EFFECTS OF USING DIFFERENT RESOURCES ON RELIABILITY AND POWER CONSUMPTION IN DIGITAL CIRCUIT DESIGN

SAYISAL DEVRE TASARIMINDA FARKLI KAYNAK KULLANIMININ GÜVENİLİRLİK VE GÜÇ TÜKETİMİNE ETKİSİ

AKIN GÖKALAN

Prof. Dr. SÜLEYMAN TOSUN
Supervisor

Submitted to Graduate School of Science and Engineering of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Engineering

January 2020

”Hayatta en hakiki mürşit, ilimdir.”

Mustafa Kemal ATATÜRK

ABSTRACT

EFFECTS OF USING DIFFERENT RESOURCES ON RELIABILITY AND POWER CONSUMPTION IN DIGITAL CIRCUIT DESIGN

Akın GÖKALAN

Master of Science, Computer Engineering Department

Supervisor: Prof. Dr. Süleyman TOSUN

January 2020, 52 pages

Designing an application in hardware under inversely competing constraints such as area, performance and power consumption with different objective functions such as reliability of the circuits is a cumbersome task. Having different versions of the same resource type handy during the design process may ease this burden since there can be several alternative resources to meet the given constraints. Although area, power consumption and speed values of commonly used arithmetic circuits can be found in previous researches, reliability analysis of these circuits that are implemented on FPGAs are not done. Thanks to reconfigurability features of FPGAs and their low cost compare to ASIC design at little production numbers make FPGAs desirable also in aerospace industry. Especially in space applications soft errors become a serious issue and requires reliability centric designs. For this reason reliability concern in FPGAs is a salient subject.

In this study, some commonly used arithmetic circuits in FPGAs in terms of speed, area, power consumption, and vulnerability to error propagation as reliability parameter are characterized to create a library. Specifically, four well-known adders and two multipliers in SRAM-based FPGA part of Xilinx's Zynq-7000 SoC platform are implemented. Then errors are injected to the configuration bits of the circuits to evaluate the error propagation. The number and the ratio of the bits which causes miscalculations are determined. The results show that different versions of the same resources can have different reliability values in addition to area, latency, and power values. Finally separately analyzed circuits are mixed in a topology consists of six adders. In this circuit impact of different resource selection on meeting constraints is presented.

Keywords: Soft Error, FPGA, Reliability, Arithmetic Circuits

ÖZET

SAYISAL DEVRE TASARIMINDA FARKLI KAYNAK KULLANIMININ GÜVENİLİRLİK VE GÜÇ TÜKETİMİNE ETKİSİ

Akın GÖKALAN
Yüksek Lisans, Bilgisayar Mühendisliği
Danışman: Prof. Dr. Süleyman TOSUN
Ocak 2020, 52 sayfa

Bir uygulamayı elektronik donanım üzerinde, devrelerin güvenilirliği parametresini göz önünde bulundurarak, alan, performans ve güç tüketimi gibi birbiriyle rekabet eden kısıtlamalar altında tasarlamak zahmetli bir iştir. Tasarım sürecinde aynı kaynak türünün farklı versiyonlarının mevcut olması bu yükü hafifletebilir, çünkü verilen kısıtlamalar birkaç alternatif kaynak arasından seçim yapılarak kullanılabilir. Yaygın olarak kullanılan devrelerin alan, hız ve güç tüketimi değerleri literatürde bulunabilmesine karşın, FPGA'ler üzerinde uygulanmış devrelerin güvenilirlik değerleri üzerine bir çalışma yapılmamıştır. FPGA'lerin yeniden programlanabilme imkanları ve düşük üretim sayılarında maliyet olarak daha avantajlı olmaları uzay ve havacılık sektöründe de FPGA'lerin yaygınlaşmasına sebep olmaktadır Özellikle uzay uygulamalarında yumuşak hatalar büyük bir problem olmakta ve güvenilirlik odaklı tasarım gerektirmektedir. Bu sebeple FPGA'lerde güvenilirlik konusu önem arz etmektedir.

Bu tezde, FPGA'lerda yaygın olarak kullanılan bazı aritmetik devreler hız, alan, güç tüketimi ve hataya karşı dayanıklılık diğer bir deyişle hata maskeleyebilme özellikleri yönünden nitelendirildi. Bu çalışmada spesifik olarak, Xilinx'in Zynq-7000 SoC platformunun SRAM tabanlı FPGA'i üzerine dört tanınmış toplayıcı devresi ve iki çarpma devresi uygulandı. Daha sonra hata yayılımını değerlendirmek için devrelerin konfigürasyon bitlerine hatalar enjekte edildi. Devrelerin konfigürasyon bitlerinden sonuca etki edecek kadar kritik olanlar bitlerin sayısı ve oranı belirlendi. Sonuçlar, aynı kaynakların farklı sürümlerinin alan, gecikme ve güç değerlerine ek olarak farklı güvenilirlik değerlerine sahip olabileceğini gösterdi. Ayrı ayrı analiz edilen devreler altı adet toplayıcı devresinden oluşan bir devrede karıştırıldı ve kaynak seçiminin kısıtlamaları karşılamak bakımından yaptığı iyileştirmeler gösterildi.

Anahtar Kelimeler: Yumuşak Hata, FPGA, Güvenilirlik, Aritmetik Devreler

ACKNOWLEDGEMENTS

First and foremost, I would like to sincerely thank to my supervisor Prof. Dr. Süleyman Tosun for his time, patience and for all his valuable guidance at every stage of my research along this long way. Without that support, It would have been impossible to write this thesis.

I would like to thank Tubitak who supported the research under the project number 116E095.

I am grateful to Roketsan and my manager Mazhar Gökhan Özkeser for their tolerance and support in this cumbersome process.

I would like to thank to my friends Gün Demirbaş, Enes Akyurt and Sibel Arslan for their ideas in every phase of this research.

Finally, I express my most sincere gratitude to my family and friends who supported me morally with their best wishes.

CONTENTS

FIGURES

Page

TABLES

1. INTRODUCTION

1.1. Motivation

A Field Programmable Gate Array (FPGA) is an electronic device that consists of a large number of configurable logic blocks (CLBs), programmable routing switches that connect the CLBs, and input-output (IO) pins [19]. In FPGAs, logic functions are realized by means of CLBs, which in turn are composed of look-up tables (LUTs) that store the truth tables of the functions, multiplexers, and flip-flops. The prevalence of FPGAs are growing in both industry and academia due to their advantages such as reduced time-to-market, reconfigurability, and the ability of parallel processing. They are also preferred as the choice of the design platform in aerospace industries. They are playing a critical role in the rover missions to Mars [20]. It is a new phenomenon for FPGAs to be in harsh conditions. FPGA vendors are also working on the reliability of the FPGAs to increase the trust and usability of them in aerospace applications. They develop radiation-resistant FPGA families and also integrate automated reliability oriented design techniques in the software program flow[20]. However, under ionizing radiation, the configuration bits of the FPGAs tend to flip, which is called SEU (single event upset) [21]. An SEU may propagate through the circuit and cause an unexpected behaviour. On the other hand, it may not change the output of the design due to the error masking capabilities of the combinational circuits. In other words, the internal structure of the circuit and the input pattern applied determine whether or not the behaviour changes in the presence of an SEU.

SEU by definition is a change of state in a semiconductor, which does not permanently change the behaviour of the circuit. If such a non-persistent error cause the data stored in memory to be erroneous even for a short period of time, all operations using that data will yield erroneous results until the data is updated. These types of errors in digital systems are called transient errors or soft errors (SEs). Since the behaviour is determined by the configuration bits in SRAM-based FPGAs, an SEU cause a permanent change of behaviour until restart.

1.2. Contributions

In this thesis, a new method for testing the SE vulnerability of the arithmetic circuits implemented in SRAM-based FPGAs is presented and in this respect, four well-known adders and two multipliers are taken into consideration where the choice of the FPGA platform is Xilinx Zynq-7000 SoC. In the design flow, firstly, the adders and the multipliers are characterized in terms of the occupied area in the FPGA, speed (i.e., latency), power consumption, and the vulnerability to SEs as a reliability metric. Later, different adders and multipliers are employed to implement the same function that is customly generated and used as a case study to illustrate how the different versions of the same resource affects the overall reliability and the other metrics (i.e., area, latency, and power) on the final design. In the last step, errors are injected into the configuration bits of the FPGA to observe if they affect the results.

The contribution of this work can be summarized as what follows below:

- A new methodology that can be used to test the vulnerability of circuits against SEs on FPGA-based implementations is proposed
- A new library that consists of four adders and two multipliers is introduced. The resources in this library are characterized in terms of the area, latency, power consumption, and the SE vulnerability metric that represents the reliability. I strongly believe that such a library is exceedingly useful for further research on high level synthesis (HLS) of integrated circuits on FPGAs and ASICs (Application Specific Integrates Circuits) as well.
- A case study is presented which shows utilizing different versions of the same resource in a design helps meeting the constraints and yielding better optimized circuits. I used a custom generated function that uses all additions for the case study and implemented it with various adder types to show how different design parameters are affected.

1.3. Organization

The rest of the thesis are organized as follows: In Chapter 2. previous research done on reliability is discussed. Research on the performance of arithmetic circuits in ASIC and FPGA is reviewed. Studies to increase reliability of well-known arithmetic circuits is evaluated.

In Chapter 3., the overview of the system architecture, the technological components used in the evaluations and concepts that are related to the research are introduced. The details of the test methodology are presented in Chapter 4.. In Chapter 5., the test results for the selected adders and the multipliers along with a case study that uses the characterized library are discussed. Finally, Chapter 6. concludes the thesis by summing up the work and results.

2. RELATED WORK

Especially in space applications, reliability is a big concern. Lots of research is done on the subject. Even though most of the research is done targeting ASIC design there is also some valuable research targeting FPGA. Firstly some of the researches which considering general reliability oriented design techniques and reliability analysis models are covered. Then the researches that are done to increase the reliability of well-known arithmetic circuits are summarized. Even though most of them target ASIC design they provide insight into the problems and possible solutions.

Triple modular redundancy (TMR) combined with a voter mechanism is a well-known method to increase the reliability and detect SEs [22]. Although this method drastically improves the mean time to failure (MTTF), the area and the power consumption increase threefold compared to the original design. Therefore, some prior research endeavors to optimize the TMR solution. Pratt *et al.* [23] analyzed the elements of the circuit and applied TMR only to the highly vulnerable parts of the design. They classified the configuration bits of the design as sensitive bits and persistent bits. Sensitive bits are the bits that impact the result and behaviour. When sensitive bits are corrected circuit recovers from error state and works as expected. Persistent bits are the bits that cause a fault in the state of the design which can not be recovered even if the error is corrected. They realized persistent bits are the feedback structures of the circuits. When they propagate incorrect values state machine itself is broken and cannot recover to normal working conditions. They applied TMR to only persistent bits to increase reliability while sacrificing little on power and area. They compared reliability results of non-mitigated design, partial TMR design and TMR design. They manage to increase reliability a hundred times while sacrificing only %20 area using partial TMR.

Another well-known method is data scrubbing (DS) [24]. DS is the process of scanning all device memory in some time intervals to correct the detected errors. It requires storing the original configuration or part of the original configuration data to replace the corrupted frames. Even though every frame of the FPGA bitstream has error correction code (ECC) bits, it may not be enough since ECC codes are only capable of correcting a single bit or two adjacent bits in a frame. Apart from the ECC codes, the bitstream is also protected by cyclic redundancy check (CRC) codes; however, CRC codes are only beneficial for integrity check and they are not useful for any error correction operation [17].

Ostler *et al.* test the effects of SEUs in SRAM-based FPGAs. They adopt a statistical method to estimate the reliability of the FPGAs at different orbit levels [25]. FPGA design they analyzed, is protected by scrubbing and TMR. Scrubbing is scanning configuration bits of FPGA and correcting the flipped bits thanks to ECC codes as explained before. They assume the scrubbing period is 15 ms. Their calculations are based on the idea that if two errors occur in a single scrub cycle the circuit produces erroneous results and fails since they adopt TMR. Using the orbital radiation levels they estimated SEU probability. In the light of this information, they created their statistical results. Although the reliability results of the tested FPGAs are very promising, the authors conclude that the FPGAs are not suitable devices to be used in every environment but is an available choice in many environments.

