Architectural Design of an 8-bit Fault-Tolerant Computer Implemented by WSI

Zhihui Wan

Supervisor: Carl McCrosky

Abstract

The main obstacle of WSI approach is due to the unavoidable wafer defects, which incur low wafer yield. Various fault-tolerant techniques designed to increase wafer yield are reviewed and evaluated; two general types: fault-masking and reconfiguring are summarized. And then, one and two level wafer scaling schemes incorporated with fault-tolerant techniques and system complexity are discussed and compared. The current research trend is dealing with homogeneous systems because they need few spare elements. This study is trying to implement a general purpose 8-bit computer by means of reconfiguring or restructuring technique; the main idea of reducing spare elements is partitioning the system into logic modules and then applying redundancy to design a universal module, which is capable of replacing any faulty module in the system; the overheads of the redundancy and interconnects are then discussed.

1 Introduction

1.1 Motivations

For at least the past few years, reduction in feature size has been the primary means for achievement of increased density. It is now becoming more technically and economically difficult to continue the trend[1]. One alternative to increase the integrated density is increasing the chip size.

It has long been a dream that a whole system is built up on a single chip of wafer scale size (Wafer Scale Integration, WSI). The obvious promises of WSI approach include greatly improved reliability, reduced power dissipation, reduced cost, smaller size, shorter signal delays and simpler packaging.

The major obstacle with WSI is that there are always some unavoidable defects in the wafer, which will cause the whole system to fail if those defects are not taken care of. The viability of a WSI system mainly depends on how well those defects are dealt with. A WSI system has to be fault-tolerant.

Although it is technically feasible to build a fault-tolerant WSI system by means of various fault-tolerant techniques, technical success is not equal to commercial success. WSI systems will only become commercially attractive when the price per function is lower than traditional VLSI system. The key problem is how to increase the yield of a wafer while keeping the system fault-tolerant.

Although there are various yield models, yield always increases with the decrease of cell area in every model. All the fault-tolerant techniques will need to introduce more circuits and therefore increase the cell areas. It is necessary to evaluate various fault-tolerant techniques ant then optimizing the wafer scaling scheme.

1.2 Brief Research History Review

Probably the first publication to treat WSI was issued by Sack, Lyman and Chang at Westinghouse in 1964. During the 60's, discretionary wiring technique was employed by Texas Instruments to allow good circuits to be connected after test. Hughes Aircraft Company made some improvement by reducing the number of discretionary masks and by testing at a point closer ot the end of processing.

During the 70's, several approachs emerged that allowed the complete fabrication of the circuit before testing and reconfiguration. McDonnell Douglas defined a scheme called "Adaptive WSI". The approach used a nonvolatile MNOS memory to isolate the defective regions. A group associated with Middelsex Polytechnic in England also suggested a scheme based on the use of volatile switches, self testing and a serial linking of good circuit modules. MIT Lincoln Laboratories introduced the famous restructurable, very large scale integration(RVLSI) concept, a key feature of which is the use of a laser to connect and/or disconnect multilevel metal patterns on the wafer. Manning described a design using rectangular arrays of programmable logic. The idea involves the use of on-chip electronically controlled interconnections added in each cell.

During 80's, Trilogy Systems Corporation undertook the development of a large, high speed computer based on WSI circuit modules. Inova Microelectronics Corporation successfully developed 1 M-bit CMOS static RAM devices by using laser cutting to restructure circuits. Westinghouse Electric initiated a WSI program for the development of complete signal and data processors.

The current research areas covered from self-testing and self-configuring techniques, incorporated programmable logics, architectural design, spare allocation algorithms, wafer-scale testing, design for testability and packaging techniques.

2 Fault-Tolerant Technique Survey

2.1 Fault-Masking Techniques

Fault-masking techniques include TMR (Triple Modular Redundancy) and NMMR (N-M Modular Redundancy. Both techniques work in the same manner. The majority of all the signals will be voted as correct signal. The error signal will be voted out. The good part of those techniques is that there is no need to do any testing. The bad part is that they are very expensive in the sense of redundancy. To mask 1 possible error, at least 3 modules are need; to mask 2 possible errors, 5 modules are needed; in general, to mask n possible errors, 2n plus 1 modules are needed.

