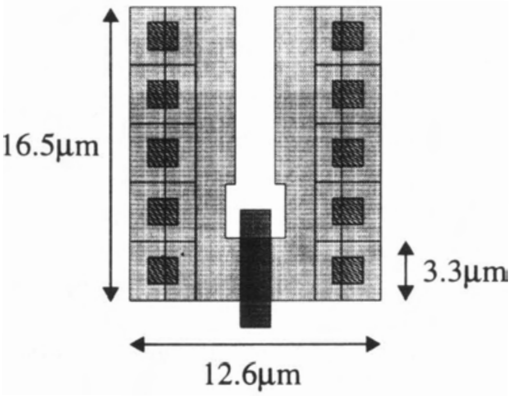


Figure 4.23 Laser Pass Transistor

By using this laser pass transistor for both the vertical and horizontal connections, and conventional N pass transistors for the other directions, the complete reconfigurable routing switch was designed. Its layout is shown in Figure 4.24. The switch box is made of one of these switches for each line in the channel. This switch has a double purpose: the transistors are used for FPGA routing only while the laser links are employed for defect avoidance.

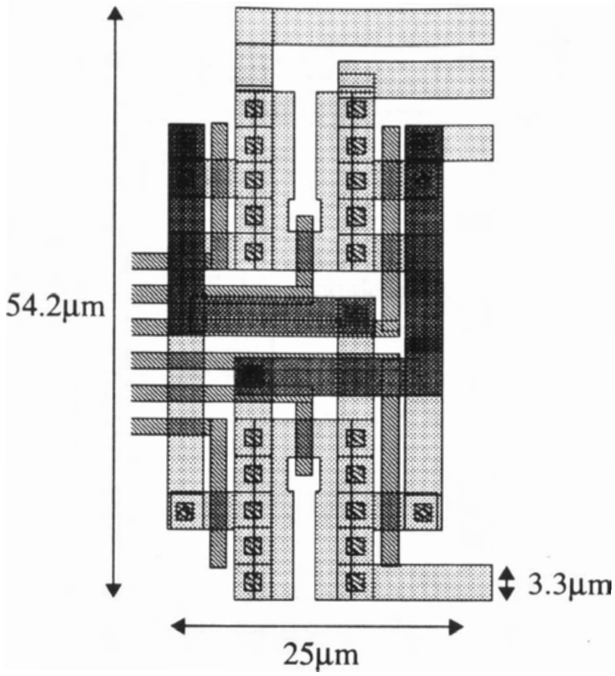


Figure 4.24 Reconfigurable Routing Switch

This method of bypassing is easy to use for single length lines. However, double length lines are trickier. The way to design them while keeping the same cell is to cross the lines inside the cell so that a routing switch is encountered only every other cell [30]. If a column is bypassed, the two logically adjacent cells will see their double lines disturbed: one line becomes a single length line while the other becomes a triple length line. This is unacceptable because the mapper would have to know which cells are bypassed. This problem can be seen in Figure 4.25. In a), there are four cells with the second being defective. In b), the restructuring is performed without uncrossing the lines. The formation of the triple length bypass line can be seen (wide line). The way developed to counter this problem is to uncross the lines in the bypassed cell by using two laser links. So proceeding, the double length lines are kept constant throughout the cell array. As shown in Figure 4.25 c), by uncrossing the lines in the defective cell, the double length lines are preserved.

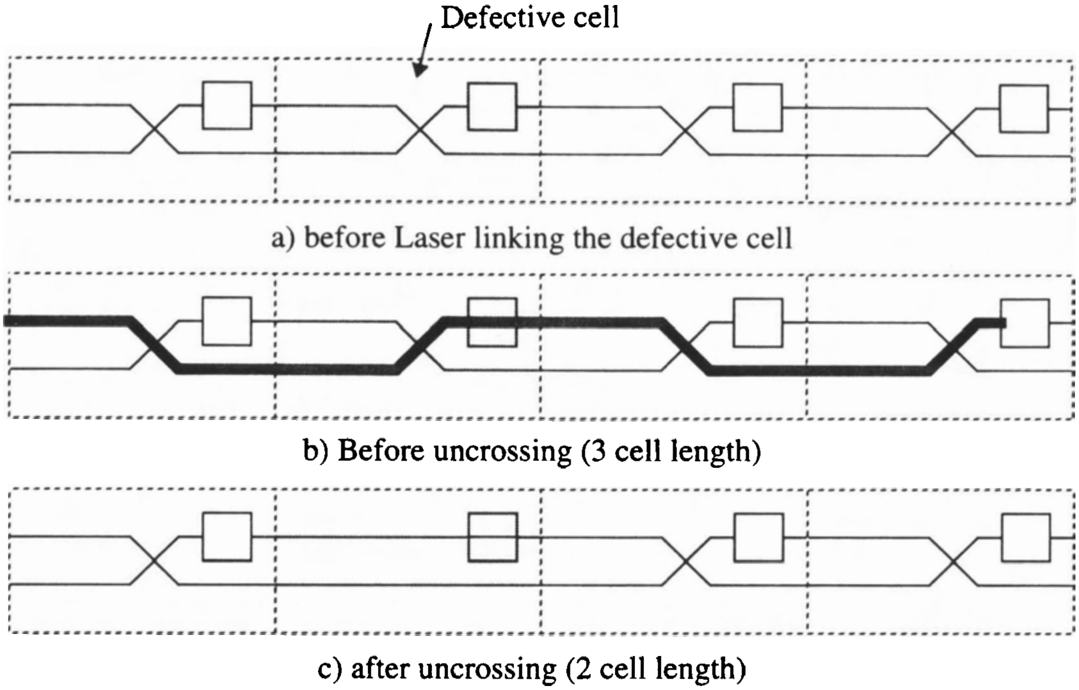

Defective cell
a) before Laser linking the defective cell
b) Before uncrossing (3 cell length)
c) after uncrossing (2 cell length)

Figure 4.25 Double Length Line Uncrossing Example

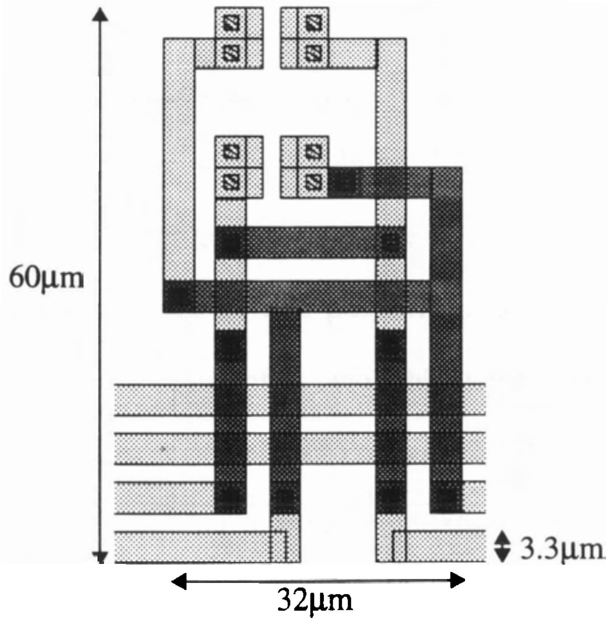


Figure 4.26 Laser Links Arrangement to Uncross the Lines

The laser link design to perform the line uncrossing is shown in Figure 4.26. Linking the laser links reestablishes the direct connection between the lines while laser cutting the original lines removes the line crossing.

With these switches, the column and row substitution is possible. In order to perform the cell by cell substitution, however, a restructuring bus is needed. As seen in the section about restructuring, one vertical restructuring bus is used. This allows cell by cell substitution in the lines while preserving the alignment in the columns.

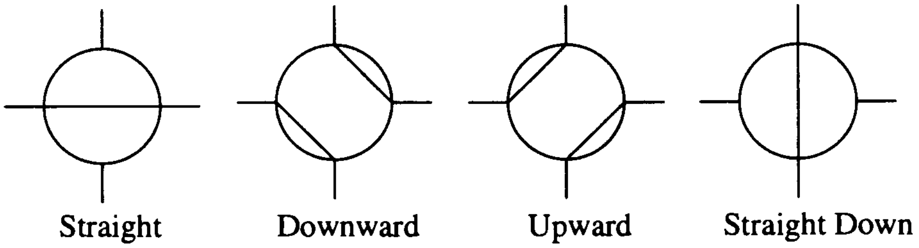


Figure 4.27 Possible Laser Switch Configurations

This restructuring bus must allow all the signals coming from the cell on the left column to connect to any cell on the adjacent column. For that purpose there must be switches allowing the signals to either go straight to the next cell, up or down. Thus the switch must be reconfigurable in one of the four possibilities shown in Figure 4.27 [31]. Each line in the channel must have a switch of its own. This switch arrangement, called the laser switch box, is placed on the right side of each cell, as shown in Figure 4.28. It is then possible to do the cell by cell restructuring, as the example shows in Figure 4.29.

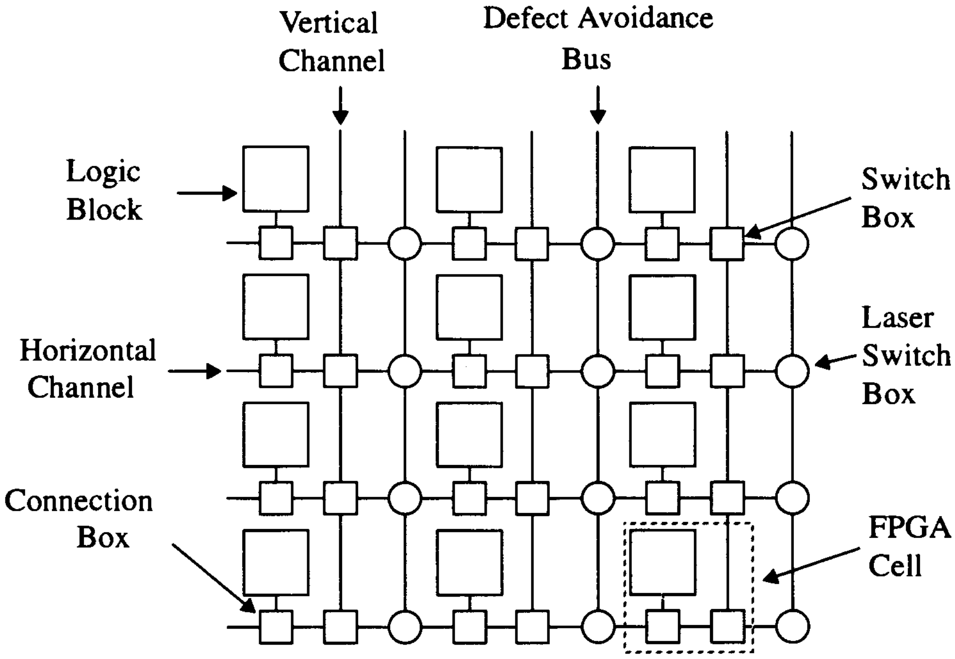


Figure 4.28 Physical Design

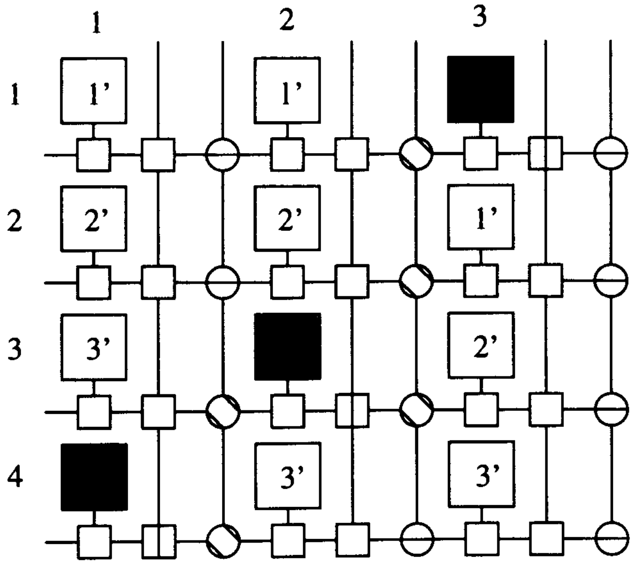


Figure 4.29 Example of Defect Avoidance (darker Logic Blocks are defective)

These switches are designed with laser links and no active switching because they are used exclusively for defect avoidance. By using laser linking and cutting, the switch configurations shown in Figure 4.27 can easily be achieved. The layout of the laser switch is shown in Figure 4.30 while its different configurations are shown in Figure 4.31.

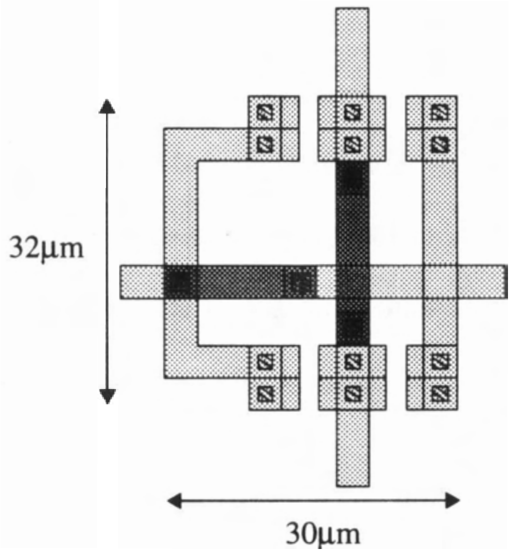


Figure 4.30 Laser Switch

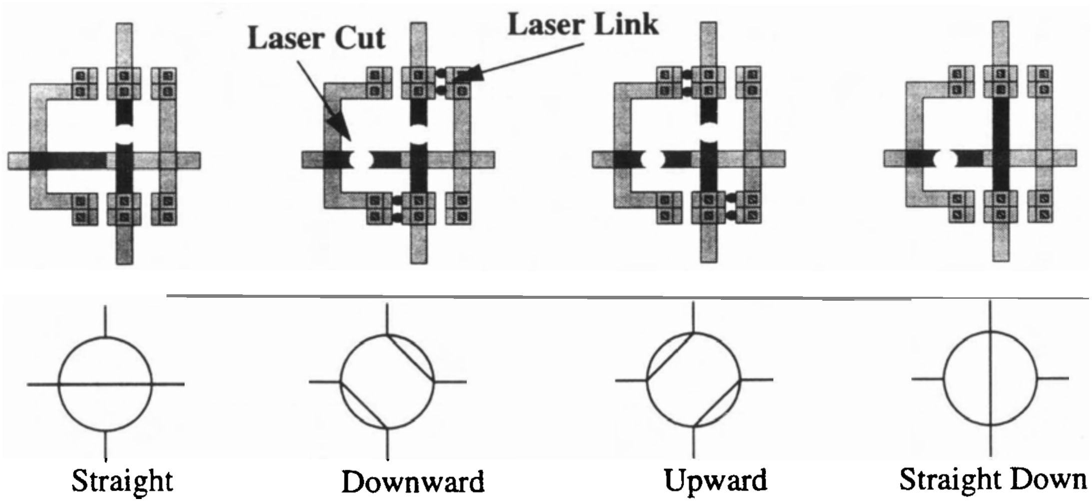


Figure 4.31 Possible Switch Configurations, with Linking and Cutting

4.4.4 Line Redundancy

Without line redundancy, the FPGA can still be restructured. As seen in the yield simulations, however, adding line redundancy increases the yield significantly and is therefore highly profitable. Line redundancy is achieved in this manner: an extra line runs in parallel with the routing channel. A laser link is placed between this line an all the other lines in the channel. This way, any line in the channel can be replaced using the extra line. The connections from the cell to the bus also have laser links to the extra line because, if one of these lines needs to be replaced, the connections have to be preserved. Depending on the number of lines, it is possible to use more than one extra line, each one being dedicated to a certain number of lines in the channel. This decreases the number of laser links on the extra line.