In another work [26], the researchers used reliability estimator tools to infer the reliability of the hardware components in FPGAs. This work shows how each component contributes to the overall system failure rate. They calculated the device reliability by using the reliability contribution of each hardware in the FPGA. They proposed hardware redundancy for less reliable parts of FPGA to increase the overall reliability of the device. They compared the reliability results of an FPGA that has one spare memory block and an FPGA that does not have any spare hardware.

Another study aims to find a model to measure the reliability of designs by analyzing hardware description language (HDL) code [27]. There are existing reliability models for conventional programming languages. However standard programming languages work sequentially. Therefore they do not adapt well to the concurrent nature of the HDLs. The presented method in [27] extends the reliability model of a standard programming language to handle the error propagation cases that stem from the parallel nature of the HDLs. The proposed model is tested and validated with a real design to prove its practical correctness.

Some previous research aimed to gain on reliability by modifying the fundamental architecture of circuits itself. In [28], authors improve the overall reliability by adding very limited additional hardware. They investigate the full adder circuit targeting application-specific integrated circuits (ASICs). By calculating the results with the original inputs and their complements in the full adder, they manage to use different paths on the full adder circuit resulting in the same output. They finally check the results and determine if a transient error emerged during one of the calculations. With the addition of negligible extra hardware, they manage to get reliability results as if there is dual modular redundancy. However, the modified full adder requires two clock cycles to produce two distinct results since two different paths have some common elements. For this reason, to prevent time redundancy, they propagate the result of the regular full adder path as soon as it is available not to make the remaining circuit wait. Afterward, they calculate the result of the alternative path. When the results are compared, they inform the remaining of the circuit if an error occurred. Therefore, the circuit does not suffer from the time redundancy in normal working conditions but only when an error emerged. In [29] authors proposed a self-repairing full adder design which is capable of repairing itself using extra logic responsible for validating the result. Extra logic is only enough to manage a single fault at a time. By their method, they significantly improved the reliability of the full adder while sacrificing very little on area compare to TMR. In [30] authors present a method that can make the defected hardware operate correctly. To show the use case of their method, they propose a redesigned version of the Kogge-Stone adder, which can run correctly even a fabrication defect exists in the circuit. Using the mutually exclusive nature of even and odd bits of the Kogge-Stone adder, they reconfigure the hardware to make it only use the healthy bit set of the adder if a manufacturing fault is detected in even or odd bits. This approach does not decrease the throughput since it can run at the same or even higher clock rates; however, it requires two clock cycles to produce a result if the circuit is reconfigured to use only even or odd bit sets.

There are also some research in the literature when it comes to investigating the performance of the arithmetic circuits. Daphni *et al.* compared commonly used parallel prefix adders (PPA) including Kogge-Stone adder and Brent-Kung adder on FPGAs, in terms of power, speed and area. In this respect, they presented their results in 16 and 64-bit variations of the circuits. Vitoroulis *et al.* [31] expanded study presented in [32]. They presented the area and the speed results of commonly used PPA adders. They implemented adders with input sizes of 16, 32, 64, 128, and 256 bits. They also produced two results: the first one is the results of the synthesis with the area optimization setting, while the latter is the synthesis results with the speed optimization setting. Jayanthi *et* al. [33] and SaiKumar *et al.* [34] did not restrict their research on PPA adders and further analyzed high speed VLSI adders. Mohanapriya *et al.* [35] investigated multiplexer-based adders using Cadence for 180nm technology. They reported the performance of the circuits in terms of speed, area and power dissipation. In [36], authors implemented various types of adders at 90nm, 130nm, and 180 nm technology. They extracted all the nets in the netlist and injected transient error to all of them. They checked if the injected error causes an error on the final result. By doing so, they tried to find the architectural vulnerabilities of the commonly used adder topologies targeted for ASIC designs. However, to the best of the knowledge reviewed in the literature, there is no previous work that compares the different implementations of the same arithmetic circuits using SRAM-based FPGAs as far as the reliability is concerned.

3. BACKGROUND

In this section, the necessary hardware and software components of the experimental setup for the proposed method is proposed. Moreover the concepts that are closely related to the study is mentioned. Specifically the architecture of the FPGA board, FPGA configuration bitstream, essential bits for testing, intellectual property (IP) that is used for SE propagation tests and the circuits that are evaluated in the thesis are explained.

3.1. Zynq-7000 SoC

Xilinx's Zynq-7000 SoC (System-on-Chip) board is selected as the implementation platform [37]. The reason for this selection is that Zynq-7000 SoC has dual Arm Cortex A9 cores and 28nm Artix-7 based programmable logic (FPGA) allow developers to easily program and control the behaviour of the reconfigurable logic via the Arm cores.The architectural overview of this board is given in Fig. 3.1. As can be seen in this figure, the board has two main components that are connected to each other via AXI buses: the processing system (PS) and the programmable logic (PL). Zynq has 256 kb on-chip memory (OCM) [38]. External DDR bus is available for connecting external memory. Since PS has two level cache architecture execution speed is not affected significantly. First level cache (L1 cache) is propriteary to core itself. Size of L1 cache is 32kb. L2 cache is a shared cache between two processor cores. Size of L2 cache is 512kb. Apart from these memory resources there is Block Ram available in PL. It is available to be used as external memory via Axi bus. Zynq-7000 has also two ports that allow developers to access and change the FPGA configuration bits: the processor configuration access port (PCAP) and the internal configuration access port (ICAP). While PCAP is used to access the FPGA configuration bits from the PS, ICAP is used to access the FPGA configuration bits from the PL part.

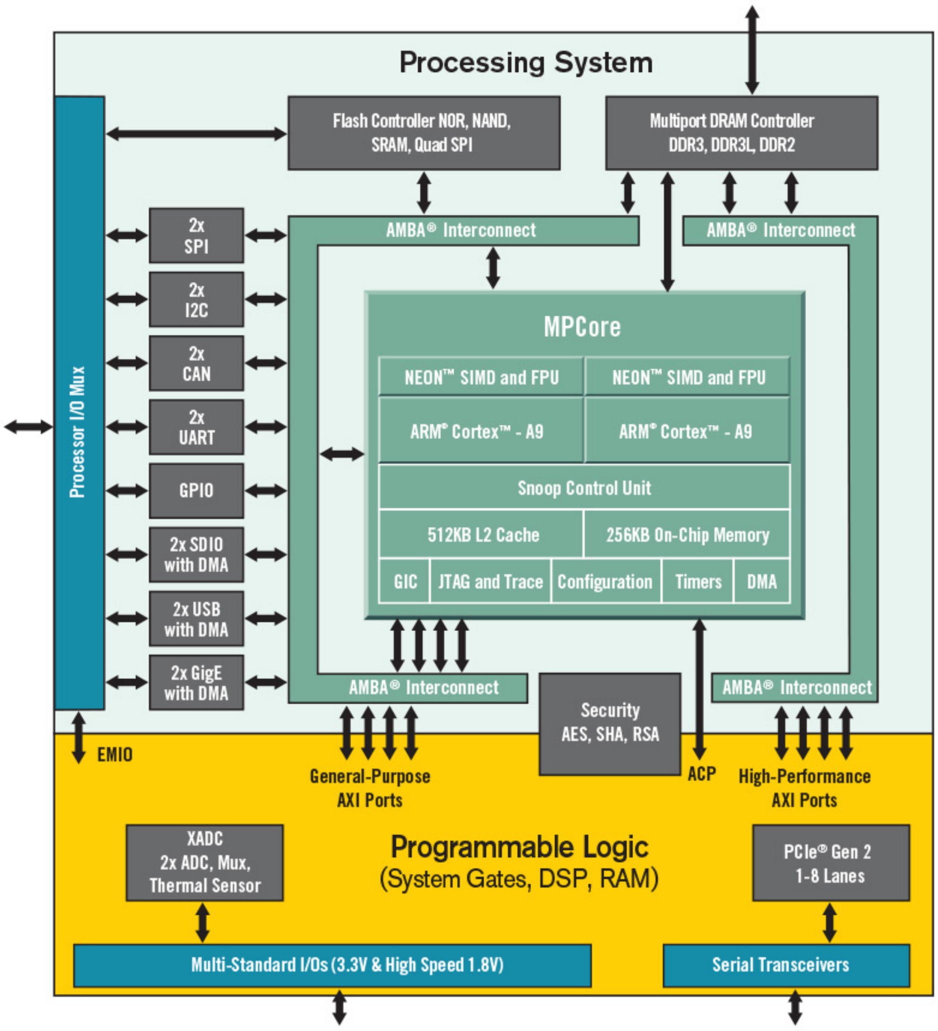


FIGURE 3.1: Architectural Overview of the Zynq-7000 SoC Board [1].

3.2. FPGA

Field Programmable Gate Arrays (FPGAs) are reconfigurable silicon devices that can be programmed at any time. For little production numbers, they are cost-effective compared to application specific integrated circuit (ASIC). Also, design in FPGA can be updated while it is not possible in ASIC. However, all these advantages come with a cost. FPGAs are less efficient in terms of speed, area and power consumption compared to ASIC. Even though the staggering innovation speed of FPGAs result in quite fast hardware compare to their predecessors, they are still not a match to ASIC in terms of performance. The main obstacle of FPGAs in performance is reconfigurability which is their main advantage. FPGAs consist of three main components. These are:

- Configurable Logic Blocks
- Programmable Routing Elements (Interconnect)
- I/O Blocks

An overview of FPGA architecture and elements are presented in Figure 3.2.

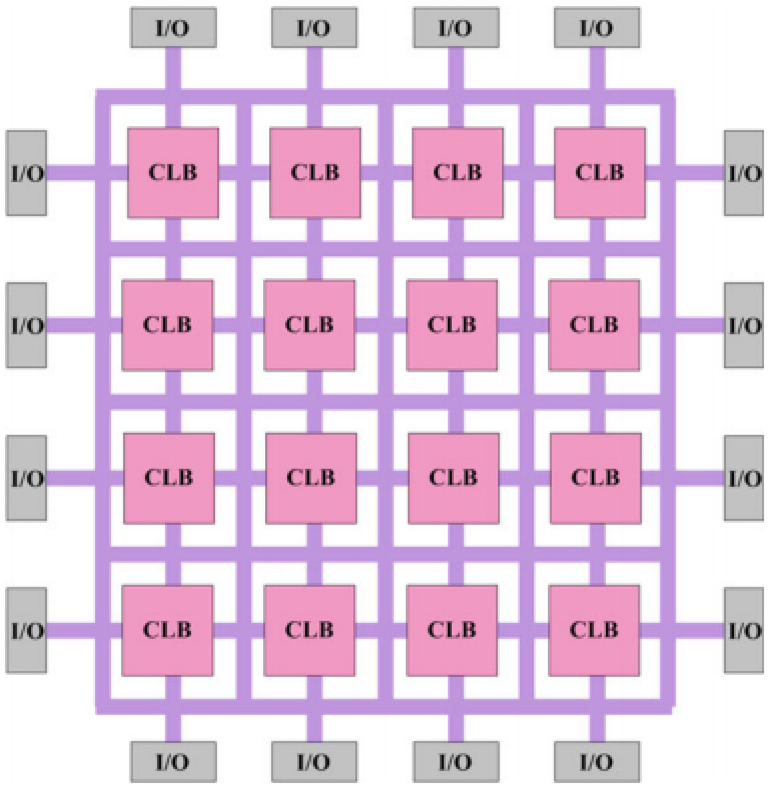


FIGURE 3.2: Architectural Overview FPGA. [2]

Configurable logic blocks (CLBs) are the main element that handles combinational logic in FPGAs. Designing of a CLB is a process restricted by constraints. A very capable CLB unit causes waste of power and resources, on the other hand, simple CLB design can not implement even simple functions also causes waste of resources. Under these constraints, FPGA vendors released a lot of CLB designs. Specifically, in the remaining of the subsection, CLB structure of Xilinx 7-Series FPGAs will be explained since it is the model which is used in the test setup. In 7-Series FPGAs one CLB unit consists of two independent slices. They have their own carry chain circuits. They are connected to other slices vertically. Overview of the slices connection and placement is shown in Figure 3.3.