2.2 Restructuring Techniques

Restructuring techniques are always involved in testing modules, isolating faulty modules and then replace the faulty ones with a spare module. In brief, restructuring techniques can be roughly divided into three categories: laser restructuring, programmable restructuring, self-testing and self-configuring.

Self-testing and self-reconfiguration techniques are very attractive approach, which is involved in capturing abnormal signals and then triggering the reconfiguring switches. The abnormal signals could be of voltage, current or capacitance. Maly and Nigh's(1988) built-in current sensor is a good example (Wojciech[5]). The nice part of this approach is that it is able to provide the compile-time reconfiguring. The tough part is that 100 per cent self-testing circuits are not easy to design. It is difficult for this approach to provide completely fault-free circuits because there are too many possibilities.

Programmable Reconfiguration involves the use of on-chip electronically controlled interconnections added in each cell. And then a few input-output leads are used to allow maintenance and customization of arbitrarily large arrays. The good part of this techniqu is its great flexibility due to its programmable nature. The other side of the technique is that the number of the input-output leads is very limited; therefore the cell size or module has to be relatively large and wafer yield may therefore be decreased; the cost of interconnects is also high.

Laser Restructuring techniques are necessary because any processes before and including packaging process could cause chip damage. Without proper afterward repairing techniques like laser restructuring, WSI will not be realistic. The idea of Laser Restructuring if to fabricate wafer with arrays of circuits and vertical and horizontal channels for interconnect. Each cell, the smallest replaceable module, is electrically isolated and has pads for wafer probe testing; each circuit signal pin has a short wire crossing the channel for making connections. Before restructuring, complete testing on circuits and channels is necessary. After testing, laser is used to connect the good cells and disconnect the faulty ones. The linking of the good cells requires a computationally intensive routing step which is followed by a computer controlled laser restructuring. Laser restructuring is much more flexible, requiring no custom mask and almost instantaneous.

The nice part of the programmable reconfiguring and laser restructuring is that they will provide 100 per cent fault-free circuits. The cost of that benefit is that every cell has to be tested before restructuring starts.

3 System Complexity Reduction

The number of subsystems or modules has a great impact on the cost of spare elements. The current research trend focuses on homogeneous system like RAM. Many algorithms and techniques are designed to efficiently allocate spare elements. But it appears that very little research was done to apply those algorithms and techniques on a diversified system like a general purpose computer. This study is trying to explore the way to reduce the complexity of a diversified system. Some of the primary thoughts are as follows.

First of all, all the digital systems are hierarchically built from CMos gates, to logic gates, to modules like flip-flops and counters, to subsystems like controllers, ALU, and then to the system. Apparently, different level of the hierarchy has different number of modules or submodules. Choosing the proper level as the floorplaning base is the important step in reducing the number of subsystems.

Another approach is to apply redundancy to some modules and make them become identical modules. For example, a 2-bit counter and a 3-bit counter can be implemented as 2 3-bit counters. In extremely case, all the combinational logic modules can be implemented as decoder modules or programmable logic arrays. But the cost of redundancy and interconnect could be very high.

The way we think is more economic is to design general purpose (universal) spare elements, which is capable to replace any faulty modules. The idea is to AND all the function sets of each module in the system to get the functions for the universal spare elem. The cost of redundancy and interconnect may vary a lot depending on the system. The implementation for the 8-bit general purpose computer will be used as an real example.

4 One and Two Level Wafer Scaling Schemes

The key issue of one level wafer scaling is how to choose proper cell size. If one chooses too large, the wafer yield will be very low because a single defect within one cell will disable the cell. If one chooses too small, testing on modules will be difficult and cost a lot. If fault-masking techniques are employed, an example of this was the Trilogy Design (Peltzer, 1983), the cost of redundancy will be high.

As a improvement, Strader and Harden (1989) introduced a two level wafer scaling scheme. The basic idea is to divided a wafer into blocks, which again be divided into cells. The cell size can be very small and therefore wafer yield will be greatly increased. Even though fault-masking techniques are very expensive in the sense of redundancy, the increased wafer yield will compensate for that. So fault-masking techniques can be applied on the cell level. Because there is some degree of fault tolerance in the cell level, the block size can be set relatively large without decreasing the wafer yield too much; the cost of testing and restructuring is therefore decreased.