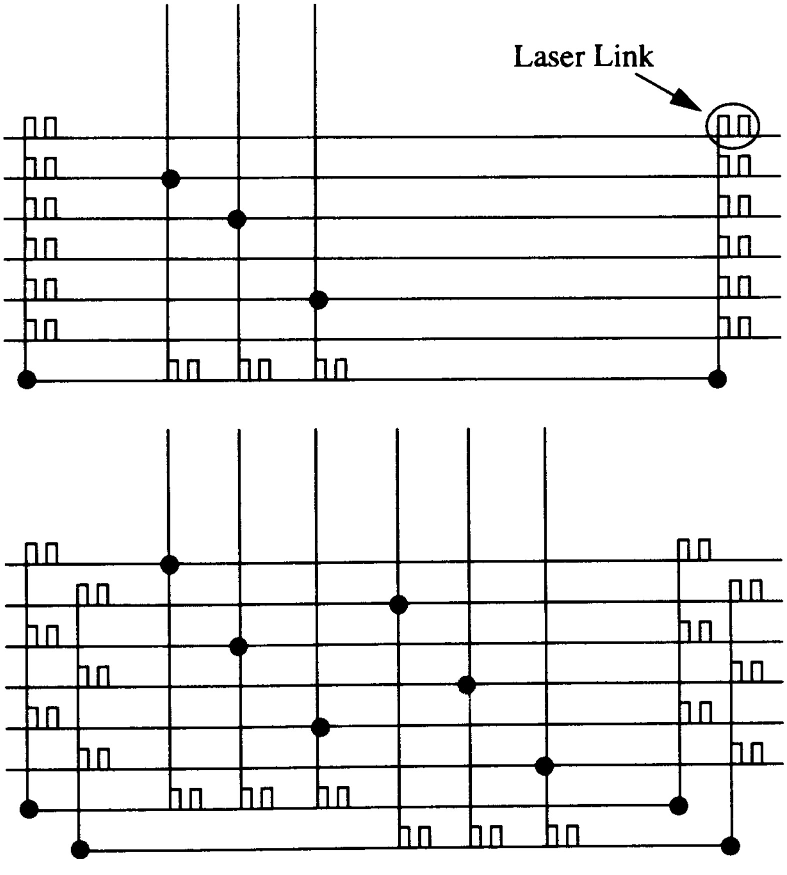


Figure 4.32 Line Redundancy. Top: one extra line; Bottom: two dedicated extra lines

4.4.5 Programming Circuit

The proposed design uses static RAM programming. A long shift register runs through the columns to program each cell. Since cell by cell substitution is used in the rows only, all the cells in one column have the same column index in the physical and logical array. So there is no problem in programming the cells with a shift register running in each physical column.

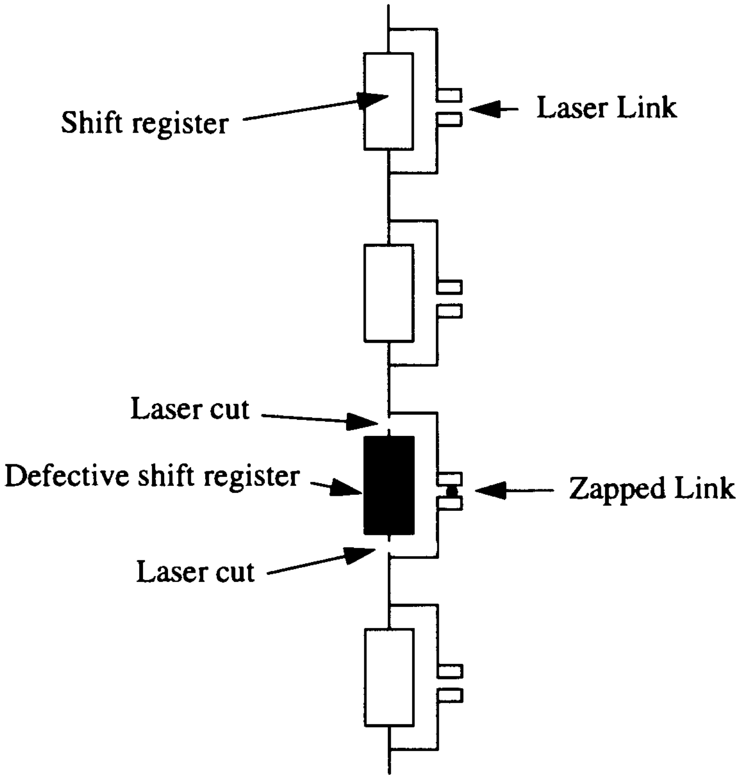


Figure 4.33 Shift Register Bypass

However, defective cells must have their shift register bypassed because the mapper will generate a bit pattern independent of the restructuring. Each cell contains a serial input and a serial output for its internal register. By using a bypass line that can be connected with a laser link, the bypass of the defective cell shift register is possible. The clock lines of the shift registers are also redundant in the same manner as the channels.

If a shift register is inoperable even with this kind of redundancy, the entire column can be bypassed.

4.4.6 Testing

The testing is an important part of any microelectronic circuit. The testing of the wafer scale FPGA has not been studied in detail. This section gives an overview of the critical aspect of testing that should be taken into account.

The testing of the wafer scale FPGA can be performed by using the same techniques available today to test the commercially available products. A reconfigurable design has however some special testing requirements:

First the power must be tested. This is done with the testable power link shown earlier. By accessing each cell individually, a map of defective cells is created. The cells presenting no problem are laser linked to the power rail and can be tested logically. The power will eventually be laser cut for the cells which are found defective afterwards, though that may not be necessary for some of those cells.

The programming circuitry must be tested up front; each output of the shift register must be accessible to test the shift register, because a defective shift register will propagate the wrong bit pattern to the cells located after the defective cell. This access can be done in a row column access, as in the case of the power test.

Each cell must be separately testable for its logic functioning; this may be done by testing an entire column at a time, with the same vectors to each cell, rejecting those who produce different results. Built-in self test (BIST) can be added to complex cells to aid in the testing phase.

Checking for shorts and open circuits is also important. The restructuring buses run through the whole chip; they are therefore easily testable. Programming the cells to perform different tests on the routing architecture is also a possibility.

Testing of a wafer scale design is a complex task that extend beyond the scope of this thesis. However, with small modifications to the techniques already employed, the testing should not cause major problems.

4.5 Software Overview

Even if the object of the thesis is to study the physical aspects of a wafer scale FPGA design, software cannot be overlooked because it is an essential part of an FPGA design. This section explains the critical aspects for the software requirements to reconfigure and run a wafer scale FPGA.

4.5.1 Restructuring Software

Once the testing has produced a map of defective cells, the circuit has to be restructured. The defect avoidance consists only in physical restructuring, so there is no need to program switches. Instead, the laser links and cuts must be performed. There are many links and cuts to perform in each cell; however, those links and cuts are the same from cell to cell for the most part of the restructuring. So it is easy to use a batch file to perform the task. A separate program must be run to restructure around the defects of the channels, because the channels needing repairs vary from cell to cell. But once again, only a limited number of coordinates are needed for each cell. Also it is easy to use the batch linking. The configuration time can be reduced by aligning the laser links so they can be zapped with limited movement of the laser table. This is the task of the designer to align the links accordingly. The best way to create the batch file is to use the CAD tool and create two additional layers: one for the laser link and one for the cut. By using these new layers, the designer is able to simulate the effect of the laser restructuring. These layers were created in the Cadence environment. The properties of the link layer establishes connectivity between the active regions of the link; for simulations, the resistivity of the link, extracted from the technology, can be added to the properties.

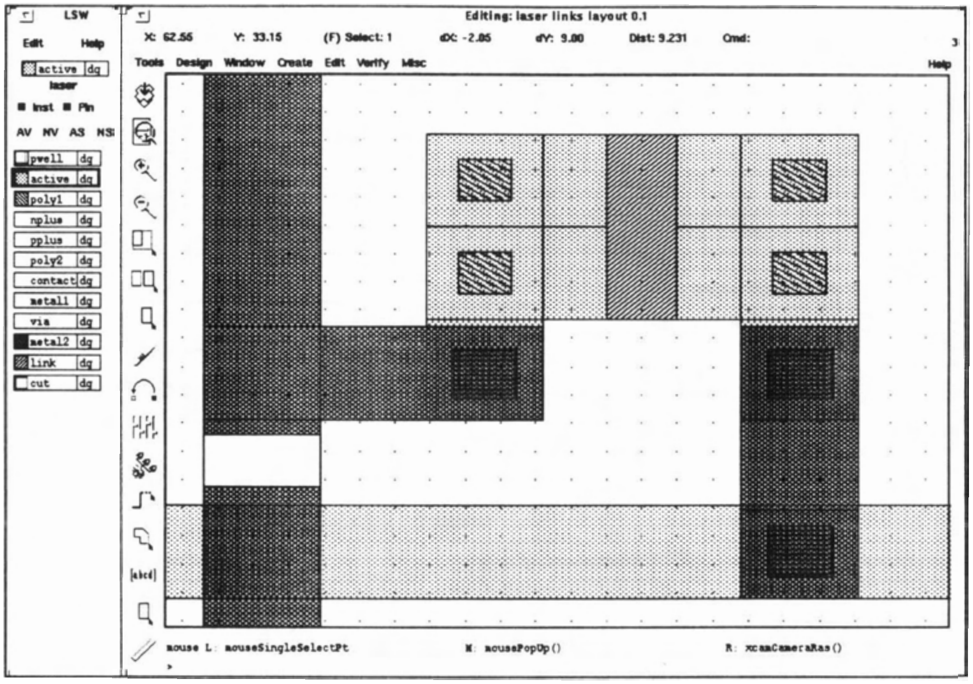
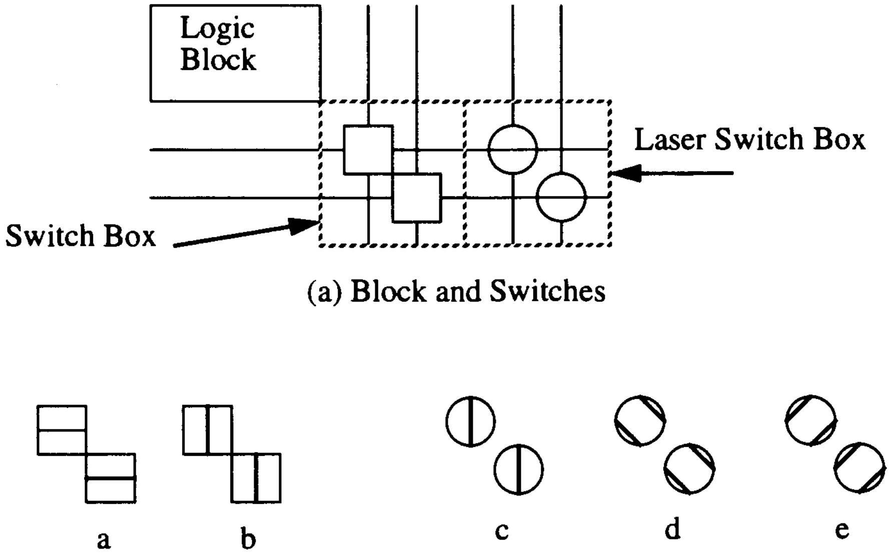


Figure 4.34 Implementation of the Link and Cut Layers in Cadence

The cut layer simply consists in breaking the connectivity in a metal line. It allows the designer to test for connectivity and also simulate the performance of the design with the laser links and cuts included in the design. It is also very easy to extract the information about the coordinates of the links and cuts, since they are separate layers. Thus there is no problem in integrating the laser links and cuts into already existing design tools. A library of restructured cells can be designed and the appropriate linking and cutting pattern chosen for each cell in the array. There is a limited number of rerouting patterns for a cell; all the laser switches in the cell have to be rerouted in one of the four possibilities shown in Figure 4.31, while the switch box has two rerouting possibilities, either horizontal or vertical. This is shown schematically in Figure 4.35: all the switches in the switch box can be laser linked in the a or b fashion and all the laser switches in the laser switch box can be laser linked in the c, d or e fashion (the channel contains only two lines for clarity). Coordinates of the links and cuts can be referred to the corner of the cell and are easily transformed.



(a) Block and Switches

(b) Possible Laser Link Restructuring Patterns

Figure 4.35 Restructuring Patterns

The amount of rework is dependant on the size of the final product. For small restructurable arrays, where only column bypass is considered, the bypass of the cells is simple and fast. For complete wafer scale systems, the process is longer because testing and restructuring is iterative. Auto routing software [32] are necessary to route very complex circuit. Such a software could be used to generate the laser link routing map.

4.5.2 Programming Software

As seen in section 2.2.4, there are six basic steps to create a circuit on an FPGA. Since the wafer scale FPGA proposed in this thesis is based on the same kind of basic cells found in commercially available circuits, there is no major differences in the programming software. Small restructured FPGAs could be used like any other FPGA and depending on their design, they could even be programmed with existing software. The complete wafer scale circuits will require special software: the design entry and optimization are still done in the same manner, only the software used must be able to handle large designs. High level capture is better suited for large designs. Placement and routing software will require research but will not differ a lot from actual software used to program prototype boards and arrays of FPGAs. The programming of the shift register requires a larger memory capacity.

Since restructuring is invisible to user and the software, the wafer scale FPGA can be considered as a larger version of a standard FPGA. A library of macro functions could be built and optimized, with complete circuits already available. The designer could chose from these circuits and implement a complete wafer scale system in a short period of time.

4.6 Summary

This chapter has dealt with the different aspects of the defect avoidance in FPGAs. The major emphasis has been made on the restructuring aspects and physical design of the defect avoidance structure. It has been shown that acceptable yields can be achieved by using appropriate methods. The required software has been briefly introduced and left as future work in the realization of wafer scale FPGAs.

Chapter 5

The Test Vehicle

This chapter presents the design and experimental work done to test the concepts presented in the previous chapter. The first section will deal with the design of the test vehicle and its different parts. The second section will present the results on power while the third section will explain the delay simulations. Finally, the last section will set out the different experimental results performed on the chips.

5.1 Design

The idea behind a test vehicle is to provide means to examine the aspects of wafer scale systems on a chip which can be produced within the Canadian Microelectronic Corporation multi-project wafer system. This section shows the design of the wafer scale FPGA test vehicle fabricated to test the techniques presented in Chapter 4. The design was done on Cadence with the Mitel 1.5μm CMOS technology.