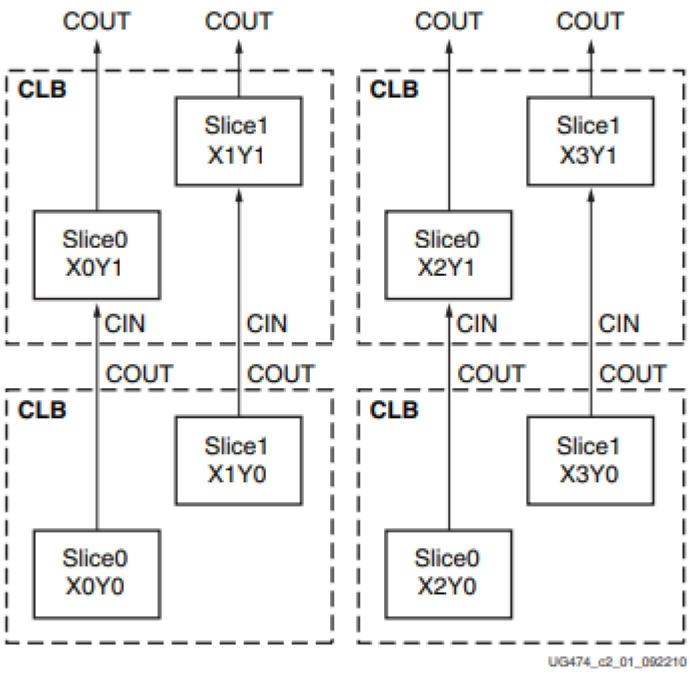


FIGURE 3.3: Placement of Slices. [3]

Each slice contains the following elements:

- Four Lookup Tables (LUT)
- Eight Storage Elements
- Multiplexers
- Carry Logic

Each LUT has six inputs and two outputs. Their main task is implementing combinational logic and propagating output to the next step. Multiplexers are the elements that help LUTs to implement logic by supplying extra functionality. If a circuit requires more than six inputs two slices are used together to satisfy the required logic. Storage elements are flip-flops that are working synchronously with the given clock. Carry logic is used to create carry signals faster. They can also perform and operations. Carry logic can form a carry chain by cascading their inputs and outputs in collaboration with LUTs. They use carry lookahead logic to generate carry signals. The width of a carry chain is limited with the number of slices that exist in an FPGA column. Detailed view of slice structure is presented in Figure 3.4

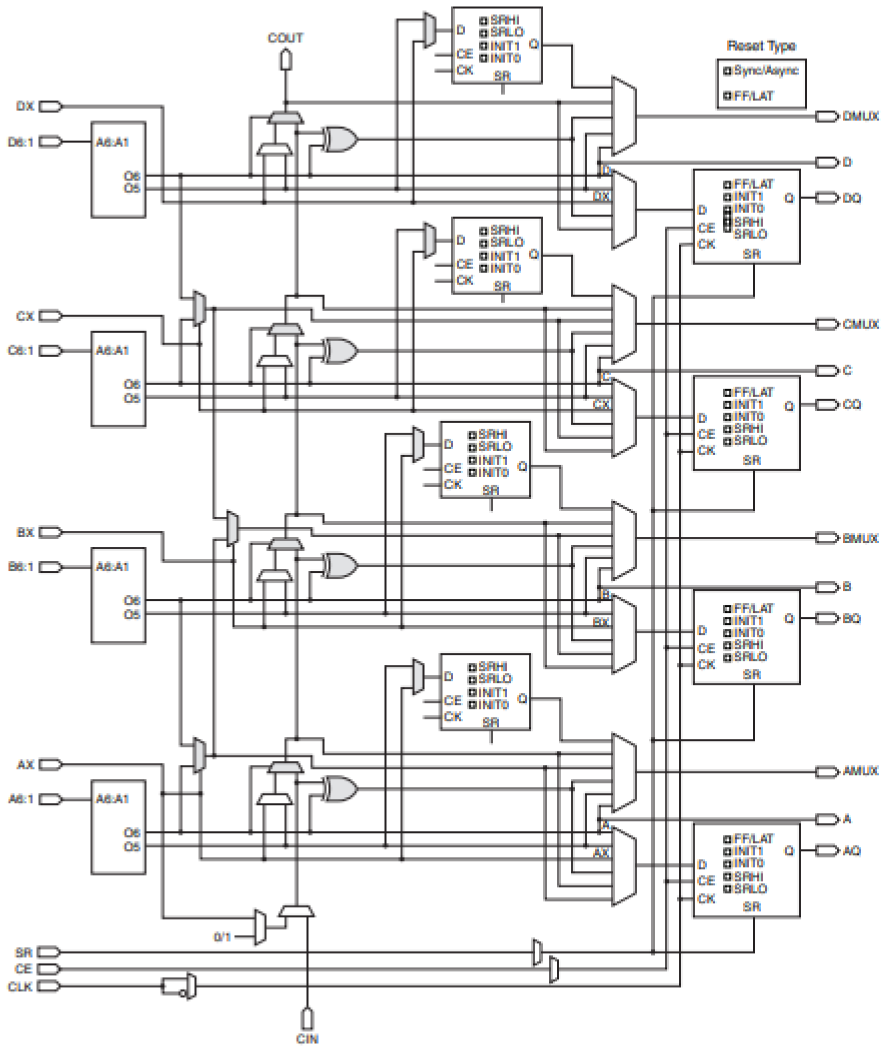


FIGURE 3.4: Slice Architecture. [3]

3.3. CAD Tool and Design Flow

ISE Design Suite v14.7 is used in the test setup. ISE design suite is especially targeted for Spartan family FPGAs. Xilinx recommends working with Vivado Design Suite for Zynq. However, in Vivado Design Suite prioritized essential bit flow is not implemented yet. Therefore, ISE design suite is chosen as the computer-aided design (CAD) tool. The flow of FPGA design consists of four main stages. These are:

- Synthesis
- Mapping
- Place
- Route

Mapping, place and route all together called as implementation phase. A typical HDL development process is shown in Figure 3.5

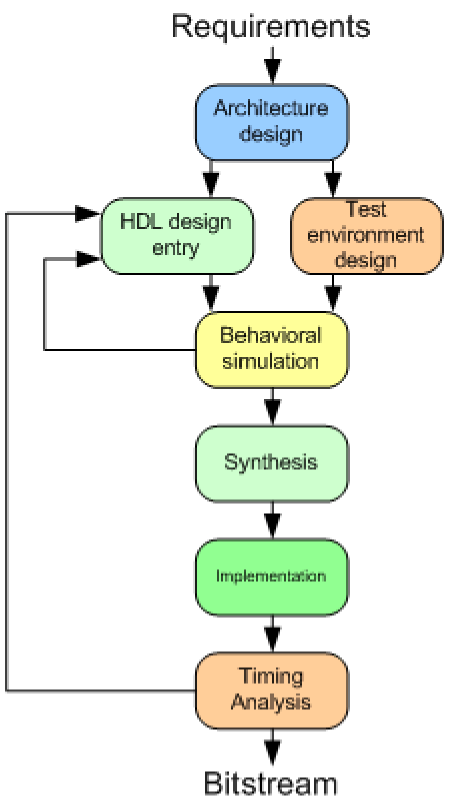


FIGURE 3.5: Typical FPGA Design Flow [4]

Synthesis The synthesis process means converting the HDL code to a netlist. The output of the synthesis process consists of boolean functions between wires, flip-flops and interconnections[2]. Synthesis tools also able to make optimizations on the boolean functions.

Mapping The mapping process takes the output of the synthesis and modifies it to the target hardware. As mentioned in section 3.2. input and output number of LUTs depend on the targeted hardware. The mapping process arranges the boolean functions that are produced in the synthesis phase to fit the hardware capabilities. The dilemma of mapping is the depth of the resulted logic[2]. For high input logical functions depth of the logic may get high and cause performance problems. Decreasing depth brings extra area requirements. Mapping algorithms put an effort to balance these two constraints in acceptable processing time.

Place Placing is the step where the tool associates synthesized logic with the existing physical elements in the hardware. The placement tool must be aware of the hardware and its extra features. It supposed to use these features effectively to save from area, speed and power. Another challenge of placing is keeping the wiring at minimum[2]. To achieve this objective placement algorithms use some approaches to keep the related logic close to each other. Details of the algorithms are not covered in this study.

Route The routing phase handles the connection between the already placed logic. Since interconnection resources are also limited on FPGAs, not all interconnection wires can have the shortest path to their target destination[2]. Therefore, the challenge of route algorithms is managing wiring without collision while sacrificing minimum at connection length.

3.4. Bitstream Structure

Bitstream is the collection of the configuration bits that are loaded into the FPGA to realize the desired logic. As a result of its technology protection policy, Xilinx does not reveal the relationship between the logical placement and the bitstream. Instead, it provides linear address scheme. The linear address does not represent a physical address; however, it provides the ability for a user to pinpoint a specific logic in the design. The linear addressing scheme also uses frames and words. The whole bitstream is firstly divided into the frames. Frames are then divided into the words, where each word consists of a specific number of bits. For example, the bitstream in Zynq-7000 consists of 7948 frames. Every frame consists of 101 words and every word has 32 bits. Therefore, a total of 25,687,936 bits exists in the bitstream destined for Zynq-7000. At this point, it must be noted that the frame numbers and internal structure of the frames change among different FPGA models.

3.5. Xilinx Essential Bits Technology

The essential bits of a design are defined as the bits that contain important information about the design. Xilinx uses a special algorithm to identify the essential bits among the entire configuration bits. If there is an SEU on an essential bit, the configuration of the circuit changes as a result of the upset. However, this erroneous configuration bit might not change the functionality of the design. Xilinx's essential bits technology aims to provide the users the ability to mark the essential bits in the design. The prioritized essential bit technology takes it one step further and allows the users to mark the essential bits only for the selected parts of the design. Xilinx synthesis tools create three classes of data files for the essential bits with the extensions of .ebc, .ebd and an extra bit file after a successful implementation. The bit file is the configuration file that is loaded into the FPGA. ebd file stores the configuration bits in the ASCII format. ebc file is a marking file of the essential bits in the ASCII format. ebc and ebd files have the same size. If a bit is 1 in the ebc file, then the corresponding configuration bit in the ebd file is marked as essential. While the essential bits provide a good opportunity for designers to reduce the failure in time (FIT) by extra precautions, they can be employed for analyzing the SE susceptibility of a circuit.

There are three options to filter the ebc file. These options are none, mask and enable [18]. If none is selected as the filtering option tool produces an ebc file that contains all essential bits for the whole design. If mask is selected as the filtering option tool produces an ebc file that contains all essential bits except the essential bits that belong to the area marked by the user during implementation. If enable is selected as the filtering option tool only extracts the essential bits of user marked area. However, the enable option also extracts essential bits that are important for the whole design to work such as the clock tree. Therefore in this study, the essential bit map is obtained by the difference of ebc file that is generated by none option and ebc file that is generated by mask option. Since common elements that are also important for other parts of the design are not masked in the ebc file that is generated by mask option, the difference of two ebc files produces the essential bit map that contains the essential bits unique to user marked area.

3.6. Soft Error Mitigation IP

Xilinx provides a soft error mitigation intellectual property (SEM IP) to detect and correct SEs occurring in the configuration bits of the FPGAs. The SEM IP uses error correction codes (ECCs) for detection and correction of the erroneous bits. The ECC codes carry enough information to correct one-bit flip or two adjacent bit flips in a frame. If more than one non-adjacent bits flip, the ECC codes cannot correct the errors. In such a case, the frame must be reloaded utilizing the ICAP or PCAP. The SEM IP is capable of reloading a frame partially as shown in [39]. The SEM IP is also able to classify errors by comparing the locations of the errors using the essential bits map.

Another important feature of the SEM IP is that it allows the designer to inject errors into the configuration memory and emulate the errors as SEs. In this research, only the error injection capabilities of the SEM IP is used. SEM IP is configured to accept commands through the UART interface for the error injection and error mitigation capabilities of different implementations are tested.

3.7. Pblocks

Pblocks are the user-drawn areas in the FPGA logic and used to supply specific place and route constraints to a specific module in the design. Pblocks are used to implement different circuits in a specified FPGA location.

3.8. Full Adder

Full adder is a very primitive unit in adders. It has three inputs and two outputs. Two inputs are the operands and the third one is carry. Outputs are the carry and the sum. The full adder is designed to create adders at the desired width by stacking and connecting carry outputs to the next full adder unit. Representation of full adder is shown in Figure 3.6.