In the following implementation of an 8-bit computer, one level wafer scheme is chosen because we want to focus on partitioning of the system and the design of the general purpose spare element and the restructuring issue. In the real world, it is possible to consider to employ the two level wafer scaling scheme.

Wafer scaling scheme more than two levels is out of consideration simply because redundancy has to be applied in each level ane therefore the higher level has to have repetition of the lower level

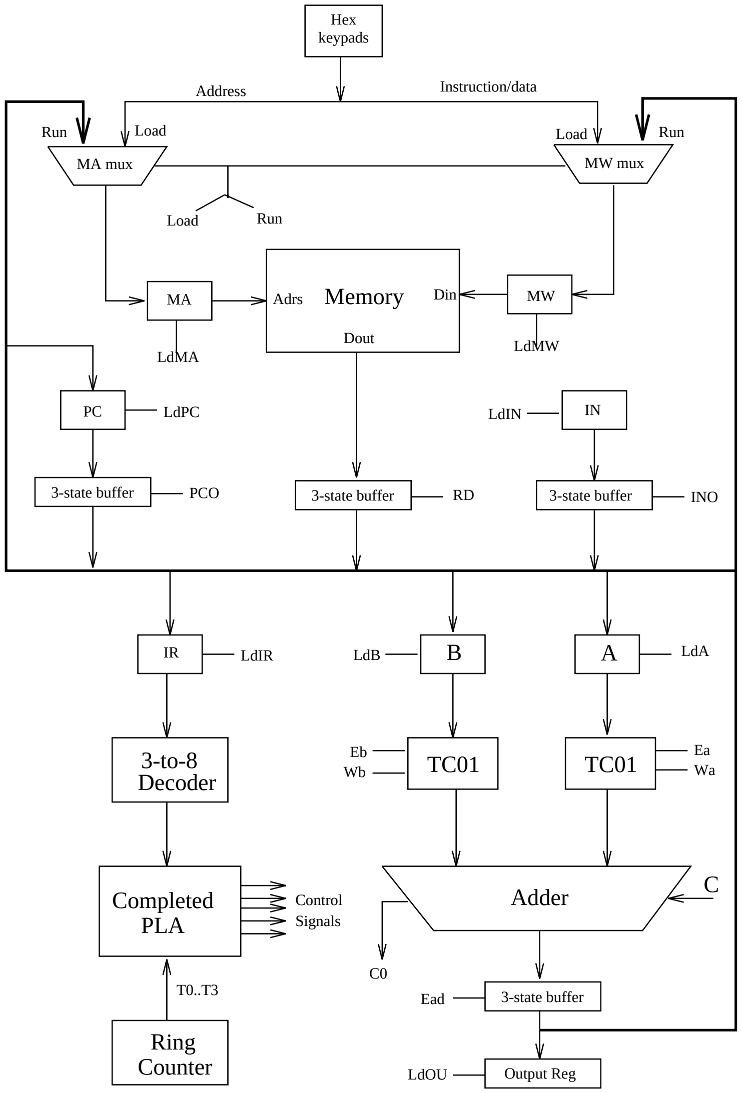

Figure 1: Block diagram of an 8-bit bus-oriented processor

redundant elements. The total amount of redundancy increases quickly with the number of levels.

5 Design of an 8-bit Fault-Tolerant Computer

An designed 8-bit computer processor is chosen as an implementation example from McCalla (1992) just for the reason of simplicity because we want to concentrate on the WSI approach rather than logic design. The results we get can be applied to other digital systems like switches, routers, etc. The block diagram of the 8-bit processor is giving in Fig. 1 (After McCalla, 1992, p677, Figure 14.9).

5.1 Partitioning of the System

The primary thought of the system partition is to view the computer as a combination of three modules: Data Path, Control Unit, Memory. The Memory Module is itself a homogeneous system and therefore can be further divided into a number of sub-modules and it will not be discussed in this study any further.