5.1.1 Architecture

The first step in designing an FPGA is to chose the architecture to employ. The symmetrical architecture is best suited because of the restructuring technique chosen. In this architecture, a square array of similar logic blocks is surrounded by routing resources. To make the design restructurable, a restructuring bus is added between each column of cells. The block diagram is shown in Figure 5.1

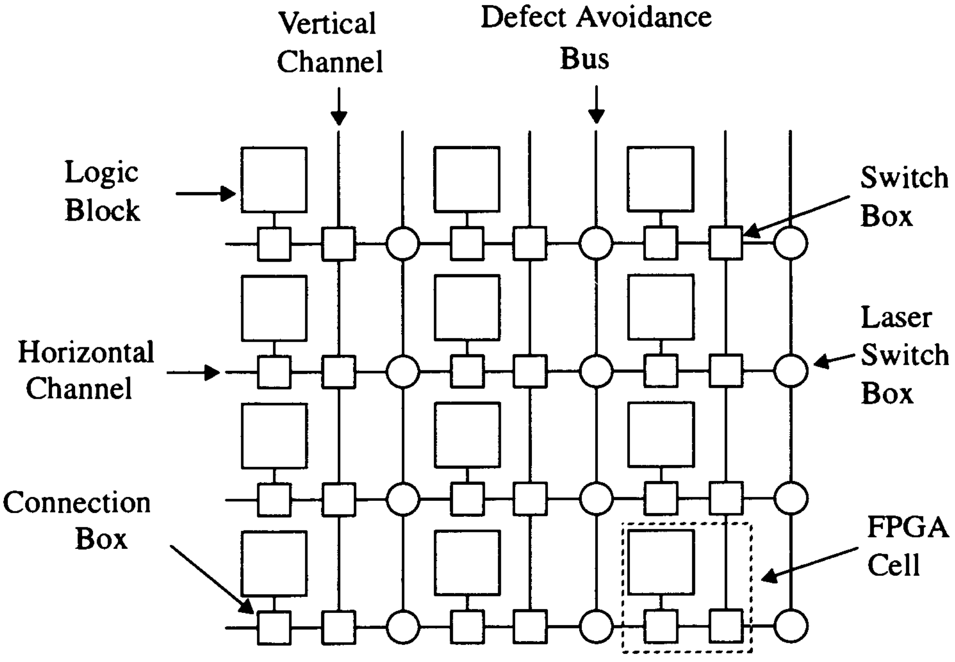


Figure 5.1 Symmetrical Restructurable Architecture

This is the basic architecture used as a starting point to design the test vehicle. The next subsections will describe the different parts of the cell in detail.

5.1.2 FPGA Programming Technology

As noted in Chapter 2, the most widely employed programming technologies are static RAMs, anti-fuses and EEPROMs [34]. Since only CMOS technology was available, the EEPROM or anti-fuse programming could not be used. Since static RAM programming is very popular in actual FPGAs, and is easily programmable with our testing equipment, it was chosen as the programming technology for the test vehicle.

A long shift register is run through the FPGA cells. Each bit in the shift register accomplishes a function, like activating a switch. The basic cell must be very simple and occupy very little area, because of the large number of programming bits required. A double non-overlapping clock shift-register was designed. The design was not optimized for area, but rather to ensure proper functioning. Two inverters in a SR latch mode with pass transistors were used. The pass transistors allow minimum size inverters. The schematic of the circuit is shown in Figure 5.2.

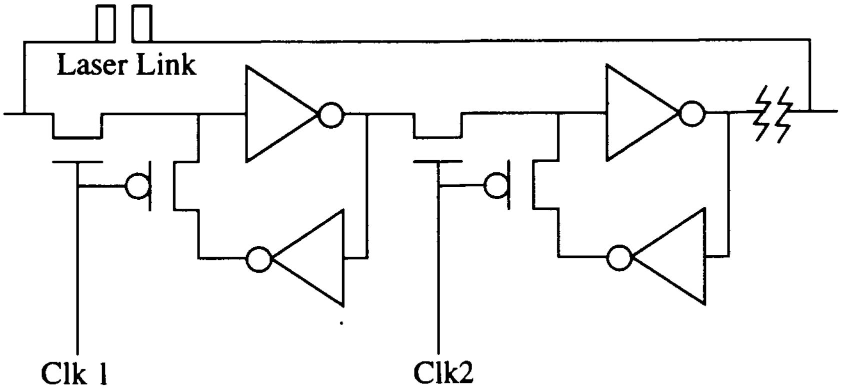


Figure 5.2 Schematic of the Shift Register Bit Cell

As explained in section 4.4.5, a laser link was placed between the input and the output of the shift register in each cell to bypass the cell in case of a defect occurring in the circuit.

5.1.3 Logic Block

Considering the silicon space available is limited and because the logic block does not need any reconfiguration, a very basic circuit was employed. As seen, the logic block is used to implement logic functions. This can be done in different ways: Look-Up tables, multiplexers or simple logic gates. A Look-up table based logic block was chosen, because it is easy to implement, requires little area and is also commonly used in currently available FPGAs [2]. The results obtained with a small look-up table can easily serve for a more complex but similar design. In order to test the sequential circuits, the logic block also includes a D flip-flop. The output of the Look-up table is either transferred directly to the output of the cell or run through the D flip flop. The number of inputs and outputs was also kept low: three inputs for the look-up table, one clock input for the flip-flop and one enable input for the cell and one buffered output. The block diagram of the logic block is shown in Figure 5.3. The schematic for the look-up table is shown in Figure 5.4.

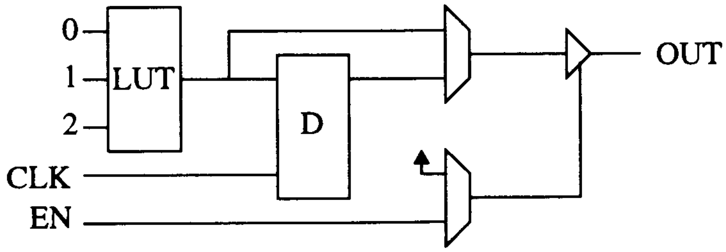


Figure 5.3 Logic Block (LUT: Look Up Table; D: D Flip-flop)

5.1.4 Connection Box

The connection box serves to connect the inputs and outputs of the logic block to the routing channels. The design is very simple: a pass transistor activated by a bit of the shift register allows the connection. In this way, connection to no line (hi-Z), one or many lines in the channel is possible.

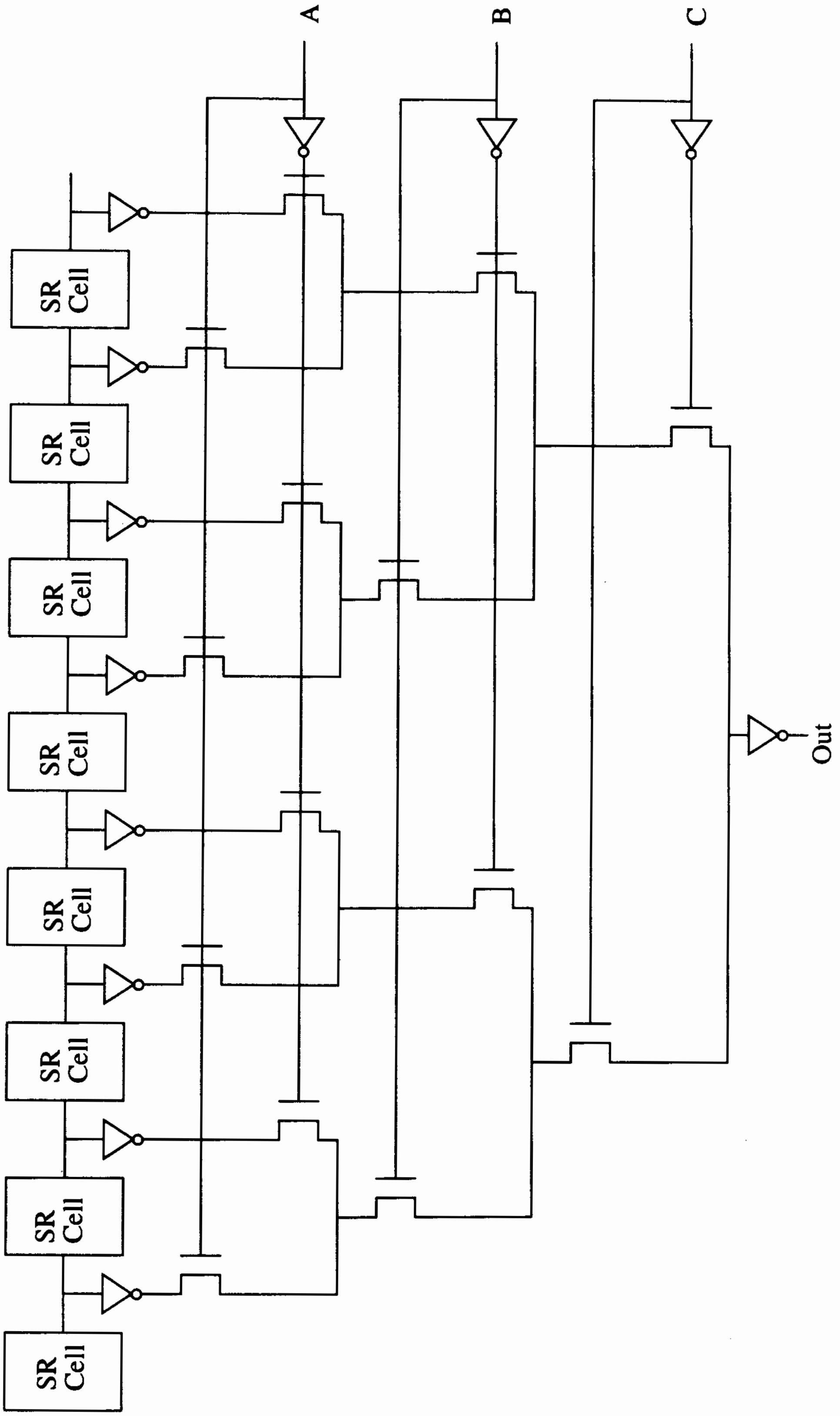


Figure 5.4 Look-up Table Schematic

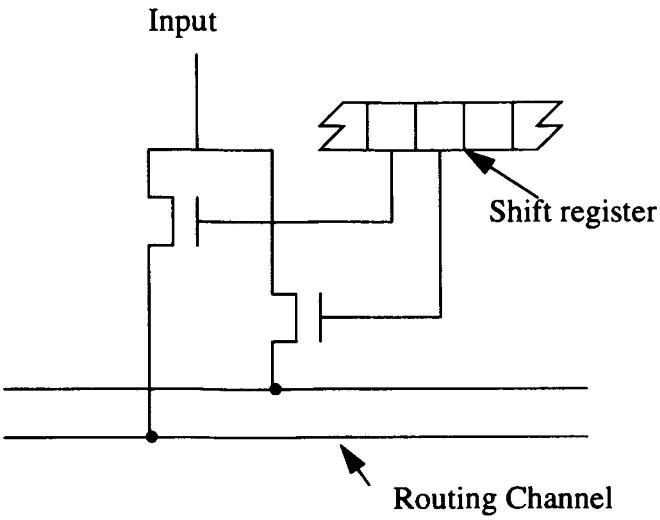


Figure 5.5 Connection Box Diagram

The number of lines to which each input and output can be connected directly influences the flexibility of the FPGA [2], but also increases the length of the shift register. The number was set to six, to achieve a certain flexibility while keeping the design small.

5.1.5 Routing

Since the symmetrical architecture was chosen, routing channels must be placed vertically and horizontally between each cell. The number of lines in each channel is critical for the flexibility of the routing. An important aspect to test with the vehicle is the utilization of single and double length lines; both were included in the routing channels. Once again, area considerations made us choose a small number of lines, 12 in total: 6 single length, 4 double length and 2 for the clock. The double length lines include the uncrossing option for the bypass of the cells.

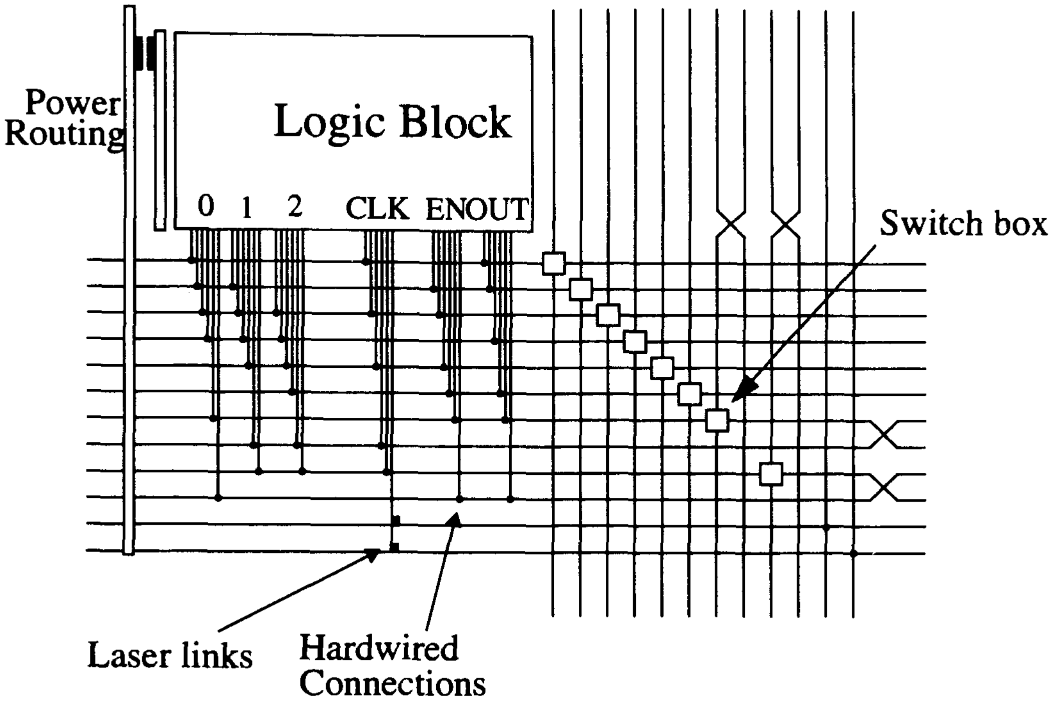


Figure 5.6 Block Diagram of the FPGA Cell

It is a small number compared to commercial FPGAs but sufficient to perform the tests and demonstrate all functional operations of an FPGA cell. The switch matrix, which makes the connections between the channels, uses 8 switches similar to those described in section 4.4.3; they allow the connection to the three opposite lines and can be laser linked to bypass the cell. In the test vehicle fabricated, there is no redundancy in the channels. The block diagram of the FPGA cell is shown in Figure 5.6. A new circuit including line redundancy was designed and submitted for fabrication.

5.1.6 Chip Layout

Two different chips were designed and fabricated in the Mitel 1.5μm CMOS technology. The first one utilizes the cell described in the previous section. The layout of this cell can be seen in Figure 5.7.