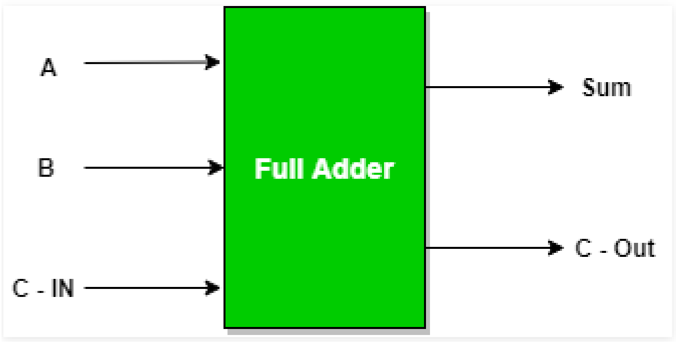


FIGURE 3.6: Full adder

3.9. Ripple Carry Adder

Ripple carry adder is a very primitive adder. As explained in section 3.8. ripple carry adder consists of cascaded full adders at the desired width. In Figure 3.7 four bit representation of ripple carry adder is presented. Ripple carry adder is a simple and easily scalable design however it is slow because carry signal needs to propagate through all full adders for a valid result.

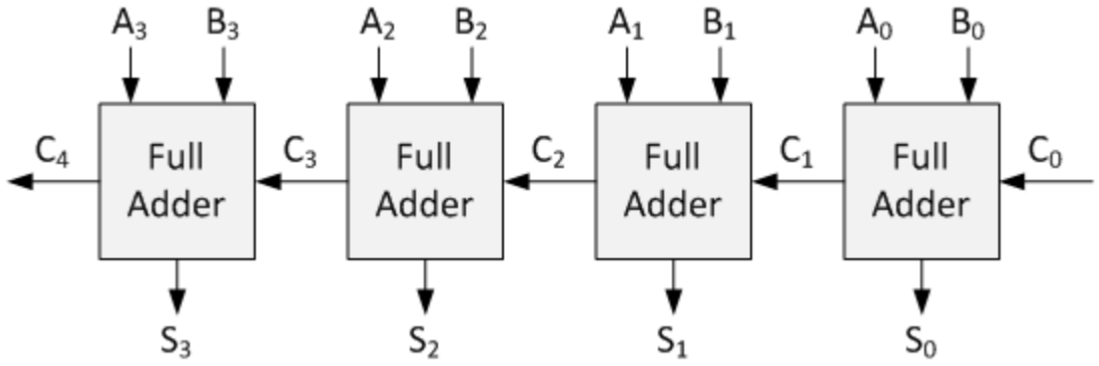


FIGURE 3.7: 4 bit ripple carry adder [5]

3.10. Carry Lookahead Adder

As explained in section 3.9. ripple carry adder is slow because of carry propagation delay. Carry lookahead adder is a more complex hardware that targets to decrease carry propagation delay. Carry lookahead adder generates two signals called carry generate (G) and carry propagate (P) which is calculated using the following formulas:

$$G_i = A_i.B_i$$

$$P_i = A_i \oplus B_i$$

Therefore carry and sum signals can be calculated using following formulas:

$$S_i = P_i \oplus G_i$$

$$C_{i+1} = C_i.P_i + G_i$$

Using the equation above the carry signals are calculated as follows:

$$C_1 = C_0.P_0 + G_0$$

$$C_2 = C_1.P_1 + G_1 = (C_0.P_0 + G_0).P_1 + G_1$$

$$C_3 = C_2.P_2 + G_2 = (C_1.P_1 + G_1).P_2 + G_2$$

$$C_4 = C_3.P_3 + G_3 = C_0.P_0.P_1.P_2.P_3 + P_3.P_2.P_1.G_0 + P_3.P_2.G_1 + G_2.P_3 + G_3$$

As it is seen from the formulas carry signals are not dependant on the previous carry signal but only to C0. Thanks to the fast carry generation logic carry-lookahead adder performs better than ripple carry adder at the cost of extra logic. Visual representation of carry-lookahead adder is shown in Figure 3.8.

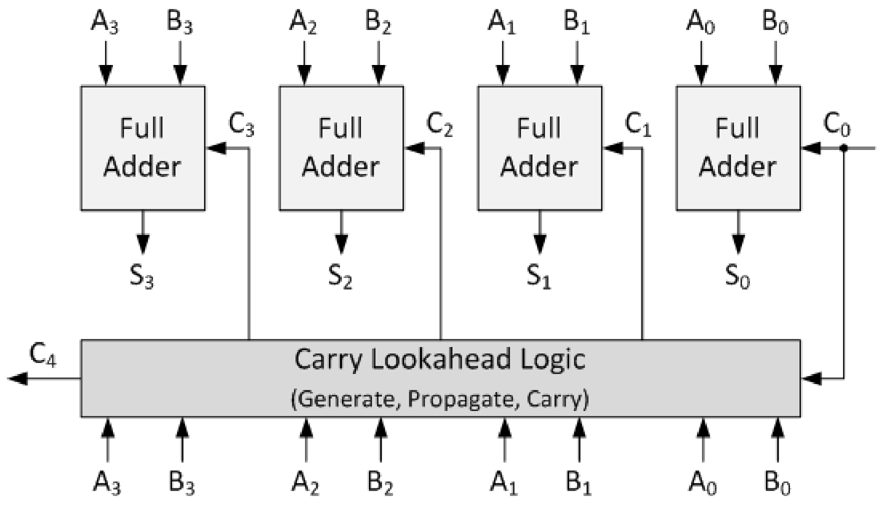


FIGURE 3.8: 4 bit carry lookahead adder [6]

3.11. Brent-Kung Adder

Carry lookahead adder implements carry logic faster compare to ripple carry adder. However, its carry logic path delay linearly increases with n which is the number of bits of the operands. Brent Kung adder is one of the parallel prefix adders(PPA) which reduces the complexity to the logarithmic level. Brent Kung adder also calculates P and G signals as it is in carry-lookahead adder. However brent-kung adder calculates P and G groups as a tree. At every stage, another bit group of P and G values is calculated. A brief look to brent-kung adder P and G signal generation is shown in Figure 3.9.

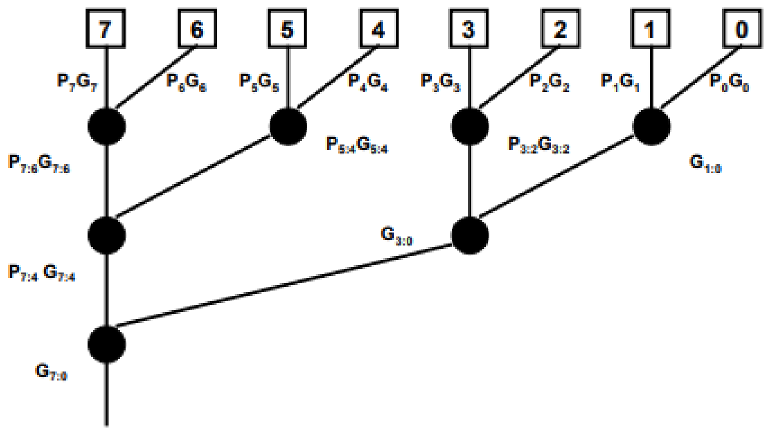


FIGURE 3.9: 8 bit brent-kung adder P and G signal generation [7]

After P and G values are calculated carry signals are being generated also using a tree-based approach. Complete working steps of Brent-Kung adder is shown in Figure 3.10. Complexity of the brent kung adder is given as (2)(logN) - 2

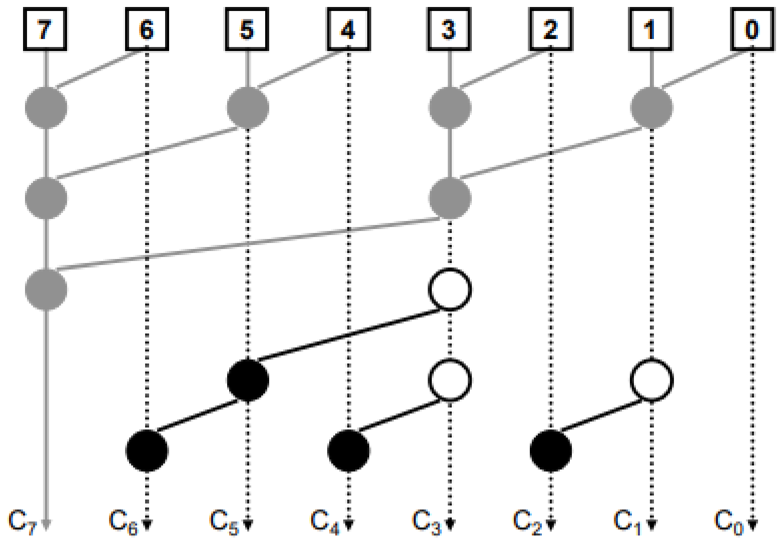


FIGURE 3.10: 8 bit brent-kung adder [7]

3.12. Kogge-Stone Adder

Kogge stone adder is also one of the PPA family. It is more complex and faster than Brent-Kung adder at the expense of power and area. The complexity of the kogge-stone adder is given as log(n). Carry generation tree of Kogge-Stone adder is given in Figure 3.11.

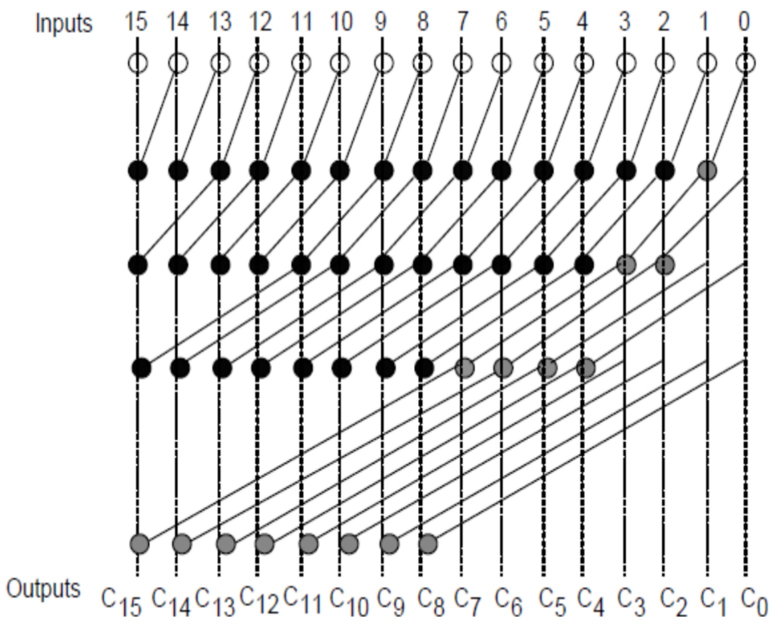


FIGURE 3.11: 16 bit kogge-stone adder [8]

3.13. DSP

Signal processing and image processing functions require a lot of arithmetic operations, especially multiplication operations. Multipliers implemented in the logic part of FPGAs are slow and require a lot of space. Therefore, Xilinx first introduced embedded multiplier in their Virtex II FPGAs. By developing the special multiplier considering the demands of the industry, modern DSP blocks emerged. Modern DSP blocks are capable of handling a wide range of operations.

DSP48E1 slice is used in the tests. DSP48E1 is introduced with 7 Series FPGAs however is being used within all families. The sheer performance of DSP blocks which can run up to 720 MHz are demanded more and more by consumers. For this reason, the number of DSP blocks in FPGAs is increasing. The number of DSP units in FPGA families are shown in Figure 3.12.

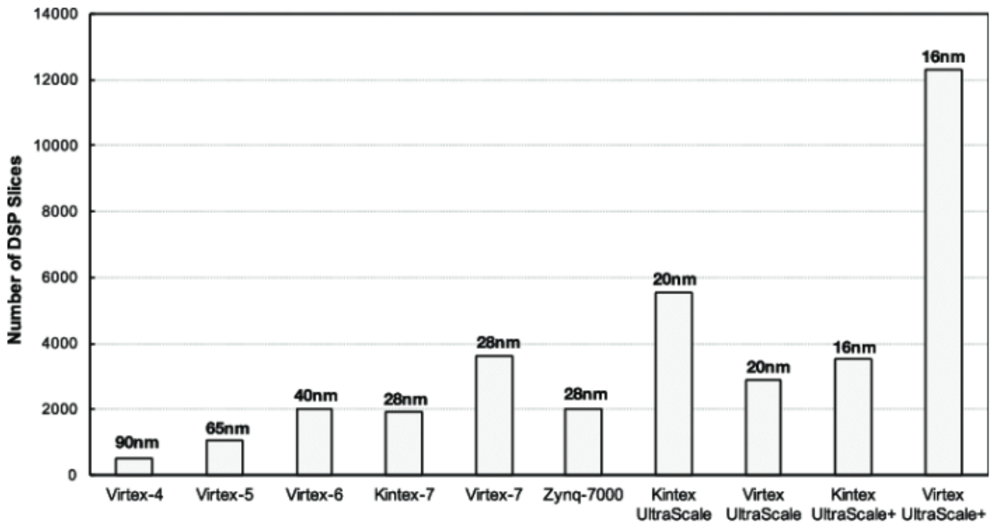


FIGURE 3.12: Number of DSP Blocks in Xilinx FPGA Families [9].