The Data Path can be implemented as two 4-bit adders. It is possible to divide the Data Path into four 2-bit adders or eight 1-bit adders, but the performance of the Data Path would be degraded because of long ripple delay as carries need to ripple from one adder to another. Two 4-bit implementation of the Data Path is consistent with the original design of McCalla's (1992).

Because the two 4-bit adders are exactly the same, redundancy is not needed for the spare element. Same partition method is applied to split the two TC01s and A, B registers into half and half. Then we will have two 4-bit Data Path submodules, which are exactly the same. There is no overhead for redundancy in spare element. The overhead for interconnections is the Carry signal, which will be implemented as a Data Path bus. Spare submodules can just simply inserted on data buses and control buses plus the Carry signal.

There are three 3-state-buffers interfacing the memory module with the data buses. There is another one 3-state-buffers interfacing the Data Path with the data buses. There are two multiplexers interfacing the data (address) buses with the memory module. If the two multiplexers are implemented as two 3-state-buffer each, then there will be eight 3-state-buffers, which will be implemented as a 3-state-buffer array. There is no redundancy overhead in spare element. There is no explicit interconnection overhead with this approach. The spare modules can be simply added at the end of the array.

5.2 Further Partitioning of the Control Unit and Spare Design

The Control Unit is composed of three submodules: a 3-8-decoder, a PLA and a Ring Counter. The amount of combinational logics in the Ring Counter is minimum. We will just leave it as it is and then apply redundant Ring Counters on it.

The 3-8-decoder is defined by Table 1 "3-8-Decoder Function Truth Table". Although the decoder is itself not very large, it can be viewed as a prototype for large decoders. From Table 1, it can be observed that every function is exactly corresponding a minterm. This is very helpful for the partitioning work because the purpose of the partition is to partition the minterms into different groups so that the spare element can be designed to implement all the partitioned group. This is possible, simply because we are going to employ restructuring techniques to isolate fault modules and then fuse spare modules on bus to replace the fault module. In doing so, we have choices to fuse either variable or its negative literal line on bus, therefore the spare module only need to consider what is the minimum number of variables needed to implement all the partitioned groups.

Table 1: 3-8-Decoder Function Truth Table

Mnemonic	Binary	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
LDADIR	000	1							
ADDIR	001		1						
SUBDIR	010			1					
STADIR	011				1				
CLA	100					1			
INP	101						1		
OUT	110							1	
JMP	111								1

The maximum number of variables equals to the number of input variables. A successful partition scheme should not only partition the functions and minterms, but also the input variables.

In partitioning, we'd like to let each partition have more or less similar size in terms of number of minterms. So, the decoder can be divided into either 2, 4, or 8 partitions. Whenever it is possible and without any other overhead, a module should be partitioned as samll as possible because samller cell areas will increase wafer yield. In the present case, the decoder is divided into 8 partitions, each of which is simply composed a minterm. The good part of the partition is that all the minterms are