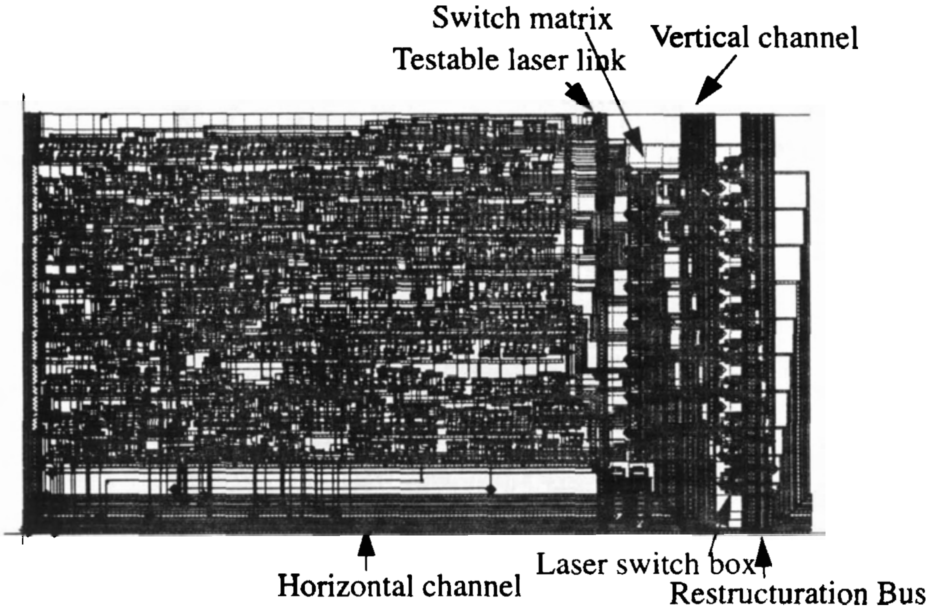


Figure 5.7 Circuit Layout of the FPGA Cell in Mitel 1.5µm (1206µm x 650µm)

Because of its large dimensions, it was impossible to build an array of such cells with the standard chip dimensions (3.1mm x 3.1mm) available from Mitel. With special arrangement with CMC, it was however possible to take four adjacent tiles of those chips. By using half the width (leaving space for other designs), the fabrication of a 1.5cmx1.5mm chip (ICBSFCD4) was possible. This chip includes a row of 12 cells with an additional row of restructuration buses to bypass defective cells. In order to test the design with a real array, another version of the cell, with smaller dimensions, was designed. All the elements of redundancy found in the larger cell are present and only the width of the channels and the size of the logic differ. Two single length lines and two double length lines were used. The look up table has two inputs. The logic block has only three connections to the channels, two inputs and one output. The layout of this cell can be seen in Figure 5.8. The large chip (ICBSFCD4) layout is shown in Figure 5.9 while the small chip (ICBSFCD3) layout is shown in Figure 5.9.

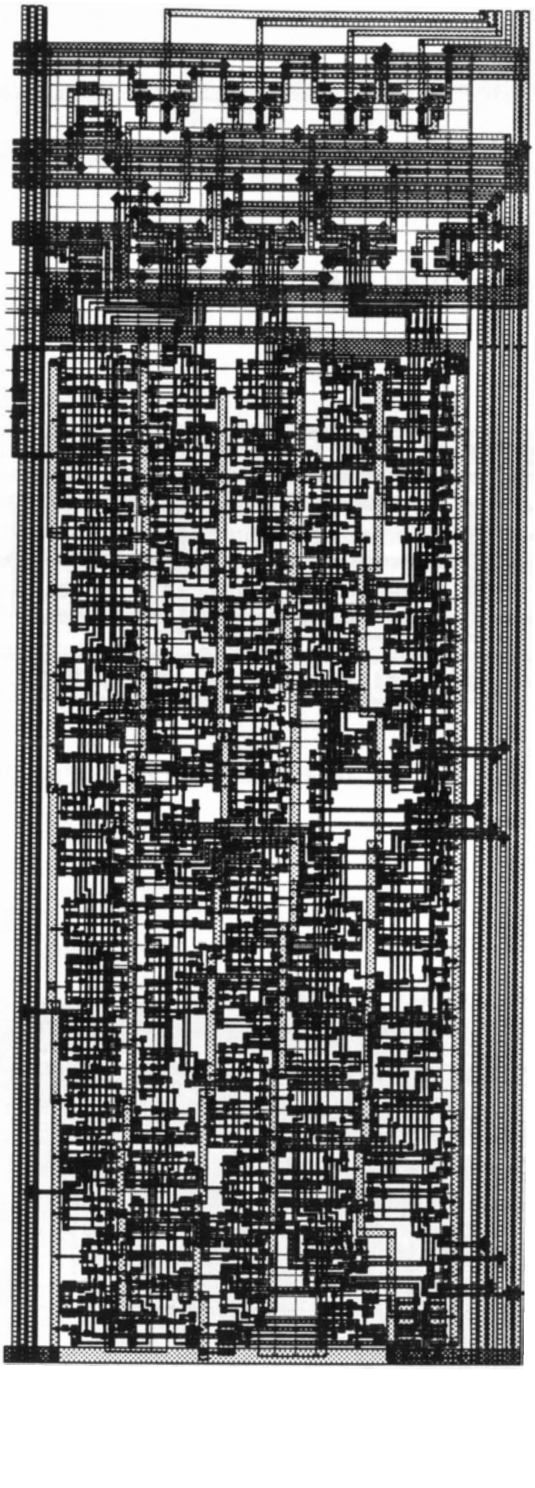


Figure 5.8 Layout of the Smaller Cell in Mitel 1.5μm (834μm x 333μm)

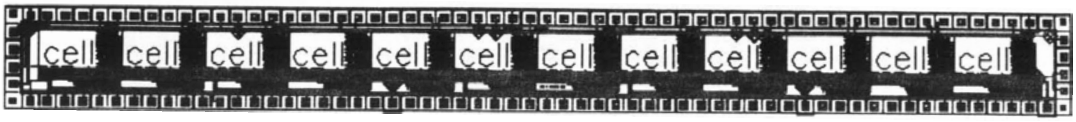


Figure 5.9 Circuit Layout of the Large Chip (ICBSFCD4) 1.5cm x 1.5mm

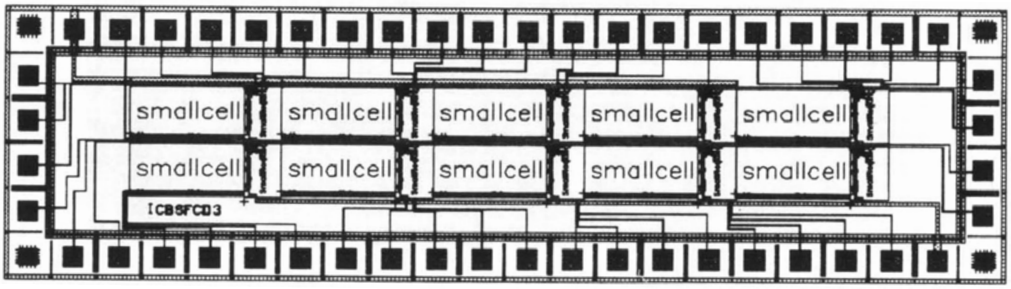


Figure 5.10 Circuit Layout of the Small Chip (ICBSFCD3) 6.2mm x 1.5mm

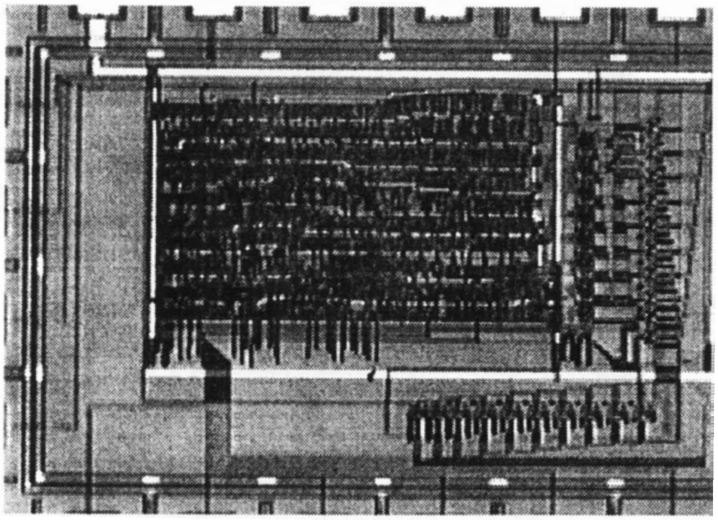


Figure 5.11 Photograph of the Large Cell Layout (1206μm x 650μm)

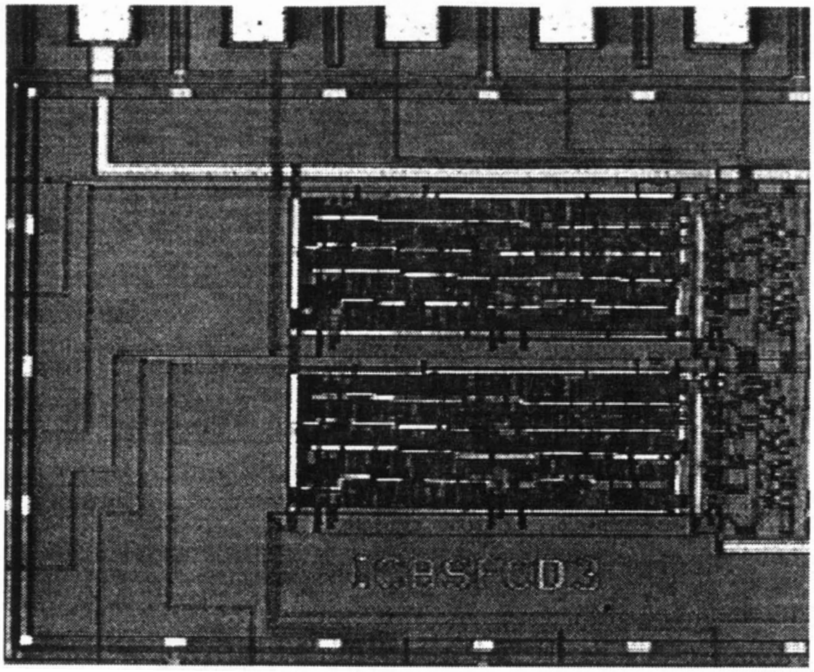


Figure 5.12 Photograph of the Small Cell Layout (834μm x 333μm)

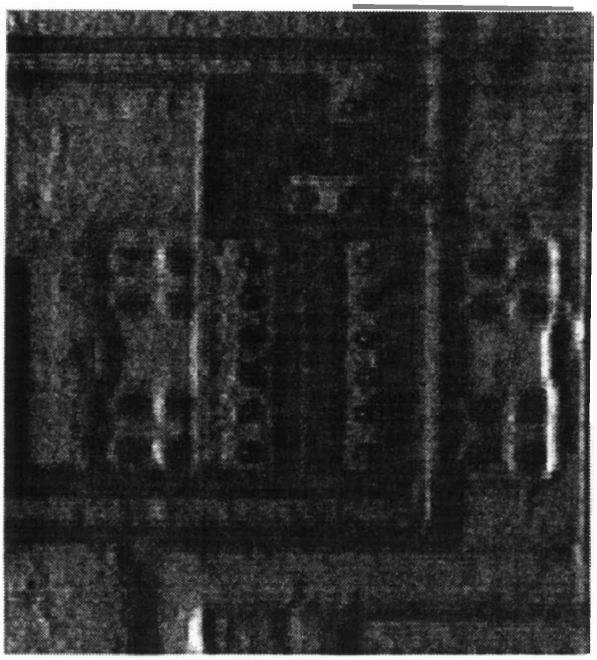


Figure 5.13 Power Testable Link Photograph (45μm x 23μm)

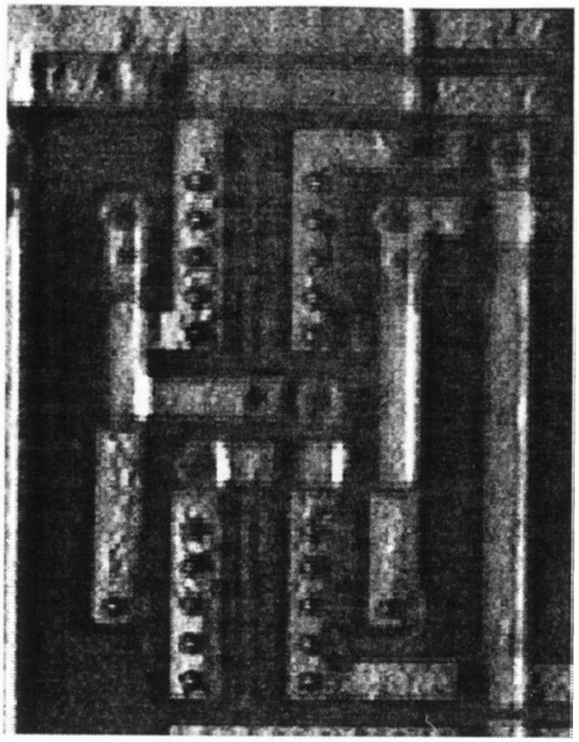


Figure 5.14 Reconfigurable Switch Photograph (54.2μm x 25μm)

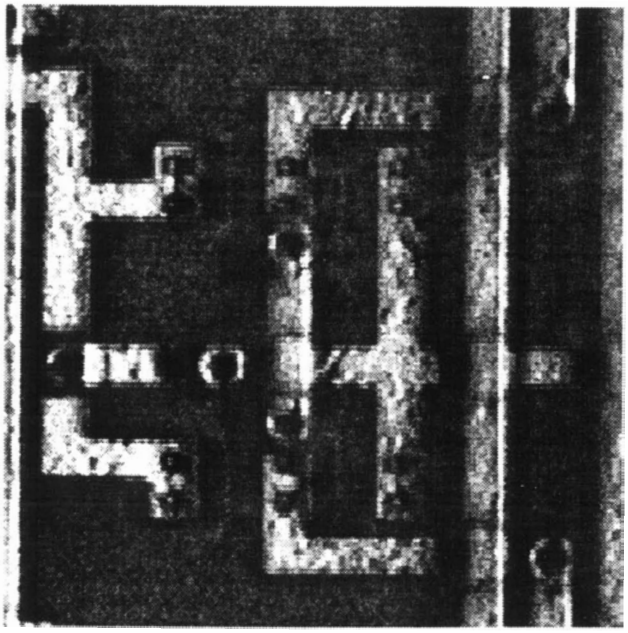


Figure 5.15 Laser Switch Photograph (32μm x 30μm)

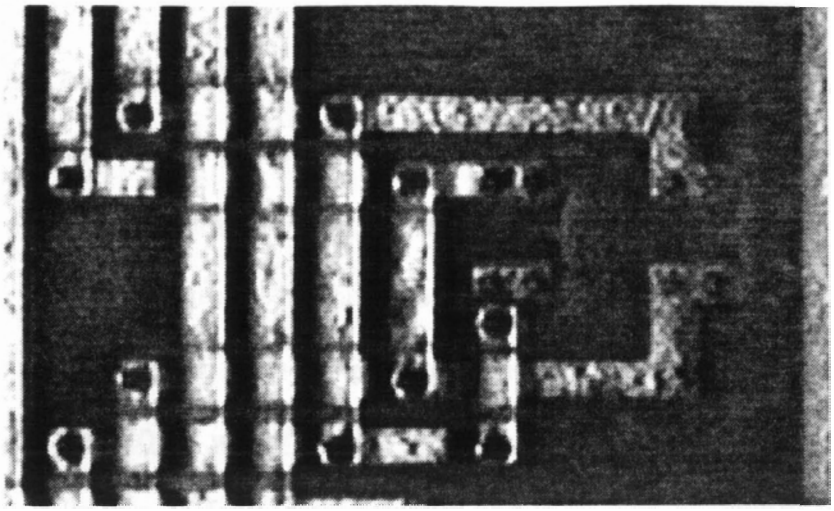


Figure 5.16 Line Uncrossing Structure Photograph (60μm x 32μm)

Figure 5.11 through Figure 5.16 are photographs of the layouts as well as the different defect avoidance structures.