The capabilities of the DSP48E1 slice is given as follows [10]:

- 25 bit preadder with a register to increase input capabilities.
- Balanced pipelining while switching between addition and multiplication operations.
- 18x25 bit multiplier

- Internal cascade capability using two DSP slices which supports up to 96 bit operations.
- Bitwise logic operations.
- Pipelining ability in the input and output pins.
- Capability to handle shift operations.

The architecture of DSP48E1 slice is given in the Figure 3.13

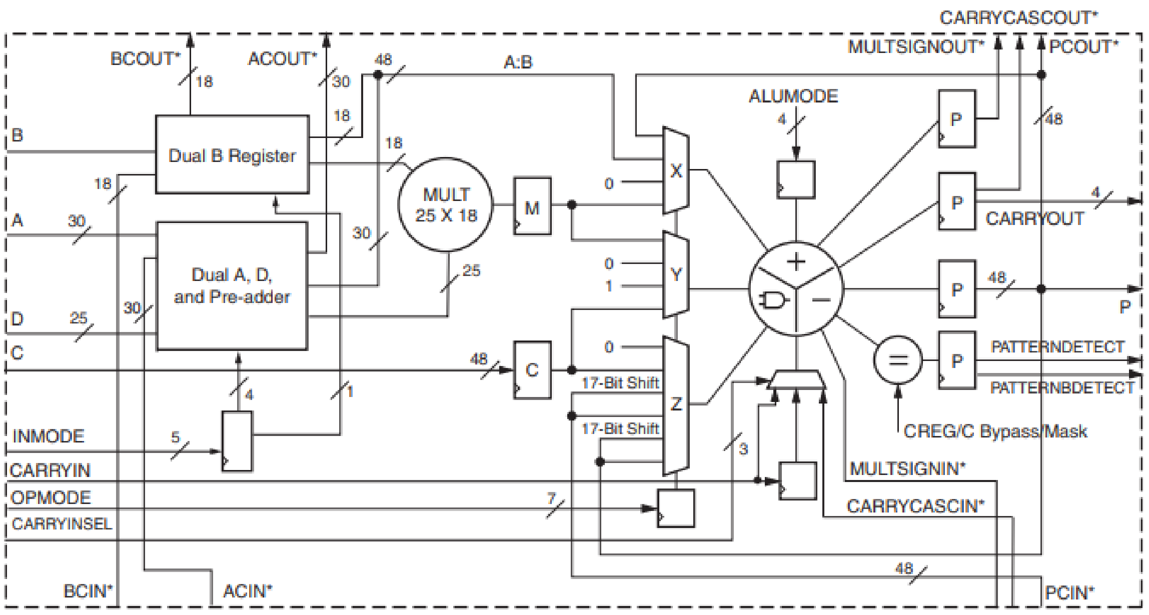


FIGURE 3.13: 7 Series DSP48E1 Slice overview [10].

3.14. SEU

Single event upset (SEU) is a change of state in the hardware which causes the hardware to produce erroneous results. The cause of soft errors is high energy particles that make contact with the semiconductor. The depiction of the phenomena is given in Figure 3.14. The type of SEU is depended on the part of the semiconductor that high energy particle makes contact. Single event upsets can be categorized as follows:

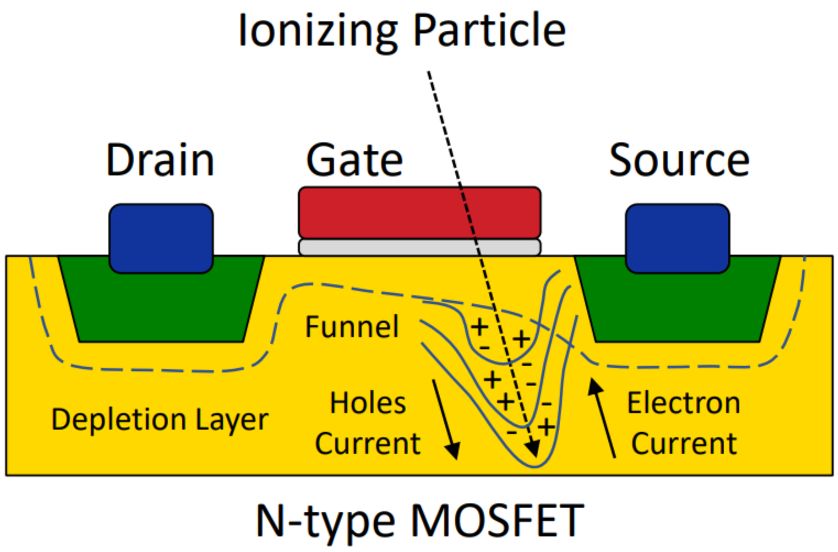


FIGURE 3.14: SEU generation of ionizing particles [11]

- Single Bit Upset: High energy particle changes the state of one bit in a memory.
- Multiple Bit Upset: High energy particle changes state of more than one bits.
- Single Event Transient: High energy particle strike a wire or interconnect. This changes the electrical level of the line for a while which is called glitch. Glitches may propagate to the remaining of the circuit depending on the logic, timing etc. If glitch causes an erroneous data to be latched, circuit produces wrong results. This type of SEU is more common in ASIC designs.

Radiation is the energy that is emitted from a source. Radiation can be classified as non-ionizing and ionizing radiation. Non-ionizing radiation is radio signals, wireless modems, cell phones, etc. This type of radiation is not able to ionize atoms or molecules because of the lack of energy they carry. Therefore ionizing radiation is the type of radiation which is important for the study. Types of ionizing radiation are:

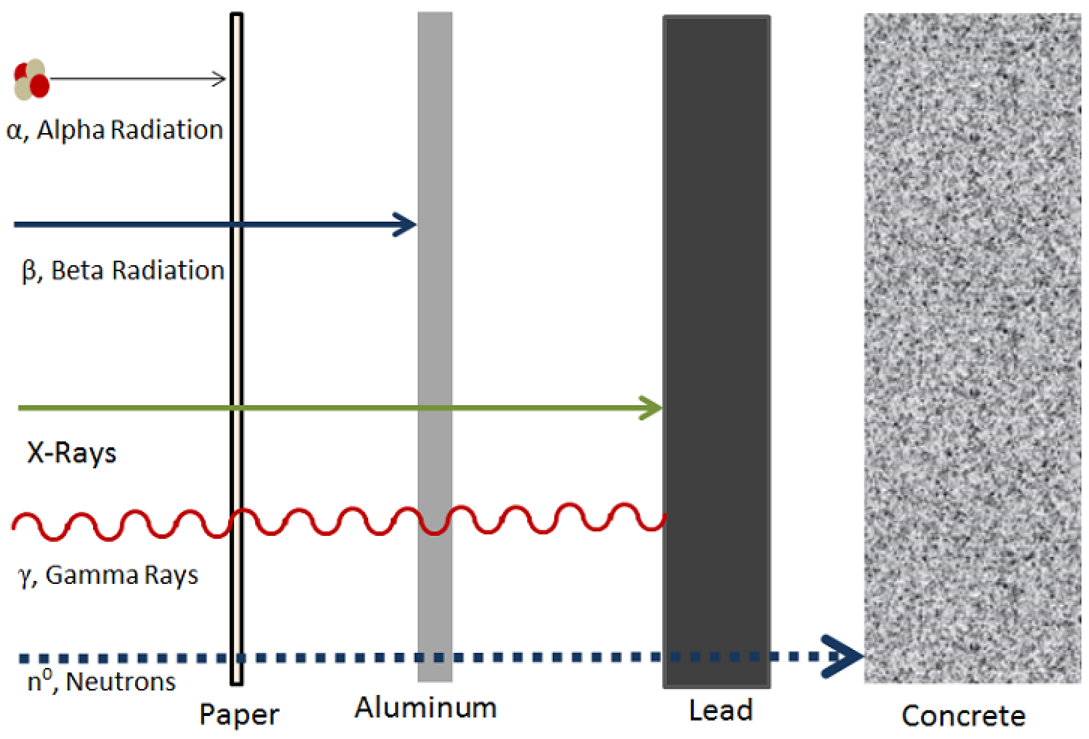


FIGURE 3.15: Penetration power of radiation types. [12]

- Alpha radiation: Consists of two protons and two neutrons. It has a positive charge and identical to a helium nucleus. They are very large particles and can be stopped with a sheet of paper. The source of alpha particles is decaying radioactive elements such as special isotopes of uranium. Decay process and radiation generation is given in the Figure 3.16.

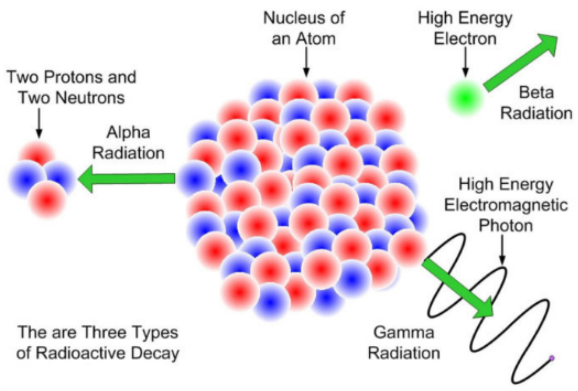

Nucleus of an Atom
High Energy Electron
Beta Radiation
Two Protons and Two Neutrons
Alpha Radiation
High Energy Electromagnetic Photon
The are Three Types of Radioactive Decay
Gamma Radiation

FIGURE 3.16: Generation of ionizing particles. [13]

Alpha radiation is terrestrial radiation. It causes problems in the semiconductor industry because materials are not pure. Impurities in the substances used in packaging emit alpha particles and cause SEU. As a precaution, semiconductor manufacturers seek for materials that do not contain radioactive substances, however, this is almost impossible and an expensive process since radioactive elements are spread through the soil of the earth. Another mitigation approach is to strengthen the resistance of the chip to radiation by a layer that prevents radiation from passing through. However, this approach makes packaging expensive and complex to produce. Therefore in semiconductors, soft errors caused by alpha particles remains an ongoing problem.

- Beta radiation: Consists of an electron, therefore beta particles are negatively charged. Beta particles are fast and have more penetration ability than alpha particles. However, they can be stopped with a thin layer of plastic. Beta particles do not pose a significant threat to SEUs because of the lack of energy.

- Gamma radiation: Usually emitted from the nucleus of an atom together with an alpha or beta particle. Gamma radiation is very penetrating. A thick layer of concrete or lead is able to stop gamma particles. They do not pose a threat to soft error generation since their energy is not sufficient to change the charge of a transistor.

- Neutron radiation: Together with the alpha radiation neutron radiation is one of the most threatening sources of SEU in terrestrial environments. The main cause of neutron radiation is cosmic rays. When cosmic rays strike the outer atmosphere they collide with atoms and create secondary particles. These secondary particles which include neutrons continue to spread in the atmosphere. Since neutrons do not have charge they continue their course without interacting with other atoms until they hit the nucleus of another atom. When they hit the nucleus of another atom they generate secondary particles again. Neutrons are dangerous because they can penetrate thick materials and have enough energy to create single event upsets in the semiconductors. Even though the generation of neutron radiation is indirect, it is a big thread to semiconductors in fail-safe applications.