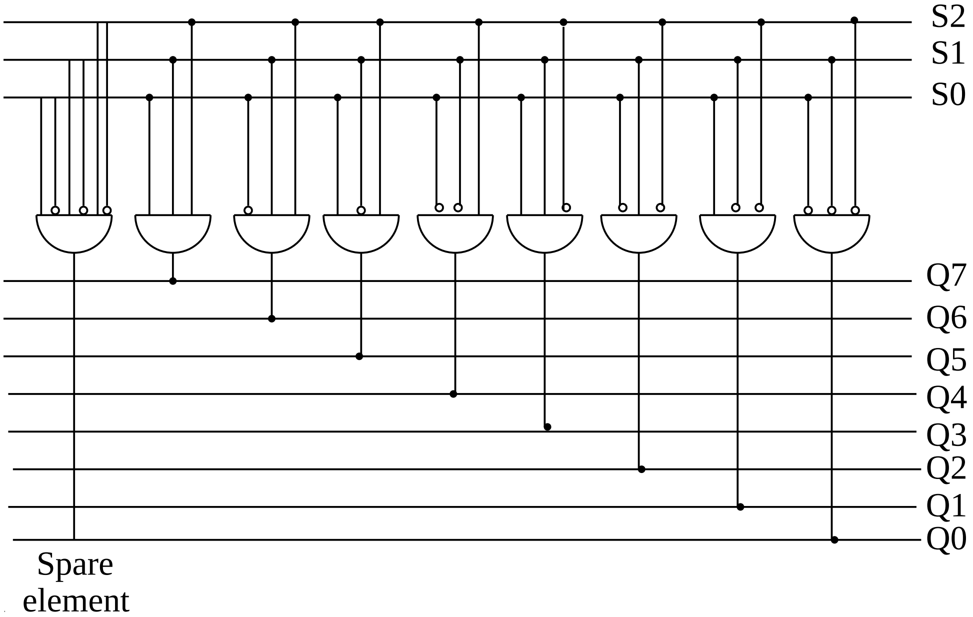

Figure 2: Fault-tolerant 3-8-decoder

partitioned without repetition. Minterms in each submodule are unique. The bad part of the partition is that this not the partition of input variables, which means that we will have all the input variables in our spare element. The spare element can be easily designed as an AND gate of three variables as shown in Fig. 2. There is no extra overhead for redundancy and interconnections. This is the best case of the general purpose spare element design method if the partitioned Data Path Unit is not count, even though there is no partition for input variables.

The partition of the PLA is much more complicated than that of the decoder. But same principles will be applied. First of all, we will try to partition the PLA so as to partition the minterms. From Table 2 (After McCalla, 1992, p675, Table14.3), it is can seen that there is no partition scheme which can exactly partition both the functions and the minterms without repetition of some minterms. If a system is partitioned with repetition of some minterms in different partition, there will be a some sort of redundancy, or partition cost, which does not contribute any to fault-tolerant functionality. If those repetitions are eliminated in a submodule by getting the signals from other submodules, the number of interconnections will be increased. Those interconnections are eventually transferred into buses in implementation. Therefore, the primary goal of partitioning is to eliminate or minimize the minterm repetition.

For the PLA as defined in Table 2, the best partition scheme is to divide all the functions into three partitions as shown on Table 2. Partition 1 includes the function PCO, LdPC, LdMA, RD, LdB and Pinc. Partition 2 includes Eb, Wb, Ea, Wa, C, EAd and LdA. Partition 3 includes LdMA, WR, LdIN, INO and LdOU. Repetition of minterms are T_2.LDADIR AND T_3.INP from the LdA function. T_3.INP is itself the function INO and it is going to be on control bus. So, T_3.INP will not cause additional interconnections. The spare module can get this signal from control bus. The other repeated minterm T_2.LDADIR will be eliminated from partition 2 by gettin a interconnection from partition 1. That is to say that the cost of the partition is only one interconnection line between partition 1 and partition 2.

The nice part of this partition scheme is that there is some sort of partitioning on the input variables. The total number of input variables is 12. Partition 1 includes 8 variables; partition 2 includes 7 variables; partition 3 includes 4 variables. So our spare element can only consider the 8 variable case.

Each partition is implemented as a submodule. The design of a universal submodule is to view

Table 2: Control Point Activation Table

P-Term	PCO	LdPC	LdMA	RD	LdIR	LdB	E_b	W_b	E_a	W_a	C	EAd	LdA	LdMA	WR	LdIN	INO	LdOU	Pinc
T_0.ALL	1		1																
T_1.ALL			1	1	1														1
T_2.LDADIR				1									1						
T_2.ADDIR				1		1													
T_3.ADDIR							1	0	1	0	0	1	1						
T_2.SUBDIR				1		1													
T_3.SUBDIR							1	1	1	0	1	1	1						
T_2.STDIR							0	0	1	0	0	1		1					
T_3.STDIR															1				
T_2.CLA							0	0	0	0	0	1	1						
T_2.INP																1			
T_3.INP													1				1		
T_2.OUT							0	0	1	0	0	1						1	
T_2.JMP	1		1																
T_3.JMP		1		1															

all the submodules as independent logics and then map their input signals each other to reduce the number of minterms in the spare submodule. Five minterms from submodule 1 can exactly map with the five minterms from submodule 2 as follows (Fig. 3): (P4, P6), (P5, P7), (P2, P8), (P3, P9), (P1, P10). The three minterms can be mapped with three of those five: (P5, P7, P11), (P2, P8, P13) and (P1, P10, P12). So, five minterms are enough for our spare submodule. The next step is to map the OR gates each other from different submodules. R5 can be mapped into R4; R6 can be mapped into R3. Other OR gates which can not be mapped each other have to be repeated in the spare submodule. They are R1, R2, R7 and R8.