5.2 Power

The first step in testing the device is to check power shorts in the cells. Each testable power link is turned on and the input current measured. Unfortunately, in the test vehicles, the P-Well of the testable link was left floating, making the current measurement difficult. The current consumption is however low when all the transistors are turned on, indicating no power shorts in all the chips tested. The input current for the small chips was in the order of 360±20μA while running at 1kHz and 440±20μA at 1MHz. The laser linking of the cells was performed successfully and there was no change in behavior or power consumption.

An earlier version of the chip (ICBSFCD1) had its P-Wells properly tied. The results for the power consumption of those chips are shown in table 5.1. For no cells connected, the input current was 7.6±0.1mA; for all cells connected, 12.9±0.1mA.

Input Current(mA)	Row1	Row2	Row3
Column1	9.0±0.1	8.4±0.1	8.4±0.1
Column2	9.0±0.1	8.4±0.1	8.5±0.1
Column3	9.8±0.1	9.5±0.1	9.4±0.1

Table 5.1: Power Test Results

The power consumption is high in the unconnected mode because of a design error in the pads, but it can be seen that the current is increased approximately the same amount for each cell tested. These tests were performed on different chips, however the design error caused the power to vary from chip to chip. The same behavior was observed in every chip, each cell showing about the same current increment when turned on.

5.3 Delay

The major drawback of an FPGA is circuit speed. For a given circuit, the custom implementation is much faster than the FPGA because of the large delays in the routing circuitry. Some papers have dealt with the optimization of the logic block complexity, the cell granularity and the different architectures to optimize the speed of FPGAs [33][35]. In this section, the results for defect free wafer scale FPGAs of the same complexity, but without the redundancy or defect avoidance, are compared with the architecture used, taking into account the defect avoidance overhead. Note the defect free Wafer Scale FPGA would have a negligible yield and are considered only as the idealized comparison target, with a speed similar to current standard FPGAs.

5.3.1 Delay Approximation

In [33], the total delay (D_{tot}) of the critical path in a defect free FPGA is approximated as follows:

$$D_{tot} = N_L \cdot (D_{LB} + D_R) \qquad (5.1)$$

where N_L is the number of logic blocks in the critical path, D_{LB} the delay of the Logic Block and D_R the delay of the routing between two blocks. The delay of the logic block can be easily calculated but the delay of the routing is much more difficult to approximate. It depends on a large number of factors, like the fanout and the length of the connections [33]. A calculated value is used to give an idea of the delay but the reader should keep in mind that this value can vary a lot, even just by remapping the circuit.

For the wafer scale circuit, new delays must be included in the calculation of the total delay D_{tot} because of the extra routing and the physical restructuring of the array.

There are two extra delays: D_{OH}, the delay of the overhead restructuring circuitry in each cell and D_{REC}, the delay of a restructuring channel. The total delay (D_{tot}) of the critical path becomes:

$$D_{tot} = N_L \cdot (D_{LB} + D_R + D_{OH}) + N_R \cdot D_{REC} \tag{5.2}$$

where N_R is the number of restructuring channels in the path. This number is hard to evaluate because it depends on the restructuring algorithm. Before doing any calculations, the delay of each section of the design must be evaluated with Hspice.

5.3.2 Delay Simulations

The following simulations were done with the dimensions of the small chip (ICBSFCD3). The Hspice model in the following sections is changed when the large chip (ICBSFCD4) is used.

The delay of the logic block is simulated as follows (the circuit is shown in Figure 5.17): the input comes from one channel line (In), goes through a pass transistor(1), then to the input of the look up table. In the worst case, the input is connected to two transistor gates and also goes through one inverter and is then connected to two other gates. The signal is then transferred from the cell bit in the shift register, through an inverter, two pass transistors (3,5) and two transmission gates (6, 7). This is connected to a large buffer (8) which goes through a pass transistor and is then connected to the output channel line (out).

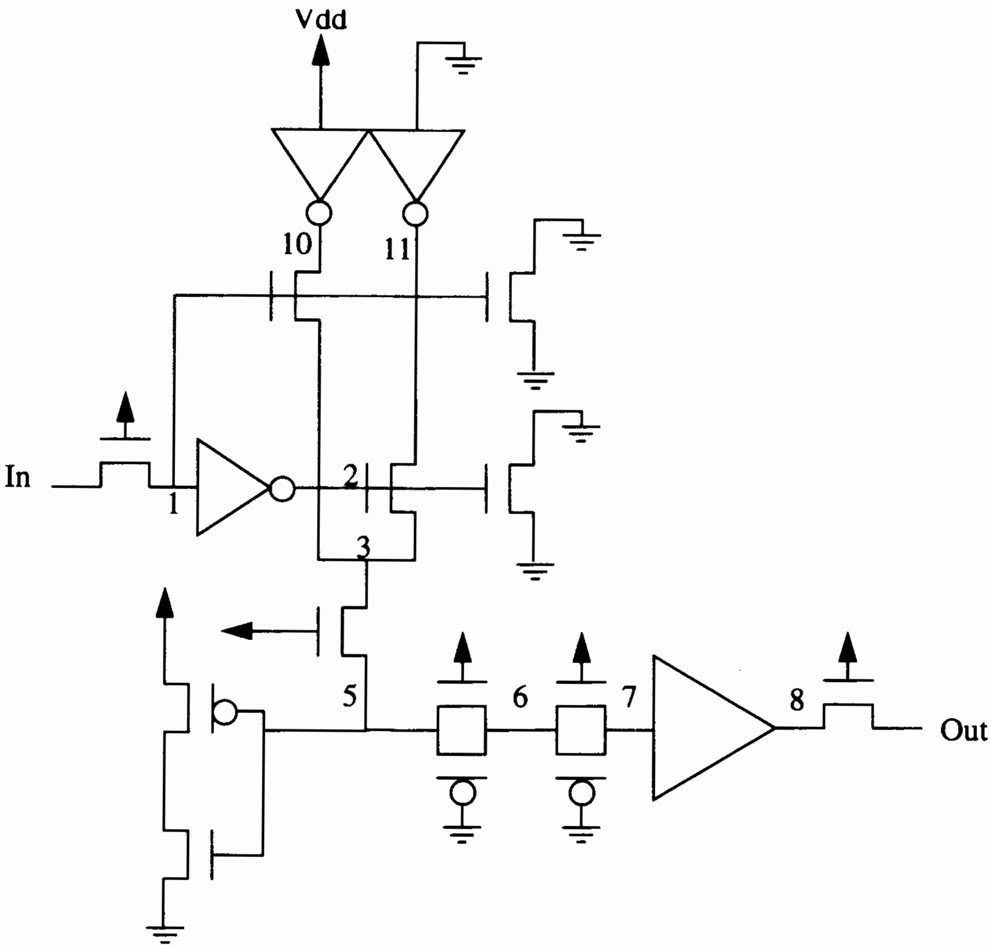


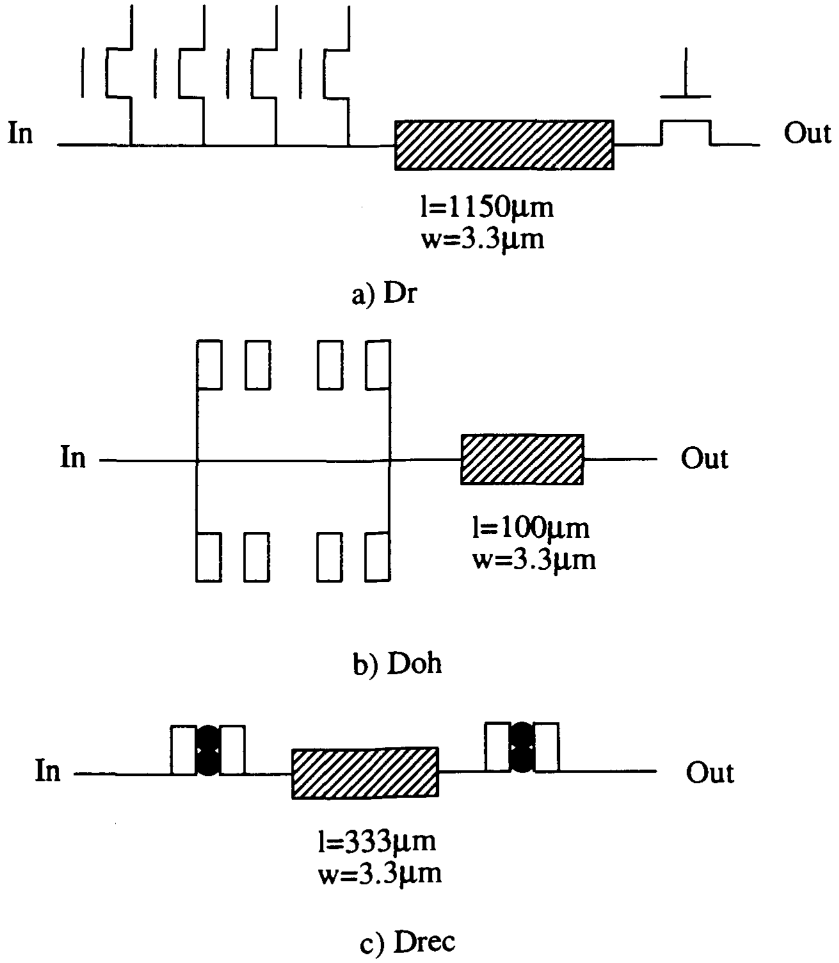
Figure 5.17 Logic Block Delay Circuit (the numbers are Hspice nodes)

To simulate the routing delay, a signal coming out of the logic block and going to a routing channel is assumed. The capacitive load of a line in the channel is composed of the metal line itself, simulated by a 1150μm long by 3.3μm wide RC line in Hspice. This line goes through a pass transistor. Other capacitive loads are simulated on the line with connections to the drains of open transistors. They represent the connections of the line to the connection box. The overhead delay, composed of the restructuring circuitry, is a simulation of the laser switch box with a 100μm long and 3.3μm wide metal line. The switch box includes 4 laser links. The last delay simulation is for the restructuring delay, Drec. This is approximated by a signal going through a laser link (modeled at 108Ω), a metal line of 550μm in length and 3.3μm in width, and another laser link.The delay model in Hspice for the metal lines is as follows:

.MODEL linemetal1 R COX=0.00014 RSH=0.04 CAPSW=2.3E-10

.MODEL linemetal2 R COX=0.00012 RSH=0.04 CAPSW=2.3E-10

where COX=area capacitance (F/m^2), RSH=sheet resistivity (Ω/□) and CAPSW=edge capacitance (F/m). These simulations are of course rough approximations of the real delays in the cell, but they give a good idea of the delays introduced by the laser restructuring.



a) Dr

b) Doh

c) Drec

Figure 5.18 Circuits used for Delay Simulations: a) Dr, routing delay; b) Doh, overhead delay; c) Drec, restructuring delay

The results for each delay simulated is shown in table 5.2.

Delay	ns
Logic Block D_{LB}	6.7
Routing D_R	9.6
Overhead D_{OH}	0.6
Restructuring D_{REC}	3.0

Table 5.2: Simulated Delays

As an example, the delay for a worst path of ten cells is calculated. The results are given in Figure 5.19

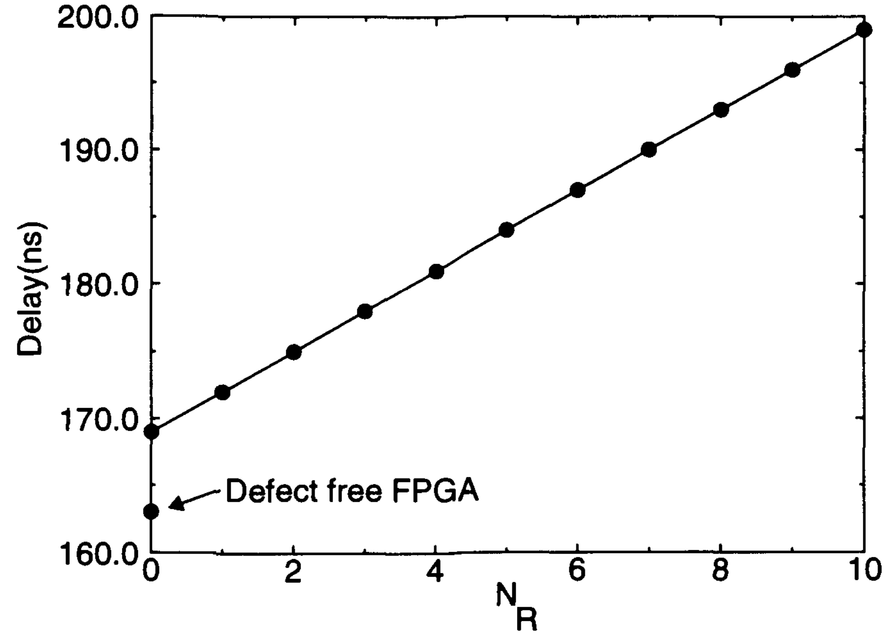


Figure 5.19 Graph of the Delay vs. Yield for a Row of ten Working Cells

The graph shows that for high yield (low N_R), the difference between the wafer scale FPGA and the defect free FPGA of the same complexity is small. The delay is increasing linearly from 4% (N_R=0) to 22% (N_R=10). To achieve good performance, the restructuring algorithm must give a number of channels N_R as low as possible.

In this example, the delay for a logic block composed of a 2 input look-up table is measured. Larger blocks show a larger delay but require less routing. A very large block is useless in standard VLSI because not enough blocks can be placed on a single chip. But for wafer scale, the choice of a larger block may improve the performance.

The Wafer Scale FPGA has a lower delay than the prototype boards composed of arrays of commercial FPGAs because of the high impedance of the board connections and the extra delay caused by the routing chips [3].

5.4 Delay Experiments

The purpose of the test vehicle was not to build and test a high performance FPGA architecture. The basic cell is very simple and not optimized. The goal was to measure the effect of the laser restructuring on the FPGAs, and the delays imposed by the overhead.

A series of experiment were conducted to measure delays in non-restructured FPGAs (with no laser links) and in FPGAs with restructured channels. The design offered limited flexibility but different paths and configurations were tested.

5.4.1 Routing Delay

The first experiment is to test the small cell FPGA (ICBSFCD3) to see if the logic is functioning properly. Because the test vehicle is a custom design, there is no software available to work with. Software was written to program the shift registers in the test FPGA: two non-overlapping clock signals and two programming bit patterns used as inputs for the shift registers were generated using a National Instrument [37] data acquisition board. Four bits of an output port were used for this purpose. A simple text file composed of the bit pattern to be recorded is needed to program the test vehicle. This method of programming is tedious but sufficient for the size of the test vehicle used.

Proper functioning of the shift registers can be checked by looking at the output (which was made accessible via a pad). On the large chip, the output of the shift register of each cell is accessible, so it is easy to tell which one is defective and bypass it. The smaller chip only have the last output available. Tests were done on three small chips. The first one had one shift register working and the other was defective. The second and third had both their shift registers working fine. An attempt to correct the defective shift register was made, but since it was impossible to tell which cell was defective (because of the unsufficient I/O pads number), it had not been conclusive. On the large chips (ICBSFCD4), the restructuring of the shift register was performed successfully.