The penetration levels of the radiation types are given in Figure 3.15

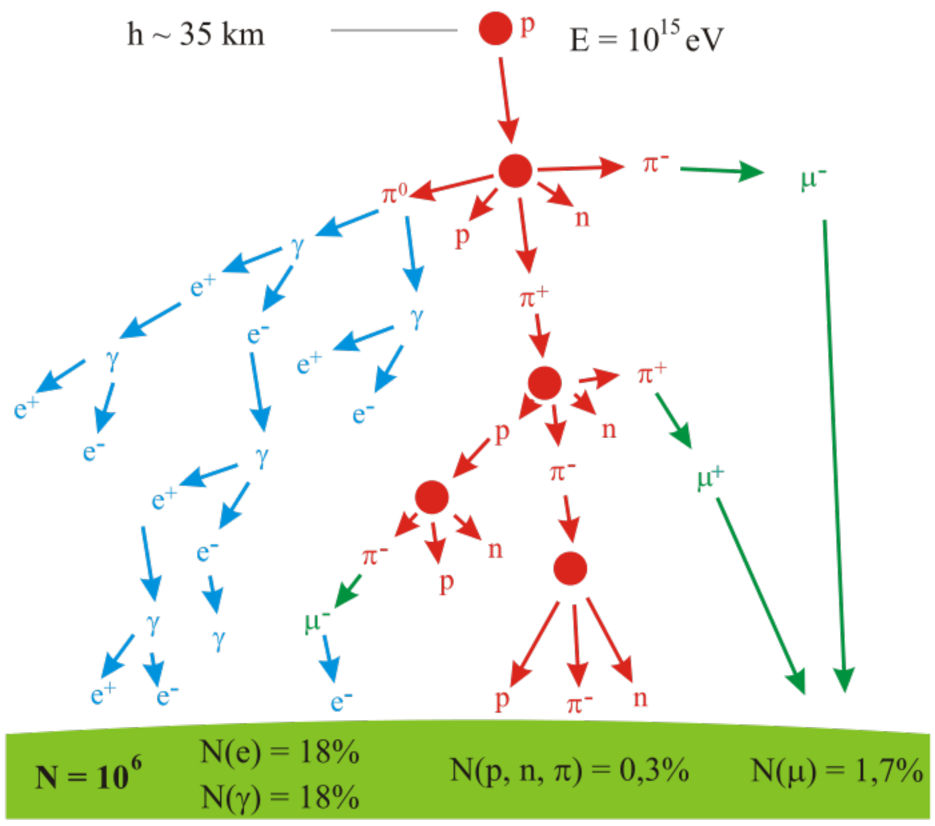


FIGURE 3.17: Cosmic ray shower.[14]

Radiation levels in the terrestrial environments and extra-terrestrial environments vary heavily. The source of extra-terrestrial radiation is galactic cosmic rays and solar storms. Only the particles that carry enough energy to penetrate the magnetosphere can reach the atmosphere. The ones that do not have enough energy to penetrate the magnetosphere create a sphere of particles which is called belts. The density of the trapped particles in belts vary in time depending on the intensity of solar winds and galactic cosmic events. Solar winds and belts formed by the magnetic field of the earth are depicted in Figure 3.18. The particles that can penetrate the magnetosphere strikes the atmosphere and creates secondary particles from the interaction with nitrogen and oxygen atoms. The chain of reactions creates more particles which is called a shower. The visual depiction of the cosmic ray shower is shown in figure 3.17. The most important and most dangerous product of cosmic ray shower is neutron radiation. Neurons radiation becomes measurable at the 350 km altitude and reaches its peak flux at 20 km. From 20 km altitude to the surface of the earth the level of neutron density drops to 1/500 of the peak flux [40].

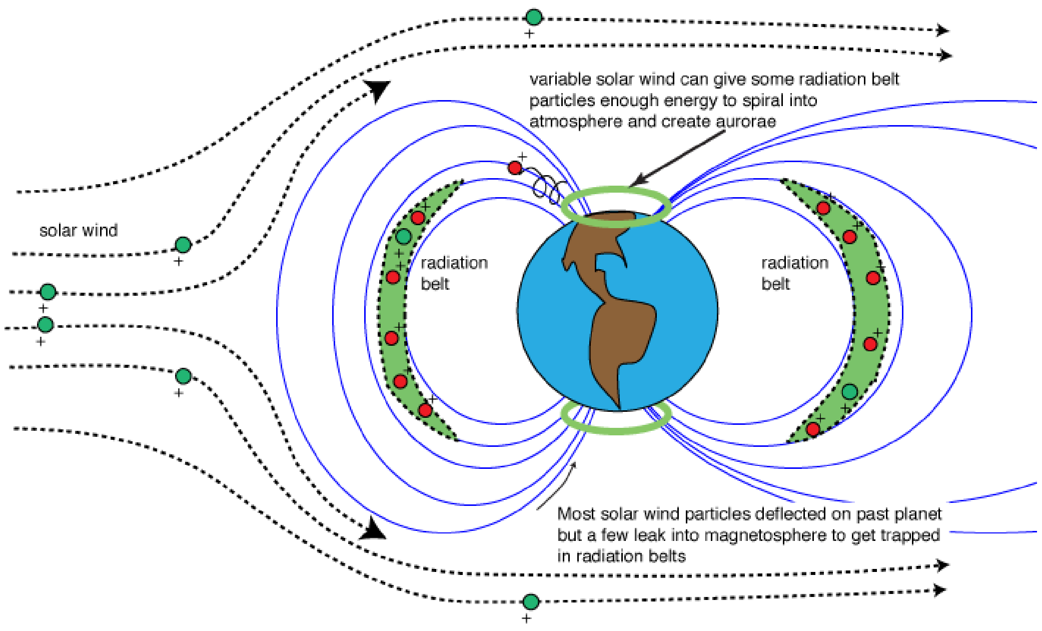

variable solar wind can give some radiation belt particles enough energy to spiral into atmosphere and create aurorae
solar wind
radiation belt
radiation belt
Most solar wind particles deflected on past planet but a few leak into magnetosphere to get trapped in radiation belts

FIGURE 3.18: Magnetosphere and belts [15].

3.15. SRAM

Static random access memory (SRAM) is a flip-flop based semiconductor data storage circuit. Contrary to dynamic access memory (DRAM), SRAMs do not require a periodical refresh. However, SRAMs should not be confused with read-only memory (ROM). ROM can keep the information while not powered while SRAM loses the information stored when it is not powered.

The main advantage of SRAM is speed. It is faster than DRAM memory however it is more expensive. Therefore SRAM is generally used for memory types where performance is the main concern such as cache memory or on-chip memory. Another advantage of SRAM over DRAM is power consumption. Since SRAM does not require refreshing, its power consumption rates are very low when it is idle. Only read-write operations at high clock speeds increase power consumption to the levels of DRAM.

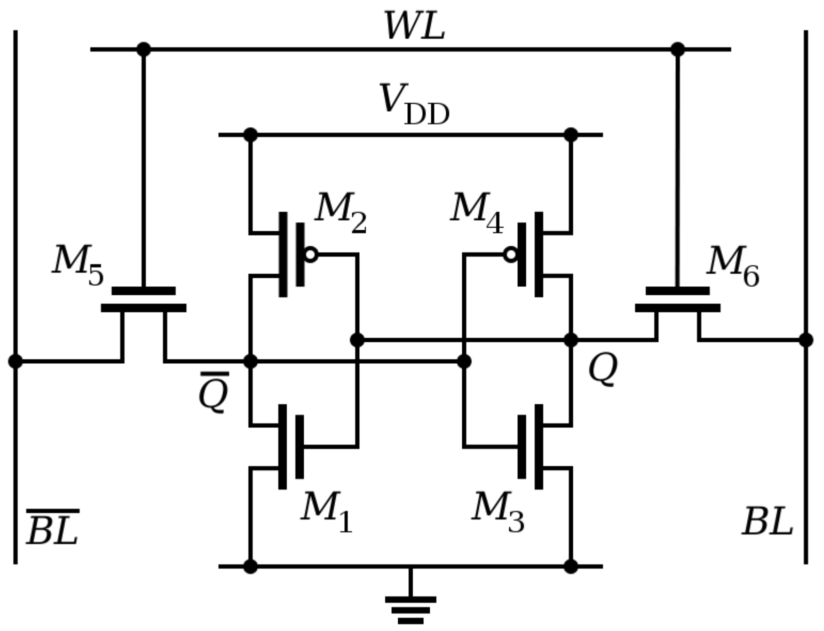


FIGURE 3.19: 6 transistor SRAM cell. [16]

A typical SRAM cell consists of six transistors. Four of them are used to store the bit and the other two are used for read-write operations. The typical SRAM cell view is given in Figure 3.19. The M5 and M6 transistors are being used for read-write operations while others are used for storing the bit.

4. TESTING METHOD

In this study, main focus is on adders and multipliers for the library characterization since they are the most commonly used resources. Additionally, several other arithmetic operations can be performed on them after small modifications. For instance, the subtraction and comparison circuits can be implemented using adders. Other arithmetic circuits and the different versions of adders and multipliers can also be added to the characterized library by following the same methodology described below.

Four most commonly used adder implementations are selected. These adders can be listed from the slowest to the fastest as follows:

- Ripple carry adder [41]
- Carry lookahead adder [41]
- Brent-Kung adder [42].
- Kogge-Stone adder [43]

The adders are implemented in the PL part using the CLBs of the Zynq-7000 board.

Two versions of multipliers are implemented: carry-lookahead multiplier (CLM) and DSP-based multiplier. While the CLM is implemented in FPGA, pre-existing DSP fabric is used for the DSP unit-based multiplier.

4.1. Overview of the Test Setup

The block diagram of the test setup is given in Fig. 4.1. In the FPGA design, five exact copies of each circuit is placed for testing since each implementation may behave differently on the FPGA because of the non-deterministic behavior of the place-and-route algorithms. All inputs and outputs are implemented as 32-bit wires, The same inputs are applied to all five circuits and their outputs are obtained. While the input and the output sizes of the adders are 32 bits, the input and outputs of the multipliers are 16 and 32 bits, respectively. Interconnection wires are driven by another module, which is called the intercommunication module. The intercommunication module is a clock-synchronous AXI slave. The task of the intercommunication module is to obtain the test data inputs from the PS via the AXI interface and drive the wires that are connected to the unit under test (UUT). The outputs of the five circuits are sent back to the intercommunication module. The PS also reads all outputs from the intercommunication module via the AXI interface.

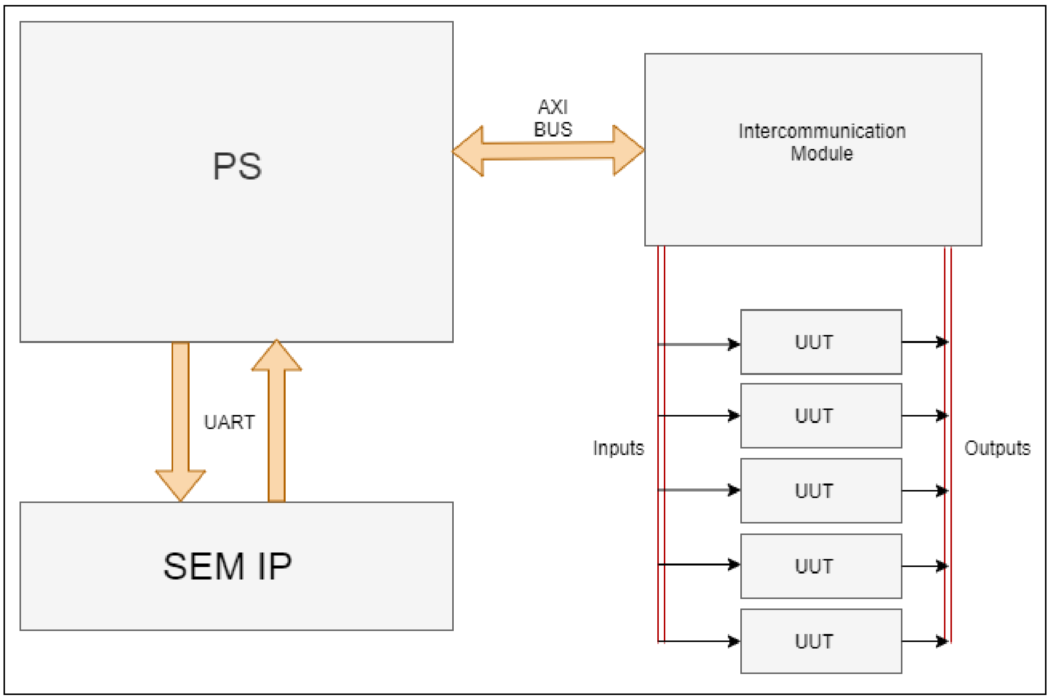
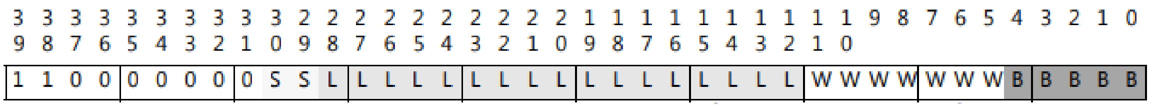


FIGURE 4.1: Block diagram of the test setup. UUT stands for unit under test.