5.3 Restructuring the System

The block diagram of the fault-tolerant 8-bit processor is given in Fig. 4. The basic idea is to interfacing modules or submodules through buses. Spare submodules are also hanging on the buses. When a faulty module is tested, it has to be isolated and then the spare submodule has to be connected on to buses to replace the faulty submodule. This process can be done in a number of techniques such as laser restructuring, programmable logics or self-testing and self-reconfiguring logics. The number of spare modules or submodules needed depend on fault-tolerant requirements.

The fault-tolerancy of buses is not considered in this study. It appears that the only way to deal with the bus fault-tolerancy is adding redundant buses. Let's assume we add the redundant buses already or there are no defects on buses.

5.4 Results and Discussion

From the design of the universal spare element in the 3-8-decoder and PLA, it is shown that in best case (3-8-decoder), the universal spare element accounts for just about 15% (1/8) of the original module. There is no interconnection and redundancy cost.

PLA can be viewed as an average case of the universal spare element approach. The PLA is composed of 13 AND gates and 8 OR gates. The universal spare element is composed of 5 AND gates and 6 OR gates. All together, the universal spare element accounts for only about 50% of the PLA. The cost this 50saving is only one interconnection line.

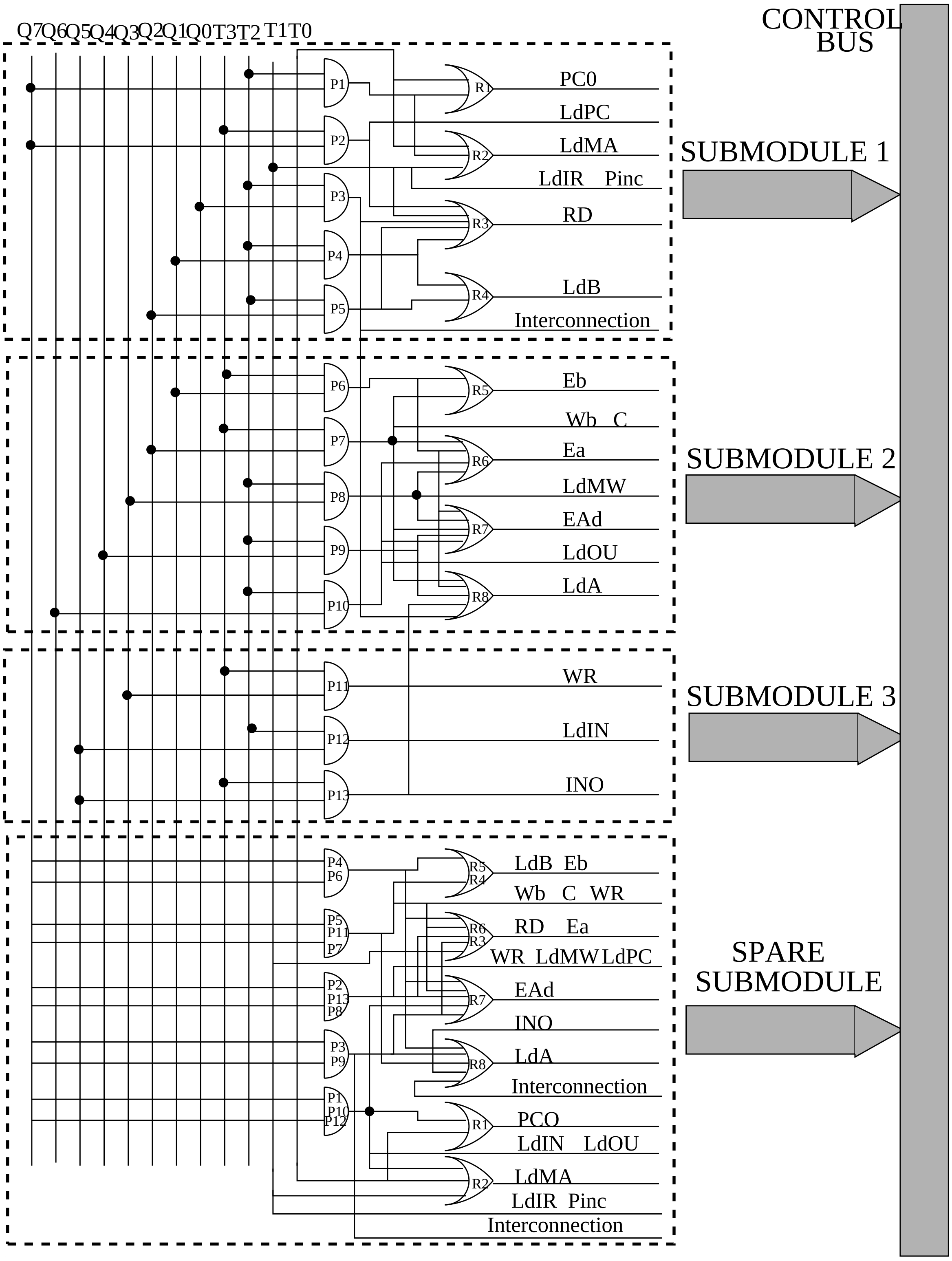

Figure 3: A fault-tolerant PLA design

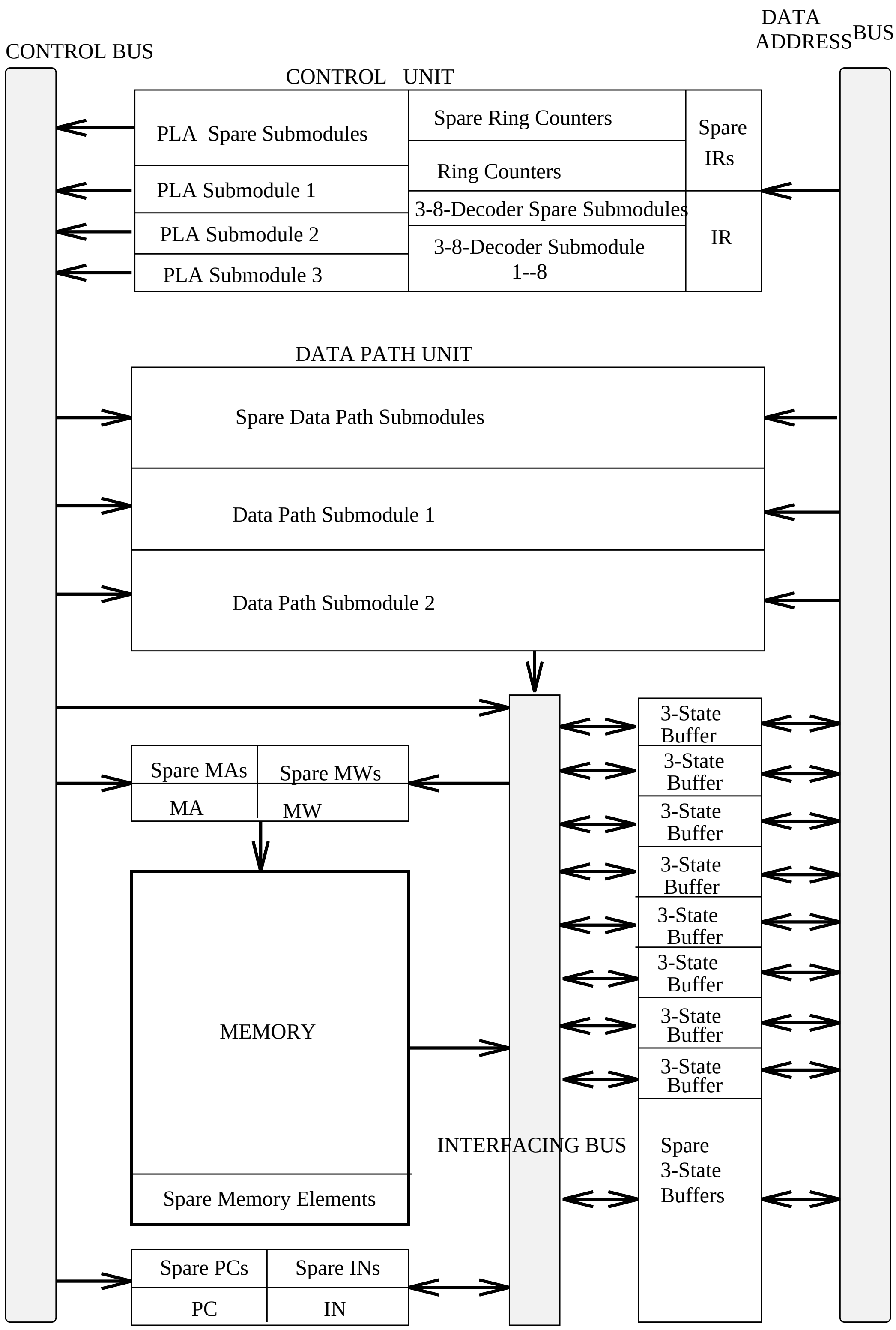