After the shift registers were tested properly, simple continuity tests were done to verify the connection between the lines. In the first test, a connection was made between two cells via the switches; only the transistors were used, no laser links. The test vehicles were designed with blank pads, to allow the measurement of the internal resistances. Care must be taken when calculating the delays, because there is no large output buffers and the capacitance of the oscilloscope probe must be taken into account.

The resistance across two cells was 2618±14Ω. The delay of a square wave going through this path was 190±2ns. The maximum amplitude of the signal was 3.80±0.02V, because the pass transistors are N type (3.5μm width). A longer path was also simulated. This time the signal was run through all the cells, ten N pass transistors. The resistance measured was 14.15±0.1kΩ and the delay 800±5ns. Unless running at very low frequencies, the signal did not reach its full amplitude (~3V) because of the very large rise time. These tests show that the routing architecture of an FPGA is quite slow compared to a full custom chip design. Such delays are typical for FPGAs. Results of the laser linked path are presented later in this section.

5.4.2 XOR Delay Test

The next test was to verify the logic operation of the cells. To do this, a two input XOR gate was simulated in a cell. The experiment was performed on the small chips (ICBSFCD3), with a two input LUT. The setup for this experiment is shown in Figure 5.20. The delay of this function was measured by keeping one input low and applying a square wave to the other input. If the delays in table 5.1 are used, the path of this function can be approximated with: Dlb+2Dr+2Doh, giving a delay of 27.1 ns. This does not include the output capacitance of the probe; by adding the probe, 10MΩ and 11pF, a new Hspice simulation gives a result of 150ns and measurements indicate a 170±2ns delay. In another experiment, the output was run through another cell before reaching the pad and the probe.

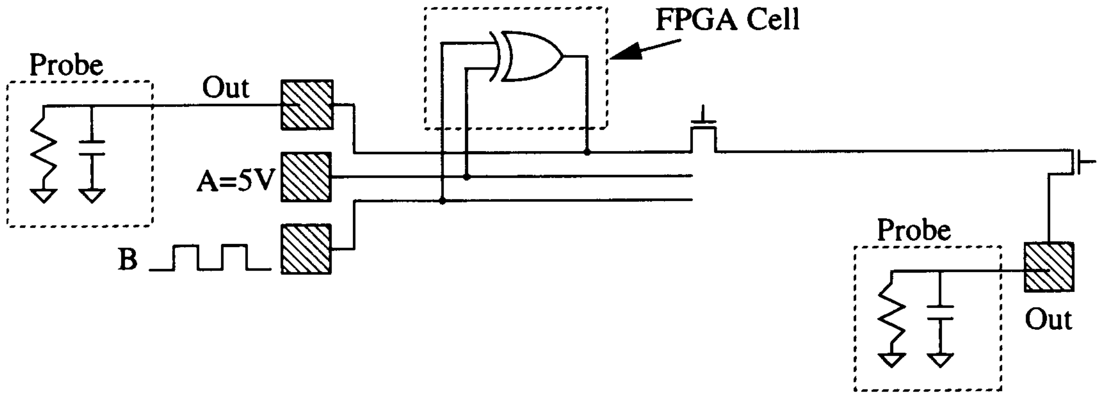


Figure 5.20 XOR Experiment Setup

The Hspice simulation for this circuit gives a 217ns delay while the measurements in such conditions show a delay of 300±2ns. With the high capacity of the probe (inaccurately known), it is hard to measure the internal delay. The simulations show however that the delay is well approximated with the Hspice simulation when considering the capacitive load. Results for the amplitudes are shown in table 5.3.

A(V)	B(V)	OUT(V)
0.00	0.00	0.40±0.01
0.00	5.00	3.75±0.02
5.00	0.00	3.75±0.02
5.00	5.00	0.40±0.01

Table 5.3: The XOR Gate

This experiment shows the little overhead delay simulated with Hspice is actually low compared to the routing delay of the FPGA.

5.4.3 Laser Linked Paths

The next experiment was done to test the restructuring of the FPGA possible with the laser links. The first path was the linking of the two single length lines through 3 cells. Access to the output pad was achieved by linking the laser switch in its downward mode. The paths are schematically represented in Figure 5.21.

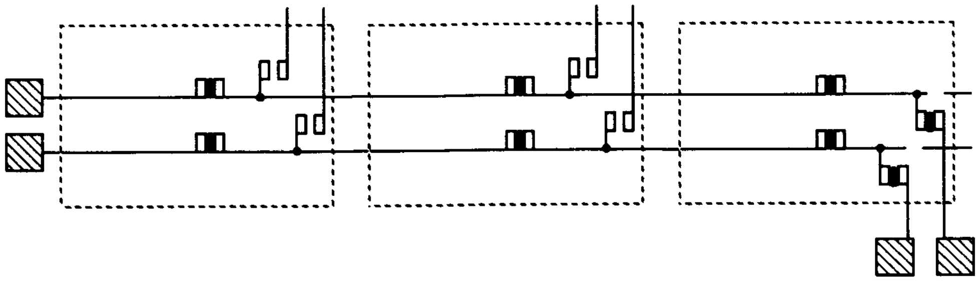


Figure 5.21 Laser Link Paths

The resistance of the first path was measured at 353±3Ω and the second path at 382±3Ω . The Hspice model for the lines and links gives a resistance of ~349Ω for such paths. Applying a 1MHz square wave gave a delay of 2.5±0.5ns for the first path and 3.0±0.5ns for the second path.

After this, another experiment was done to check the influence of bypass restructuring on the delays of the paths. The second cell was bypassed by using the laser link switches. The schematic representation can be seen in Figure 5.22. With this experiment, the delay added by such a restructuring can be measured.

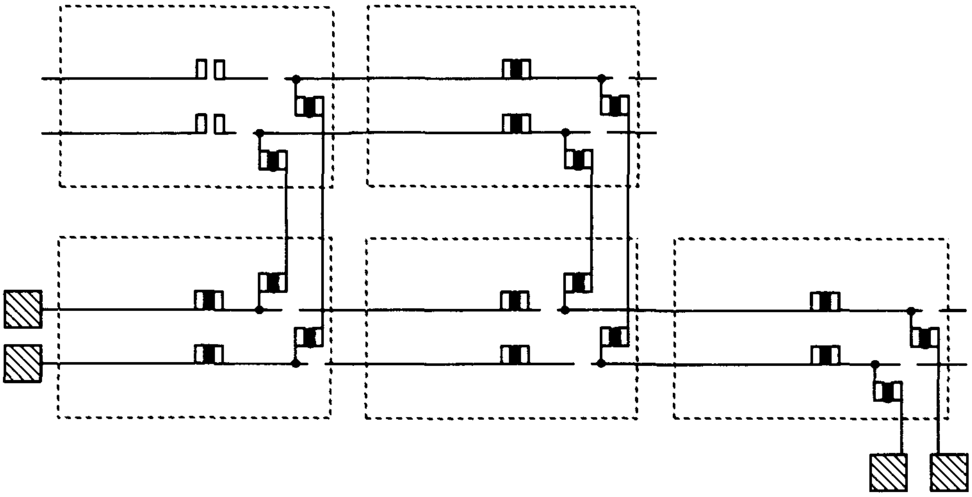


Figure 5.22 Restructuring Experiment

The results are as follows: the new resistance was 677±5Ω for the first path and 659±5Ω for the second path. Delays were measured as 11.0±0.5ns for both paths. Table 5.4 shows the results and the change in resistance and delay after the restructuring.

Path	R (Ω) before	Delay(ns) before	R(Ω) after	Delay(ns) after	Change on R	Change on Delay(ns)
Hspice	349	3.6	803	8.8	454	5.2
1	353±3	2.5±0.5	677±5	11.0±0.5	324±8	8.5±1
2	382±3	3.0±0.5	659±5	11.0±0.5	277±8	7.0±1

Table 5.4: Resistance and Delay of the Laser Linked Paths

The change in delay predicted by table 5.2 can be estimated to be 2xDrec which gives 6.0ns. The results show that this estimation is quite reasonable, almost matching the results within the expected error. In comparison, the same experiment was performed using the active reconfiguration in the cells. Note that this time only two cells were used instead of three. The results are shown in table 5.5.

Path	R (Ω) before	Delay(ns) before	R(Ω) after	Delay(ns) after	Change on R(Ω)	Change on Delay(ns)
1	2618±14	190±2	5710±29	370±2	3032±43	180±4
2	3340±18	202±2	6115±31	391±2	2775±49	189±4

Table 5.5: Resistance and delay of the Active Switch Paths

These results show well the large delays of the active switching compared to the laser links. The experiment was repeated on different chips and similar results were obtained.

5.4.4 Double Length Lines

Another similar experiment was done on the double length lines. This time to test the capacity to uncross the lines rather than the actual delay. The path was still going through three cells. The first step was to link the lines and measure the resistance and the delay. The second step of the experiment was to uncross the lines. First the two paths were laser cut; their resistance was too high to measure (>32MΩ), showing the cut was properly made. Thereafter the links were made to reconnect the lines: the effect was to invert the lines. The extra delay comes from the laser link. The results of the experiment are shown in table 5.6.

Path	R (Ω) before	Delay(ns) before	R (Ω) after	Delay(ns) after	Change on R(Ω)	Change on Delay(ns)
1	289±3	3.0±0.5	360±3	4.5±0.5	71±6	1.5±1
2	202±2	1.5±0.5	287±3	3.0±0.5	85±5	1.5±1

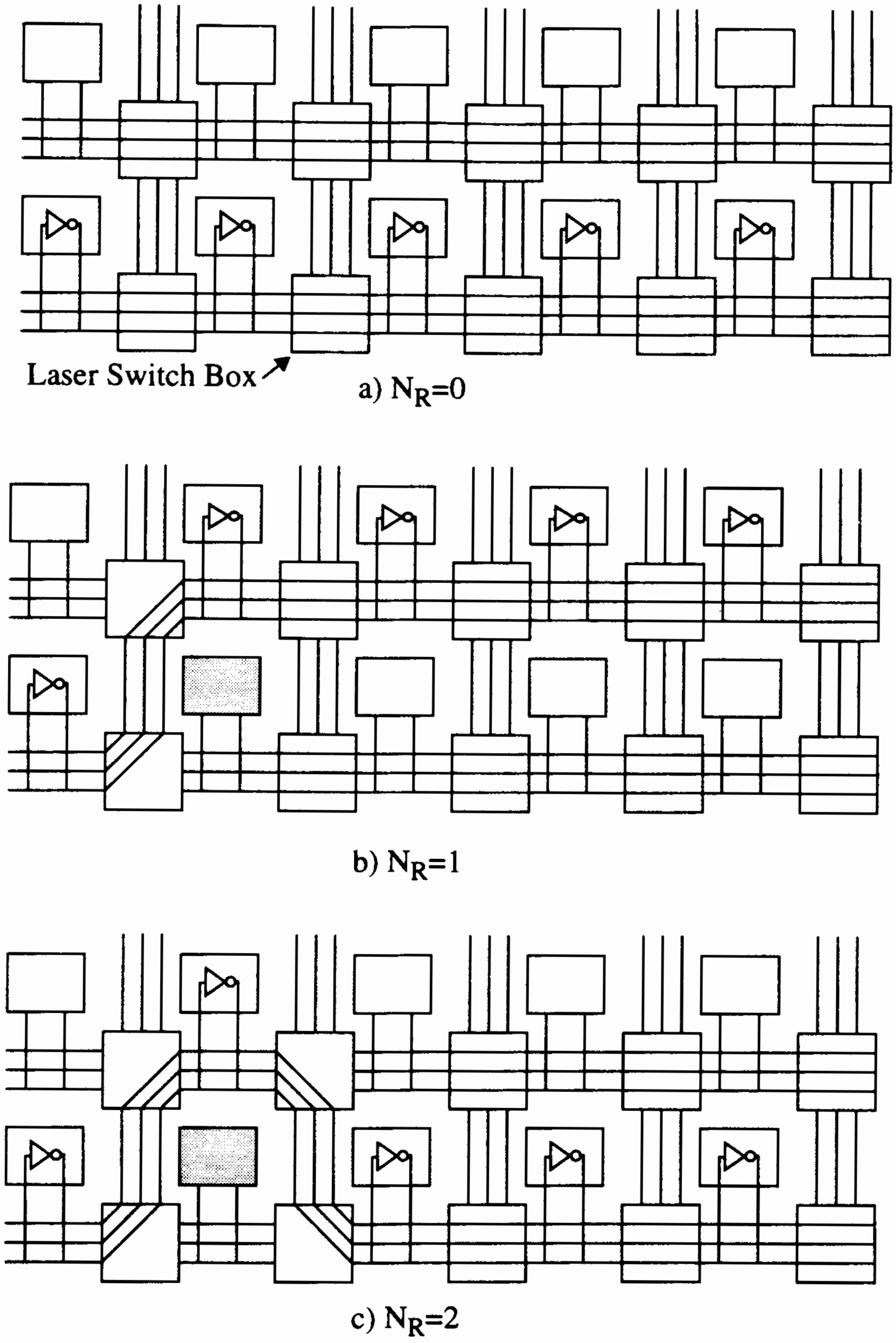
Table 5.6: Double Length Paths Uncrossing Results

This experiment shows that the line uncrossing can be done with no problems and the delay overhead is small.

5.4.5 The Ring Oscillator Test

A ring oscillator was programmed with the FPGA to verify the impact of the restructuring on the maximum frequency of operation of a circuit. Due to the small chip (ICBSFCD3) dimensions, a 2x5 array, the oscillator was made using six FPGA cells, five to simulate the inverters and one as an output buffer, so the capacitance of the oscilloscope would not affect the frequency.

The idea of the test is to simulate the oscillator first without restructuring, then with one restructured path and so on, until the maximum number of restructured paths possible with the chip was obtained (N_R=4). Hspice simulations using the proposed delay models gave the results shown in table 5.7. Some modifications on the length of the lines and the logic block circuit were made to take into consideration the smaller dimensions of the cell used in this experiment. The experimental results were obtained as follows: the ring oscillator was incrementally restructured and the frequency of operation measured for each value of N_R. The procedure is illustrated in Figure 5.23 (only the laser switch boxes are shown for clarity). In a), there is no restructuring; in b) one restructured path is included and in c) two restructured paths are shown.



a) $N_R=0$

b) $N_R=1$

c) $N_R=2$

Figure 5.23 Restructuring Experiment, Small Chip (ICBSFCD3)

Measurements were also taken for $N_R=3$ and $N_R=4$. The results of the experiment are shown in table 5.7.

This experiment shows the impact on the delay of an FPGA restructured in the cell by cell fashion.

Number of restructured paths (N_R)	Simulated Frequency (MHz)	Frequency Change from non-defective path(NR=0)	Experimental Frequency (MHz)	Frequency Change from non-defective path(NR=0)
0	4.89	-	4.82±0.01	-
1	4.67	-4.5%	4.73±0.01	-1.9%
2	4.49	-8.2%	4.50±0.01	-6.5%
3	4.34	-11.2%	4.29±0.01	-10.9%
4	4.23	-13.5%	4.28±0.01	-11.1%

Table 5.7: Hspice Simulation and Experiments for the Ring Oscillator, Small Chip (ICBSFCD3)

The waveforms were measured using a 100MHz digital oscilloscope. The period was measured with the cursors of the instrument. The precision on the period is ± 0.5ns.

The comparison between simulations and measurements is shown graphically in Figure 5.24.