The PS is connected to the SEM IP via the UART. The instruction parsing interface of the SEM IP is called the monitoring interface. The PS generates the instructions and sends them to the SEM IP via the UART pins. A dedicated PS-UART0 is used for the communication. There are also two addressing schemes: linear addressing and physical addressing. Linear addressing scheme does not give any information about a physical element or its location; however, the ebc file is generated according to the linear addressing scheme. For this reason, the linear addressing scheme is used for the error injection. An example of error injection instruction looks like as shown in Fig. 4.2. It consists of 40 bits as shown in the figure. It is designed to fully depict a specific bit in the whole bitstream. In the instruction bits, L, W, and B bits represent the frame number, word number, and the bit number, respectively. S bits are used to indicate if the device is a stacked silicon interconnect (SSI) device or not. A zero value indicates that the device is a non-SSI device while a value of one indicates that it is an

SSI device. Since the hardware used in this research is a non-SSI device, the S bits is set to zero in this study.



39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	S	S	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	W	W	W	W	W	W	W	B	B	B	B	B

FIGURE 4.2: Structure of linear frame instruction bits. The numbers on the top, from 0 to 39, represent the bit index numbers. [17]

4.2. Error Injection and Testing Method

The PS flowchart of the error injection and the critical bit identification is given in Fig. 4.3. PS first injects the errors to the configuration bits and feeds the test data. It then compares the results generated by the FPGA logic with the correct results. To perform these steps, it first searches the pre-loaded ebc file for determining the essential bit locations. Whenever an essential bit is determined, it infers the linear address of the bit and injects an error into that location. Error injection is basically flipping the value of the configuration bit. After flipping the bit, PS feeds the test data to the intercommunication module and collect the results. If any of the results are erroneous, PS assumes the bit as critical and collects all the statistics from the test. Afterward, PS corrects the configuration bit. In order to make sure whether the miscalculations are really caused by the injected error, PS feeds the same test data to the intercommunication module again. If all the returned results are determined as valid, then PS accepts the bit as the critical bit. If any erroneous result is detected from any of the circuits, PS marks that configuration bit as fake critical bit and increments the unexpected situation counter. Observing a fake error means that there is an unexpected error in FPGA and the bitstream should be reloaded to continue on the tests. Since not any fake errors are encountered in the tests, the behaviour of the program is not changed.

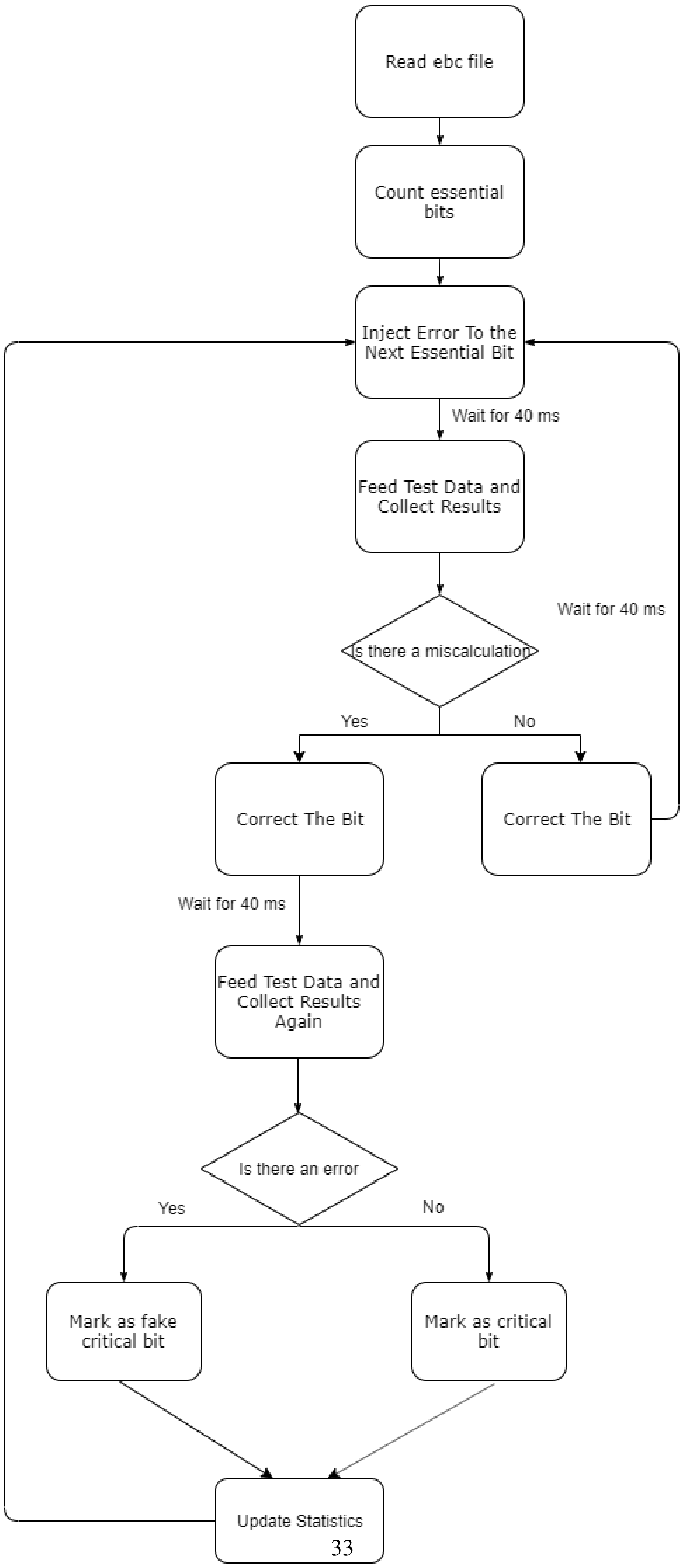


FIGURE 4.3: PS program flowchart

Xilinx's prioritized essential bits technology is used to mark the essential bits of the five circuits that are under test. Maximum clock frequency of the SEM IP is 100 MHz. At maximum clock frequency, the latency of the error detection is determined as 8.0 ms [17]. Since the clock frequency is chosen as 50 MHz for the design to satisfy the timing requirements all over the circuit, the latency is doubled to 16 ms for the error detection. The error detection mechanism scans all configuration bits and verifies the ECC codes of the frames. Therefore, PS program must wait 16 ms for the injection to propagate through the circuit after every error injection. 40 ms is selected as the waiting time as the safe latency duration. Feeding the test data and collecting the results also take another 40 ms. After correcting the error injected into the configuration bit, PS waits 40 ms again for the correction to propagate through the circuit. Therefore, testing a single bit takes 120 ms if the bit is not critical. If the bit is critical, then it takes 160 ms since the test data are fed twice. If every configuration bit is tested, testing of the whole configuration bits would take around 40 days considering the fact that there are 25,687,936 bits in the device. For this reason, Xilinx's essential bits technology plays an important role in terms of a feasible testing.

As stated before, essential bits are the bits that are important to the design. Prioritized essential bits are the bits that are essential in the marked area and critical bits are the bits that actually change the behaviour of the circuit. In the light of these definitions, the configuration bits can be classified as shown in Fig. 4.4, which is also reported in [18].

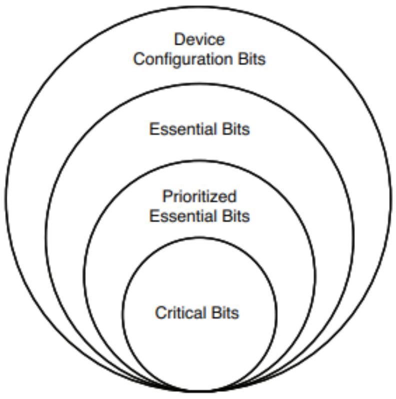


FIGURE 4.4: Hierarchy of the configuration bits. [18]

ISE Design Suite is selected for the design and synthesis platform since it supports the prioritized essential bit flow. To place the circuits in the FPGA, pblocks are used to restrain the tested circuits in a specific location. Specific constraint settings are given to implementation tool not to allow another logic to be placed or routed through the Pblocks which also guarantees that only the specified circuit can be placed in the Pblock area [44]. Fig. 4.5 shows how five implementations of the same function are constrained to the specified locations of the FPGA in the place-and-route phase. Since place-and-route algorithms are not deterministic, each implementation of the same function may result in a different configuration. Thus, each implementation may give us a different set of error propagation results. Therefore, five exact copies of the circuits are implemented to be tested and averages of the detected errors and essential bits are taken in the evaluations.

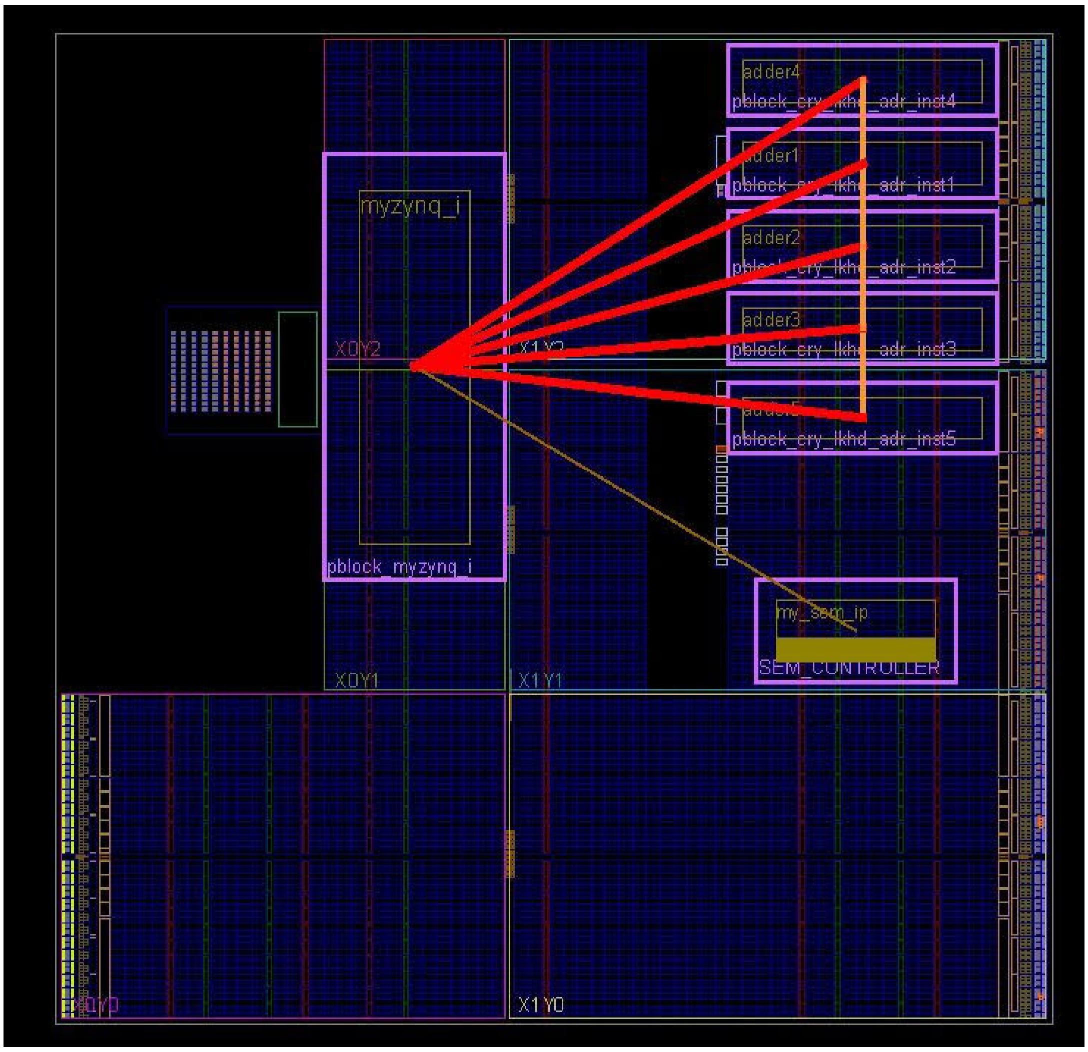

adder4
pblock_cry_lkhd_adr_inst4
adder1
pblock_cry_lkhd_adr_inst1
adder2
pblock_cry_lkhd_adr_inst2
adder3
pblock_cry_lkhd_adr_inst3
pblock_cry_lkhd_adr_inst5
myzynq_i
pblock_myzynq_i
X0Y2
X1Y2
X0Y1
X1Y1
my_sem_ip
SEM_CONTROLLER
X1Y0

FIGURE 4.5: An example design placement in FPGA.