Figure 4: Block diagrams of a fault-tolerant 8-bit processor

The worst case of the universal spare element approach is difficult to imagine at this time. It probably depends on the partitioning algorithms. If all the functions of a system are very similar, let's say, in extreme case, differ only one minterm from one another, it seems to be that any partitioning scheme will have lots of repetition of minterms in each partition. But if you try to partition the complement of all the functions, you may get an very good partition scheme. The only thing can be concluded is that the worst case can not be worse than doing nothing and just using whole module as redundant elements.

6 Summary

The key step of designing a universal spare element is partitioning. The rules for partitioning is to partition a module or a system into equal parts; like we did with the multiplexers and Data Path Unit. If so, there will be no need to design any spare element. If a a module or a system can not be partitioned as equal parts, it should be partitioned as parts of similar size in terms of numbers of minterms.

In best case, a partitioning scheme should be able to partition the function set so as to cause the minterm set to be partitioned, and further cause the input variable set to be partitioned. In real world, there are alway some minterm repetition in some partitions, a good partition algorithm should minimize those repetitions.

The system we are dealing with in this study is relatively samll and simple. Both the partitioning process and the mapping process of minterms and OR gates (design spare element) are straightforward. If the approach is going to be applied on large systems, a partition algorithm and minterm and OR gates mapping algorithm have to be designed. This will be the further possible research direction.

In general, universal spare element approach is very attractive in saving redundant elements. In this study, it shows an 50% to 85% saving on redundancy.

7 References

References

[1] Brewer, J.E. Promise and Pitfalls of WSI. In *Wafer Scale Integration.* Ed. Swartzlander, E. E., Kluwer Academic Publishers, Norwell, 1989, pp. 1-30.

[2] Johnson, B.W. *Design and Analysis of Fault-Tolerant Digital Systems.* Addison-Wesley Publishing Co, New York, 1989, pp. 584.

[3] McCalla, T.P. *Digital logic and computer design.* Maxwell Macmillan Canada, Toronto, 1992, pp. 790.

[4] Raffel, J, Anderson, A.H. and Chapman, G.H. Laser Restructurable Technology and Design. In *Wafer Scale Integration.* Ed. Swartzlander, E. E., Kluwer Academic Publishers, Norwell, 1989, pp. 319-364.

[5] Sedmak, R.M. and Liebergot, H.L. Fault Tolerance of a General Purpose Computer Implemented by VLSI. In *Tutorial: Fault-Tolerant Computing. Computer Society Press.* Eds. Nelson, V.P. and Carroll,B.D., Computer Society Press, Washington, D. C., 1987, pp. 372-380.

[6] Strander, N.R. and Harden, J.C. Architectural Yield Optimization. In *Wafer Scale Integration.* Ed. Swartzlander, E. E., Kluwer Academic Publishers, Norwell, 1989, pp. 57-118.

[7] Wojciech, M. Feasibility of Large Area Integrated Circuits. In *Wafer Scale Integration.* Ed. Swartzlander, E. E., Kluwer Academic Publishers, Norwell, 1989, pp. 31-56.