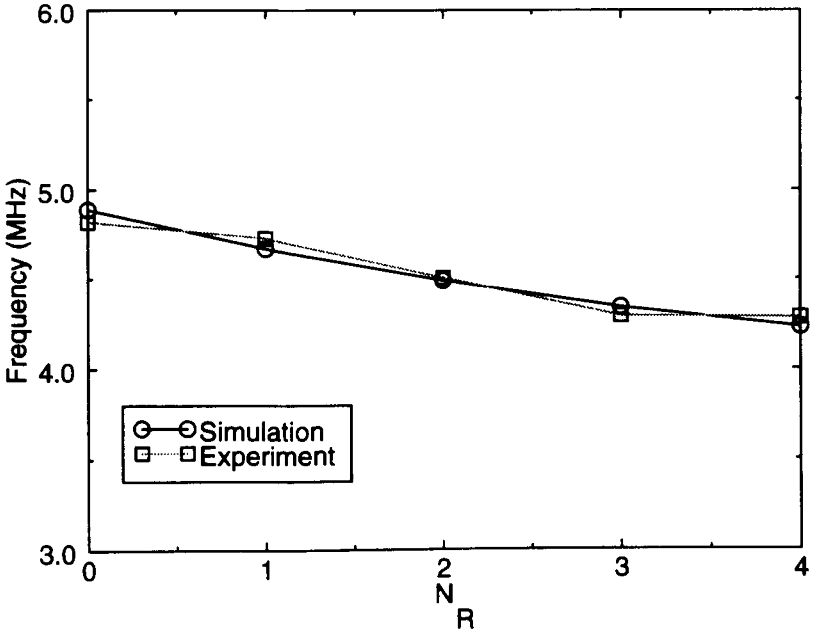
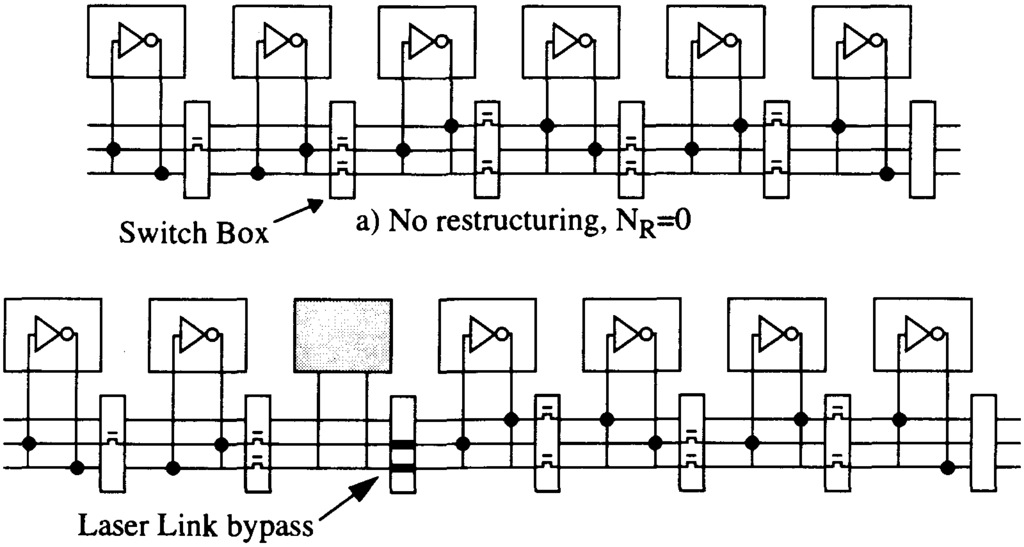


Figure 5.24 Ring Oscillator Test, Small Chip (ICBSFCD3)

Both the simulation and the experiment show the expected trends of decreasing frequency with increased N_R. The same experiment was repeated on another chip and the results were similar.

The same ring oscillator test was also performed on the large chip (ICBSFCD4). Only this time the restructuring was different. Since this chip is a long row (1x12 cells), it was used to measure the impact of the column substitution, where the number of laser links in the restructuring path is smaller, but where the bypass routing length is longer. Another interesting aspect of this experiment is that it uses the new device called the laser pass transistor shown in Figure 4.23 so it was possible to show the difference between active switching and laser linking. The experiment was done as shown in Figure 5.25. In a), there is no laser restructuring. Then in b), a cell was bypassed and the new frequency was measured. The experiment was repeated for N_R=2, 3 and 4. The results of both the simulations and the experiments are given in table 5.8 for active switching and in table 5.9 for laser linking.



a) No restructuring, N_R=0

b) One laser restructuring, N_R=1

Figure 5.25 Ring Oscillator Restructuring Experiment, Large Chip (ICBSFCD4)

Number of restructured paths (N_R)	Simulated Frequency (kHz)	Frequency Change from non-defective path(NR=0)	Experimental Frequency (kHz)	Frequency Change from non-defective path(NR=0)
0	943	-	956±2	-
1	855	-9.3%	859±2	-10.1%
2	787	-16.5%	727±2	-24.0%
3	690	-26.8%	629±2	-34.2%
4	549	-41.8%	543±2	-43.2%

Table 5.8: Results for the Ring Oscillator, Active Switching, Large Chip (ICBSFCD4)

Number of restructured paths (N_R)	Simulated Frequency (kHz)	Frequency Change from non-defective path(NR=0)	Experimental Frequency (kHz)	Frequency Change from non-defective path(NR=0)
0	943	-	956±2	-
1	885	-6.2%	906±2	-5.2%
2	833	-11.7%	820±2	-14.2%
3	787	-16.5%	761±2	-20.4%
4	746	-20.9%	722±2	-24.5%

Table 5.9: Results for the Ring Oscillator, Laser Linking, Large Chip (ICBSFCD4)

Those results are shown graphically in Figure 5.26. It shows clearly that the laser linking offers better performance when compared with active switching. While both are the same for N_R=0, by N_R=4 the laser linking oscillator is 33% faster than the actively switched device. Both simulations and experiments match very well. The experiment was repeated on another chip and the results were similar.

Those restructuring experiments showed it is possible to use the laser link to restructure FPGAs to bypass defects, and showed that it gives better performance then active switching. The cases with N_R=4 are cases where a very low cell yield is obtained and would rarely be encountered on a real wafer. The simulation was repeated with wider pass transistors, double the size of those designed (7.0μm instead of 3.5μm). The active switching showed a decrease of 18.3% in performance while the laser link showed a 12.3% decrease. At this width, the pass transistors are the same size as the laser link.

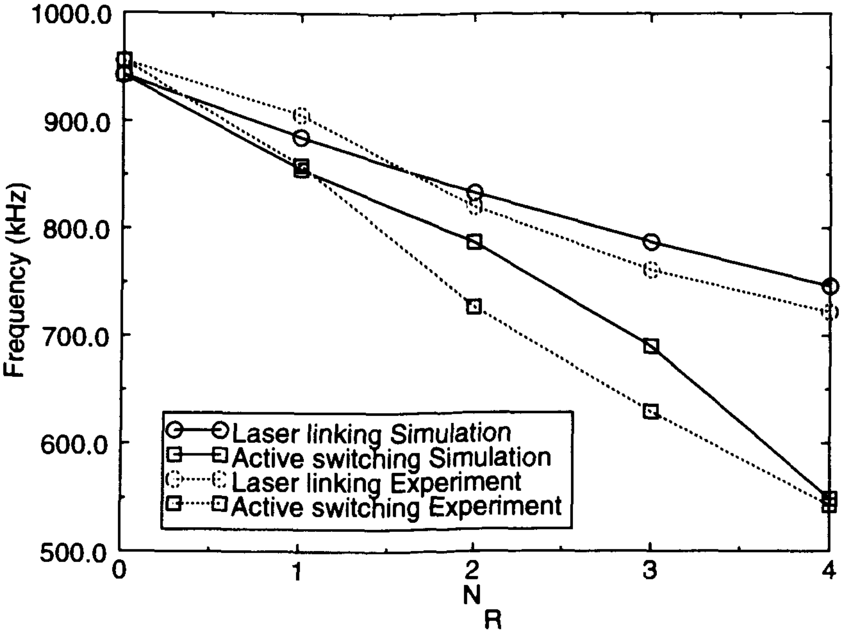


Figure 5.26 Ring Oscillator Test, Large Chip (ICBSFCD4)

5.4.6 Larger Cell Simulation

The simulations from the previous section were done for the fabricated devices. In this section, the comparison is made for cells of larger size, which are more likely to be used in a real wafer scale design. The logic block is optimized for speed in these simulations, so better results are obtained than for the simulations which take the fabricated design into account. A comparison is also made with a custom hard-wired circuit, with the routing switches replaced by a direct line connection. Cells of 1, 2 and 5 times larger were simulated. The same restructuring scheme as the previous experiment was used. The results are shown in table 5.11 for the direct connection, active switching and laser linking. The 1x means the designed cell size was used (1206μm x 650μm), in 2x a 2412μm x 1300μm cell was used while 5x means a 6030μm x 3250μm cell was used. A graph of these results for active switching and laser linking is shown in Figure 5.27.

	Direct connection			Laser linking			Active switching		
N_R	1x	2x	5x	1x	2x	5x	1x	2x	5x
0	4080	2750	1400	2850	1850	926	2850	1850	926
1	3720	2520	1220	2420	1510	719	2270	1430	680
2	3400	2230	1050	2100	1290	592	1850	1120	515
3	3140	2000	943	1850	1110	500	1470	862	382
4	2920	1800	826	1690	990	442	1030	565	221

Table 5.10: Larger Cells Comparison, Frequency in kHz

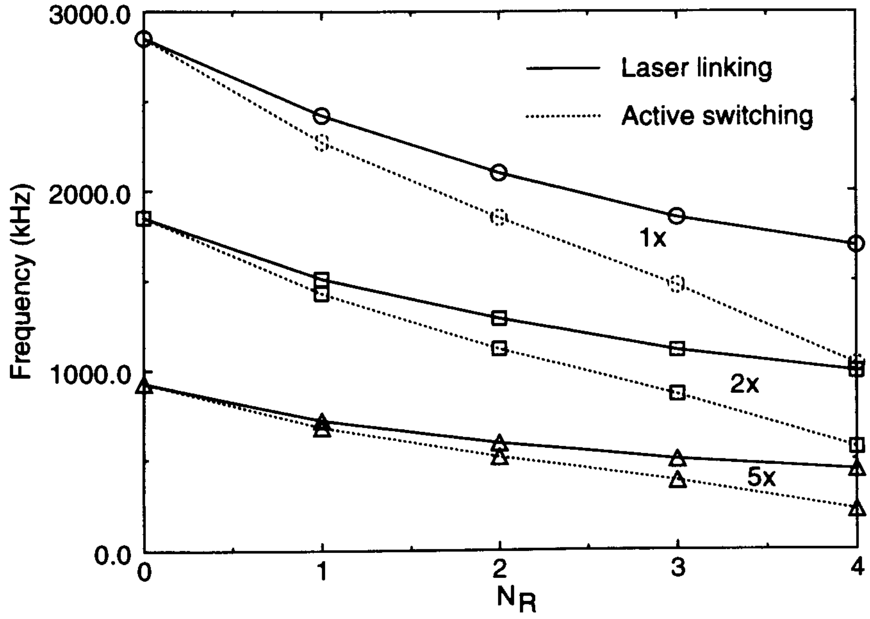


Figure 5.27 Larger Cells Comparison, Active Switching and Laser Linking

The values indicate the laser link is better even for very large cells, going from 1.6 times faster for the small cells to two times better than active switching for a cell of 6030μm x 3250μm. Note these results are comparing the results for the length of the routing architectures, and do not take into account the change in performance of the logic block. Clearly, with larger cells, more complex circuits can be done with each block.

5.5 Summary

In this Chapter, the design of the test vehicle was presented. Results of the HSpice simulation were compared with the experiments performed on two chip designs fabricated in Mitel 1.5μm technology. The effect of the overhead restructuring circuitry was analyzed in detail and compared to the simulations. It has been shown that the delay overhead is reasonably small and that there is advantages in using the Laser Link to restructure FPGAs to bypass defects.

Chapter 6

Conclusion

The work presented in this thesis was aimed at investigating a solution to the interconnection problems associated with a wafer scale FPGA. The design and testing of a test vehicle was done to show the influence of the restructuring circuitry on the performance of the FPGA. The first section of the conclusion deals with the results obtained with the test vehicle and how they can be used to expand the size of the devices. The second section deals with the technical and economical feasibility of the wafer scale and large FPGAs. The last section will present some future work.

6.1 The Test Vehicle

A seen in Chapter 5, there is an overhead associated with the laser restructuring of the FPGA compared to a standard design. One of the objective of the thesis was to show that while this overhead is not negligeable, it is small because of the large delays already imposed by the routing structure of the FPGA. The simulations and tests have shown the overhead in delay in the order of 10-15% for highly defective areas (around 50% cell yield) for the cell by cell restructuring, while the row-column restructuring gave decreases in performance in the order of 25% for highly defective areas. One thing that must be taken into account is that it is unlikely that the worst path of a given circuit will actually be mapped on the worst physical path of the FPGA, and this can result in circuits with performances close to standard FPGAs.

The limitation in silicon area available did not allow us to test cells with a size used today. It had been shown however that the increase in the area of the cell actually decreases the influence of the laser restructuring, since the resistance of the laser link is constant and the speed of the circuit decreases with the increase of the cell area. A design with large cells also requires a smaller number of cells and thus a smaller number of restructuring circuits.

The results obtained with the test vehicle demonstrate the possibility of laser restructuring FPGAs and therefore increasing their size or yield with a small penalty on circuit performance.

6.2 Technical and Economical Feasibility

Two different approaches to the laser restructuring of FPGAs have been presented in this thesis. The first one is to use a complex defect avoidance scheme that allows the size of FPGAs to increase up to full wafer devices. The other one deals with the yield of standard size devices. By using a low overhead circuitry and a simple row column defect avoidance scheme, it is possible to increase the yield of the devices. A similar approach is used in dynamic RAMs today. This approach could decrease the cost of large FPGAs and even be used to increase the size of high-end devices.

The full wafer scale FPGAs would offer considerably higher capability than current FPGAs but requires more area and the restructuring costs more than standard devices. Their use is however targeted for applications that use very expensive equipment today. The main advantage of a wafer scale product is once the production is launched, the cost of rework is kept small. One of the problem with boards of FPGAs, apart from being slower than a potential wafer scale design, is they cannot use standard FPGAs, so a custom design has to be build. The amount of rework on a board or multi chip module is also very high, and the probability of breaking a device is large when the rework is performed. While a wafer scale product would be technically superior, an analysis of the market should be done to build a system that suits the needs of the demand. As one designer said: "As far as density, the sky's the limit. If the gates are there, I'll use them-40,000 or 500,000 gates" [38].

6.3 Future Work

The experiments on the test vehicles show it is possible to build a wafer scale device with low overhead. More tests should however be done to slowly increase the size of the devices. A more complex cell should also be used. The next logical step in the project would be to employ a commercially available cell and implement the restructuring on it. Building a small area restructurable FPGA that can be programmed using available software and compare its performance to a custom device would give useful insights. This kind of project requires cooperation and the work of many people, which is beyond a student's thesis. The work presented in this thesis is however the necessary first step and the results presented are showing that the next step is possible.

Other interesting options come to mind when talking about a wafer scale FPGA. One of the main problem of today's designs is the lack of embedded memory. With a wafer scale design, it is possible to include significant amounts of memory, distributed on the chip, that can be used by the circuit. It is also possible to build a wafer scale design with many different blocks, to reach better performance. This is the trend today, but the low area of the chip is an obstacle. FPGAs are very popular today and will probably be more popular in the future. If the chip size barrier can be broken, this future could be even more interesting.

6.4 Summary

The goal of the thesis was to propose a way to restructure FPGAs by using Laser Link Technology. Different defect avoidance approaches are presented and simulations performed. It is shown that a combination of row and cell restructuring, and line redundancy inside the channels, gives good yields. A test vehicle was designed and fabricated to test the different aspects of a wafer scale design. While further work still need to be done, results show a laser restructured wafer scale FPGA is feasible.

List of References

[1] B. Fuller, “AT&T rolls 40 000-gate Orca FPGA”, *Electronic Engineering Times*, pp. 1 and 108, February 1995.

[2] S.D. Brown, R.J. Francis, J. Rose and Z.G. Vranesic, “Field Programmable Gate Arrays” *ed. Kluwer Academic, Boston,* 1992.

[3] D.E. Van Den Bout, J.N. Morris, D. Thomae, S. Labrozzi, S. Wingo, D. Hallman, “AnyBoard: An FPGA-Based, Reconfigurable System”, *in IEEE Design and Test of Computers, vol. 9, no. 3*, pp. 21-30, September 1992

[4] J.I. Raffel, A.H. Anderson and G.H. Chapman, “Laser Restructurable Technology and Design”, *in Wafer Scale Integration, E. Swartzlander, ed. Kluwer Academic, Boston,* ch. 7, 1988.

[5] J.F. McDonald, B. Philhower and H.J. Greub, “A Fined Grained, Highly Fault Tolerant System based on WSI and FPGA Technology”, *in FPGAs, ed. W. Moore and W. Luk, Abingdon EE&CS Books, Abingdon*, pp.114-126, 1991.

[6] S.K. Tewksbury, “Wafer-Level Integrated Systems: Implementation Issues”, *Kluwer Academic Publishers, Boston,*1989.

[7] G.H. Chapman and K. Fang, “Comparison of Laser Link Crossbar and Omega Switching for Wafer-Scale Integration Defect Avoidance”, *Proceedings of the IEEE International Conference on Wafer Scale Integration*, pp. 352-361, San Francisco, CA, January 1994.

[8] J. Wilson, J.F.B. Hawkes, “Lasers, Principles and Applications”, *Prentice Hall, New-York,* 1987.

[9] D.R. Messier and W.J. Croft, “Silicon Nitride”, *ch. 2 of Preparation and Properties of Solid State Materials, vol. 7, edited by W. Wilcox, New York,* 1982.

[10] W.M. Steen, “Laser Material Processing”, *Springer-Verlag, London,* 1991.

[11] V.I. Belyi, et al., “Silicon Nitride in Electronics”, *Elsevier, Amsterdam*, 1988.

[12] S.S. Cohen and G.H. Chapman, “Laser Beam Processing and Wafer Scale Integration”, *chapter 2, VLSI Electronics: Microstructure science*, vol. 21, pp 10-

111, 1989.

[13] M. J. Syrzycki, L. S. Carr, G. H. Chapman and M. Parameswaran, "A Wafer-Scale Visual-to-Thermal Converter", *Proceedings of the IEEE International Conference on Wafer Scale Integration*, pp. 1-10, San Francisco, CA, January 1993.

[14] G.H. Chapman, J.I. Raffel, J.M. Canter and F.M. Rhodes, "Advances in laser link technology for wafer scale circuits", IFIP Int. Workshop on Wafer-Scale Integration, Brunel University, Sept. 23-25, 1987.

[15] G. H. Chapman, L. S. Carr, M. J. Syrzycki and B. Dufort, "Test Vehicle for a Wafer Scale Thermal Pixel Scene Simulator", *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 17, no. 3 , pp. 334-341, August 1994.

[16] L. E. Laforge, "What Designers of Wafer Scale Systems Should Know about Local Sparing", *Proceedings of the 1994 Intl. Conference on Wafer Scale Integration*, pp. 106-131, San Francisco, CA, January 1994.

[17] R. Negrini, M.G. Sami and R Stefanelli, "Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays", *MIT Press, Cambridge*, ch. 14, 1989.

[18] C. H. Stapper, F. M. Armstrong and K. Saji, "Integrated Circuit Yield Statistics", *Proceedings of the IEEE*, vol. 71, no. 4, pp. 453-468, April 1983.

[19] C. H. Stapper, "Yield Model for Fault Clusters within Integrated Circuits", *IBM Journal Res. Develop.*, vol. 28, no. 5, pp. 636-639, September 1984.

[20] S.K. Tewksbury, "Wafer-Level Integrated Systems: Implementation Issues", *Kluwer Academic Publishers, Boston*, Ch. 5: Yield Models and Analysis, 1989.

[21] W. Moore, W. Maly and A. Strojwas, "Yield Modeling and Defect Tolerance in VLSI", *Adam Hilger, Bristol*, 1988.

[22] C.H. Stapper, A.N. McLaren and M. Dreckmann, "Yield Model for Productivity Optimization of VLSI Memory Chips with Redundancy and Partially Good Product", *IBM Journal Res. Develop.*, vol. 24, no. 3, pp. 398-409, May 1980.

[23] C. H. Stapper, "Block Alignment: A Method for Increasing the Yield of Memory Chips that are Partially Good", in Defect and Fault Tolerance in VLSI Systems I, pp.243-255, 1989.

[24] S. Kuo and W. K. Fuchs, “Efficient Spare Allocation for Reconfigurable Arrays”, in IEEE Design and Test, pp. 24-31, February 1987.

[25] R.M. Lea, “A 3-D WASP Module for Real-Time Signal and Data Processing”, *Proceedings of the 1992 Intl. Conference on Wafer Scale Integration*, pp. 95-104, San Francisco, CA, January 1992.

[26] D.C. Keezer and V.K. Jain, “Clock Distribution Strategies for WSI: A Critical Survey”, *Proceedings of the IEEE International Conference on Wafer Scale Integration*, pp. 277-283, San Francisco, CA, January 1991.

[27] S.H.K. Embabi and D.E. Brueske, “Clock Synchronization for WSI Systems”, *Proceedings of the IEEE International Conference on Wafer Scale Integration*, pp. 228-234, San Francisco, CA, January 1994.

[28] D. Audet, Y. Savaria and N. Arel, “An Architectural Approach for Increasing Clock Frequency and Communication Speed in Monolithic-WSI Systems, *Proceedings of the IEEE International Conference on Wafer Scale Integration*, pp. 235-243, San Francisco, CA, January 1994.

[29] Xilinx, “The Programmable Logic Data Book”, *San Jose*, 1994.

[30] P. Chow, S.O. Seo, D. Au, T. Choy, B. Fallah, D. Lewis, C. Li and J. Rose, “A 1.2μm CMOS FPGA using Cascaded Logic Blocks and Segmented Routing”, *in FPGAs, ed. W. Moore and W. Luk, Abingdon EE&CS Books, Abingdon* pp. 91-102, 1991.

[31] V.N. Doniants, V.G. Lazarev, M.G. Sami and R. Stefanelli, “Reconfiguration of VLSI Arrays: a technique for increased flexibility and reliability”, *Microprocessing and Microprogramming*, vol. 16, pp. 101-106, 1985.

[32] “Restructurable VLSI Program: Semiannual Technical Summary”, Lincoln Laboratories Technical Report, Sept. 1985.

[33] S. Singh, J.Rose, D. Lewis, K. Chung, P. Chow, “Optimization of Field-Programmable Gate Array Logic Block Architecture for Speed”, *in IEEE 1991 Custom Integrated Circuits Conference*, pp 6.1.1-6.1.6, 1991.

[34] S. M. Trimberger, “Field-Programmable Gate Array Technology”, *Kluwer Academic Publisher, Boston*, 1994.

[35] J.L. Kouloheris, A.E. Gamal, "FPGA Performance versus Cell Granularity", *in IEEE 1991 Custom Integrated Circuits Conference*, pp 6.2.1-6.1.5, 1991.

[36] D. Galloway, D. Karchmer, P. Chow, D. Lewis, J. Rose, "The Transmogrifier: The University of Toronto Field-Programmable System", *Proceedings of the 1994 Canadian Workshop on Field-Programmable Devices*, pp. 1.4.1-1.4.6, June 1994.

[37] National Instrument, "NI-DAQ Function Reference Manual for PC Compatibles", November 1993.

[38] "ORCA: Opening Design Doors", *Supplement to Electronic Engineering Times*, p.6, April 17 1995.

Appendix A: Hspice Ring Oscillator File

Hspice file sample, osc.sp, to simulate the restructuration on the ring oscillator for the small chip (ICBSFCD3).

```
*
.MODEL PMITEL PMOS LEVEL=3 COX=1.28E-3 DERIV=1 KAPPA=5.69
+ KP=17.65E-6 TOX=27E-9 VMAX=216.54E3 LD=152.91E-9 LMLT=1
+ WD=97.74E-9 WMLT=1 XJ=403.39E-9 DELTA=1.53 ETA=146.55E-3
+ NFS=224.75E9 NSUB=3.6E16 PHI=765.08E-3 VTO=-559.63E-3
+ THETA=35.94E-3 UO=138.03
*
.MODEL NMITEL NMOS LEVEL=3 COX=1.28E-3 DERIV=1 KAPPA=10E-3
+ KP=68.03E-6 TOX=27E-9 VMAX=160.4E3 LD=239.68E-9 LMLT=1
+ WD=100E-12 WMLT=1 XJ=296.9E-9 DELTA=1.06 ETA=140.68E-3
+ NFS=868.21E9 NSUB=1.6E16 PHI=725.15E-3 VTO=827.59E-3
+ THETA=34.67E-3 UO=531.92

*** Lines Models

.MODEL linepoly R COX=0.00012 RSH=20.0
.MODEL linemetal1 R COX=0.00014 RSH=0.04 CAPSW=2.3E-10
.MODEL linemetal2 R COX=0.00012 RSH=0.04 CAPSW=2.3E-10

.subckt laserlink in out
R1 in out 108
m1 in 0 out 0 nmitel l=1.5u w=6.6u ad=34.98p as=34.98p pd=23.8u ps=23.8u
.ends

.subckt laserlinkopen in out
R1 in out 10E6
m1 in 0 out 0 nmitel l=1.5u w=6.6u ad=34.98p as=34.98p pd=23.8u ps=23.8u
.ends

.subckt nswitch in out
m1 in vdd out 0 nmitel l=1.5u w=3.5u ad=11.55p as=11.55p pd=13.6u ps=13.6u
vdd vdd 0 5v dc
.ends

.subckt openswitch in
m1 in 0 0 0 nmitel l=1.5u w=3.0u ad=9.9p as=9.9p pd=12.6u ps=12.6u
.ends

.subckt inv in out
m1 vdd in out vdd pmitel l=1.5u w=3.0u ad=10.87p as=10.87p pd=13.2u ps=13.2u
```

```
m2 out in 0 0 nmitel l=1.5u w=1.5u ad=10.87p as=10.87p pd=13.2u ps=13.2u
vdd vdd 0 5v dc
.ends

.subckt bigbuffer in out
.subckt inv in out
m1 vdd in out vdd pmitel l=1.5u w=3u ad=10.87p as=10.87p pd=13.2u ps=13.2u
m2 out in 0 0 nmitel l=1.5u w=1.5u ad=10.87p as=10.87p pd=13.2u ps=13.2u
vdd vdd 0 5v dc
.ends
.subckt inv2 in out
m1 vdd in out vdd pmitel l=1.5u w=60.0u ad=198p as=198p pd=126.6u ps=126.6u
m2 out in 0 0 nmitel l=1.5u w=30.0u ad=99p as=99p pd=66.6u ps=66.6u
vdd vdd 0 5v dc
.ends
x1 in 1 inv
x2 1 out inv2
.ends

.subckt tgate in out
m1 in 0 out vdd pmitel l=1.5u w=6.0u ad=24.78p as=24.78p pd=19.8u ps=19.8u
m2 in vdd out 0 nmitel l=1.5u w=3.0u ad=10.87p as=10.87p pd=13.2u ps=13.2u
vdd vdd 0 5v dc
.ends

*** Logic block

.subckt dlb in out
x1 in 1 nswitch
x2 1 2 inv
x3 vdd 10 inv
x4 0 11 inv
m1 10 1 3 0 nmitel l=1.5u w=1.5u ad=10.87p as=10.87p pd=13.2u ps=13.2u
m2 0 1 0 0 nmitel l=1.5u w=1.5u ad=10.87p as=10.87p pd=13.2u ps=13.2u
m3 11 2 3 0 nmitel l=1.5u w=1.5u ad=10.87p as=10.87p pd=13.2u ps=13.2u
m4 0 1 0 0 nmitel l=1.5u w=1.5u ad=10.87p as=10.87p pd=13.2u ps=13.2u
m5 3 vdd 5 nmitel l=1.5u w=1.5u ad=9.9p as=9.9p pd=12.6u ps=12.6u
m6 vdd 5 9 vdd pmitel l=1.5u w=7.2u
m7 9 5 0 0 nmitel l=1.5u w=5u
x5 5 6 tgate
x6 6 7 tgate
x7 7 8 bigbuffer
x8 8 out nswitch
vdd vdd 0 5v dc
.ends
```

```
*** Routing

.subckt dr in out
x1 in openswitch
x2 in openswitch
x3 in openswitch
x4 in openswitch
r1 in 1 linemetal1 l=1150u w=3.3u
x5 1 2 nswitch
r2 2 out linemetal1 l=100u w=3.3u
x6 2 0 laserlinkopen
x7 2 0 laserlinkopen
x8 2 0 laserlinkopen
x9 2 0 laserlinkopen
.ends

*** Direct routing(for double length lines)

.subckt dir in out
x1 in openswitch
x2 in openswitch
x3 in openswitch
x4 in openswitch
r1 in 1 linemetal1 l=1150u w=3.3u
r5 1 2 linemetal1 l=2u w=3.3u
r2 2 out linemetal1 l=100u w=3.3u
x6 2 0 laserlinkopen
x7 2 0 laserlinkopen
x8 2 0 laserlinkopen
x9 2 0 laserlinkopen
.ends

*** Reconfiguration

.subckt drec in out
x1 in 2 laserlink
r1 2 3 linemetal1 l=333u w=3.3u
x2 3 out laserlink
.ends

x1 1 2 dlb
x2 2 3 dr
*x20 30 3 drec
x3 3 4 dlb
x4 4 5 dir
*x40 50 5 drec
```

```
x5 5 6 dlb
x6 6 7 dr
*x60 70 7 drec
x7 7 8 dlb
x8 8 9 dir
*x80 90 9 drec
x9 9 10 dlb
x10 10 11 dr
*x100 110 11 drec
x11 11 12 dir
*x110 120 12 drec
x12 12 13 dr
*x120 130 13 drec
x13 13 14 dir
*x130 140 14 drec
x14 14 1 dr

x15 3 15 dr
x16 15 out dlb

c1 out 0 10p
c2 1 0 1.4p
c3 2 0 1.4p

vdd vdd 0 5v dc

.ic v(1)=0v
.tran 1ns 500ns
.OPTIONS post
.END
```