The statistics of the tests are stored in a log file. These statistics include the number of miscalculations for each circuit for every essential bit. This log file is used for the classification and statistical analysis of each arithmetic circuit. The log data is evaluated by means of the algorithm whose flowchart is given in Fig. 4.6. If a bit creates an error for only one circuit, then it is classified as a critical bit for the related circuit. If a bit creates errors for more than one circuit but not to all of the circuits, then that bit is considered as a strangely affecting bit. Strangely affecting bits might exist in a circuit since the input nets of all five circuits are common. In other words, the input nets arrive at the region of tested circuits as one net and start leaving the main net as they reach to the location of the related circuit. The net distribution continues until the inputs arrive at the last circuit. As a result, some bits do affect all circuits. In this study, the errors affecting only one circuit are assumed to contain value for the result. Therefore, the effects of the wires is ignored in calculating the SE vulnerability. If a bit affects all five circuits, then it means that this particular bit is affecting the net tree before any of the nets are left the net tree to reach their target circuit. In the same way, if a bit affects more than one circuit that bit is also related to the net tree which does not carry valuable information about the circuit itself.

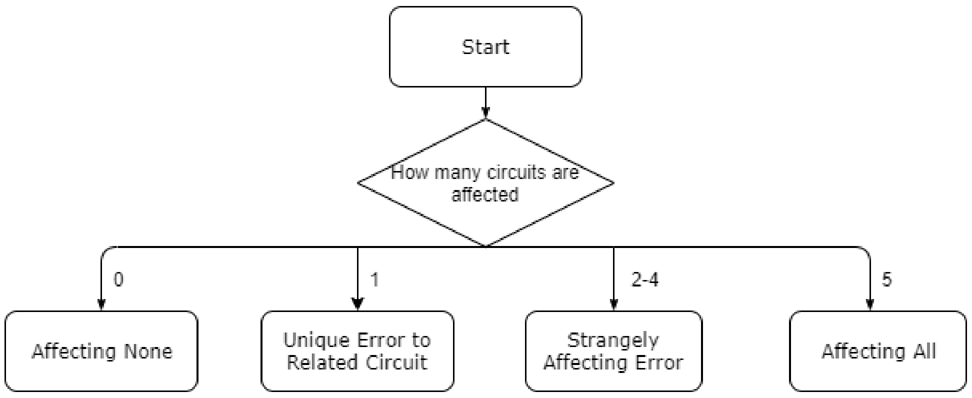

Start
How many circuits are affected
0
1
2-4
5
Affecting None
Unique Error to Related Circuit
Strangely Affecting Error
Affecting All

FIGURE 4.6: Classification of the circuits by the number of effected implementations by a bit flip.

In order to select the minimum required number of test data in terms of the accuracy and the test time, 1000, 5000, and 10000 inputs are fed to the ripple-carry adder implementation. The number of critical bits and the number of erroneous results are stored in a log file. Later, number of the erroneous results is counted for each bit. If number of erroneous results were less than five, the counter parameter is incremented, which is named as critically low error count. This value is expected to be as close to zero as possible for having accurate test results. The test results for the ripple carry adder when the size of the test data are 1000, 5000, and 10000 are given in Table 4.1. In the tests, 382 bits appear to create less than five erroneous results out of 1000 test inputs. This number decreases to two when the test data size is increased to 5000 while it becomes zero when the test data size is 10000. As a result, test input data size is selected as 10000 since it decreases the critically low error count to zero. Furthermore, increasing the number of test data from 5000 to 10000 does not change the detected number of critical bits.

TABLE 4.1: Error counts of the ripple carry adder.

Number of Test Data	**1000**	**5000**	**10000**
Number of Critical Bits	1815	1836	1836
Critically Low Error Count	382	2	0

5. RESULTS & ANALYSIS

5.1. Results of Library Characterization

As stated above, four adders and two multiplier circuits are implemented in the resource library. Power consumption, area, latency, and reliability metrics of the circuits are characterized. In a separate project, only one instance of each arithmetic circuit on the same FPGA platform is implemented to obtain the area, latency, and power consumption values. Since pipelining is not applied to the circuits, the delay of the critical path is measured as the latency value. As the area parameter, total number of slice LUTs utilized by the circuits are used. However, since DSP is a special resource in PL that does not require any additional logic, its area is taken as zero. Only the dynamic power consumption of the logic part is considered. Since a clock restriction is not given to the tool, dynamic power values are calculated under maximum switching frequency that the tool can handle. Therefore, they only represent the relative power consumption values in each circuit but not the absolute power consumption values under a specific clock frequency. The power, area, and latency values of each circuit are depicted in Table 5.1. The table indicates that when a circuit is faster than another circuit, its area and power consumption tends to increase proportionally. Since the DSP-based multiplier is not a CLB-based implementation, its parameters do not fit into the behaviour of other circuits. When four adders are considered, while the area and power parameters are directly proportional, the latency is inversely proportional to these two metrics as expected.

TABLE 5.1: Comparison of circuits in terms of power, speed, and area.

	Area*	***Latency(ns)***	***Power(W)***
Ripple Carry Adder	31	20.913	0.253
Carry Lookahead Adder	47	15.476	0.236
Brent-Kung Adder	117	9.213	0.359
Kogge-Stone Adder	185	8.982	0.737
Carry Lookahead Multiplier	383	36.425	7.445
DSP Based Multiplier	0	3.884	0.961

*Note: Unit of area is the total number of slice LUTs.

As the reliability metric, the number of critical bits and the number of erroneous results are counted for each critical bit as explained in the previous section. Five identical copies of the same circuit is averaged to determine the reliability results. 10000 input data is fed to each five of the circuits and number of incorrect calculations is collected. The incorrect calculation number per critical bit and number of critical bits can be seen in Tables 5.3, 5.2, 5.4, 5.5, 5.6, and 5.7. The number of critical bits shows the critical bits, only unique to that specific circuit and excludes the number of critical bits affecting all circuits and strangely affecting bits. In the remaining of the section the reliability values of the circuits with the average value of the five identical copies is presented.

TABLE 5.2: Average number of miscalculations of ripple carry adders

	Number of Critical Bits	***Average Miscalculation Per Critical Bit***
Circuit1	1836	456
Circuit2	1314	542
Circuit3	1227	594
Circuit4	1297	542
Circuit5	1085	646
Average	1351	556

TABLE 5.3: Average number of miscalculations of carry lookahead adders

	Number of Critical Bits	***Average Miscalculation Per Critical Bit***
Circuit1	1241	582
Circuit2	1051	628
Circuit3	1200	646
Circuit4	1495	424
Circuit5	1114	536
Average	1220	563

TABLE 5.4: Average number of miscalculations of brent kung adders

	Number of Critical Bits	***Average Miscalculation Per Critical Bit***
Circuit1	1830	736
Circuit2	1425	857
Circuit3	1668	715
Circuit4	1584	836
Circuit5	1726	739
Average	1646	776

In Table 5.8 the results of five identical circuits are given altogether to show the ratio of strangely affecting bits and bits that affects all circuits. We give the number of total essential

TABLE 5.5: Average number of miscalculations of kogge stone adders

	Number of Critical Bits	***Average Miscalculation Per Critical Bit***
Circuit1	6555	624
Circuit2	6778	623
Circuit3	6827	652
Circuit4	6998	645
Circuit5	6853	626
Average	6802	634

TABLE 5.6: Average number of miscalculations of carry lookahead multipliers

	Number of Critical Bits	***Average Miscalculation Per Critical Bit***
Circuit1	37393	457
Circuit2	29968	536
Circuit3	32854	485
Circuit4	33376	514
Circuit5	36161	246
Average	33950	447

TABLE 5.7: Average number of miscalculations of DSP based multipliers

	Number of Critical Bits	***Average Miscalculation Per Critical Bit***
Circuit1	11	10000
Circuit2	11	10000
Circuit3	11	10000
Circuit4	11	10000
Circuit5	11	10000
Average	11	10000

bits and critical bits (i.e., the bits that results in an error at the output when flipped) in the second and third columns of the table for each adder and multiplier instances. In the last three columns, the categorized bits for each circuit are listed. The bits affecting only one circuit are the bits that are causing an error at the output in one of the five identical copies. Strangely affecting bits are the bits that affect more than one circuit but not all circuits. As it can be seen from the table, the number of essential bits are directly proportional to the area of the circuits. However, critical bits do not follow this trend since each circuit may have various error masking capabilities. The proportion of the critical bits tends to decrease while the total essential bits increase as it is shown in Table 5.9. The average essential and critical bits for each implementation are listed in this table. The ratio shows how vulnerable the circuit to SEs. In other words, the bigger the ratio, the lower the reliability of the circuit. One can use this ratio as the vulnerability or 1-ratio as the reliability metric.

TABLE 5.8: Test results of the five identical circuits together in terms of vulnerability (reliability).

	Number of Bits				
Circuit Name	**Essential**	**Critical**	**Strangely Affecting**	**Affecting All Circuits**	**Affecting One Circuit**
Ripple Carry Adder	13040	6796	33	4	6759
Carry Lookahead Adder	12200	6124	23	0	6101
Brent-Kung Adder Adder	20427	8282	49	0	8233
Kogge-Stone Adder Adder	96443	34338	317	10	34011
Carry Lookahead Multiplier	439202	169848	95	1	169752
DSP Based Multiplier	735	55	0	0	55

TABLE 5.9: Error rates of the circuits on average.

	Essential Bits	***Critical Bits***	***Ratio***
Ripple Carry Adder	2608	1359	0.521
Carry Lookahead Adder	2440	1224	0.501
Brent-Kung Adder	4085	1656	0.405
Kogge-Stone Adder	19228	6867	0.357
Carry Lookahead Multiplier	87840	33969	0.386
DSP Based Multiplier	147	11	0.074

[45] reports that that FIT (Failure in Time) rate for Zynq-7000 families is 76 FIT/Mb. The test results are obtained in 10^9 device operation hours. Therefore, FIT of one bit is calculated as $76 \cdot 10^{-6}$. If one of the critical bits fails, the circuit produces incorrect results. Therefore, the FIT of a circuit can be calculated using the following simple equation:

$$FIT = (FITRatePerBit) \times (NumberofCriticalBits) \tag{1}$$

FIT rates of the arithmetic circuits are calculated and the results are reported in Table 5.10. FIT rates can also be used in designing circuits either as a constraint or as an objective function.

TABLE 5.10: FIT rates.

	FIT Rate
Ripple Carry Adder	$1.03 \cdot 10^{-1}$
Carry Lookahead Adder	$9.3 \cdot 10^{-2}$
Brent-Kung Adder	$1.25 \cdot 10^{-1}$
Kogge-Stone Adder	$5.2 \cdot 10^{-1}$
Carry Lookahead Multiplier	2.58
DSP Based Multiplier	$8.36 \cdot 10^{-4}$

5.2. Case Study

In this section, it is shown, how using the different versions of the same resource in an application can result in different reliability and power consumption values under given area and latency constraints. Six additions for the case study are used as their dependencies are depicted by the data flow graph (DFG) given in Fig 5.1. Three variations of the topology are implemented using only carry lookahead adder (CLA), only Brent-Kung adder (BKA), and mixed circuit using both CLA and BKA. For the mixed circuit, BKA is used for the nodes one, two, three, and four while CLA is used for nodes five and six. In Fig. 5.2, the schedule of the nodes are given for the circuits that only use CLA and BKA. The schedule takes four clock cycles, which is expressed as step in the figure. However, the clock cycles for each circuit implementation use the maximum delays of CLA and BKA. The schedule of the mixed circuit is given in Fig. 5.3. In this implementation, the clock rate is selected based on the delay of BKA. Therefore, CLA is pipelined into two clock cycles in order to synchronize with the BKA under four clock cycle delay. Between the every step of the schedule in Fig. 5.2 and 5.3, intermediary registers are placed to store the result of each addition. In the mixed circuit, the critical path uses only BKA not to increase the latency of the circuit. Since the CLA is used on the non-critical paths, it does not increase the latency although it takes two clock cycles to finish its execution. Note that resource sharing is not used in the implementations. Resource sharing adds extra steering logic and control logic that effect the overall area, power consumption, and the number of critical bits. Therefore, the simulation results for the implementation becomes unreliable since it is difficult to calculate the contribution of these extra logic to overall simulation results.

The performance characteristics (speed, area, and power) of three implementations are shown in Table 5.11. The second and third columns depict the clock rates and total latency of the

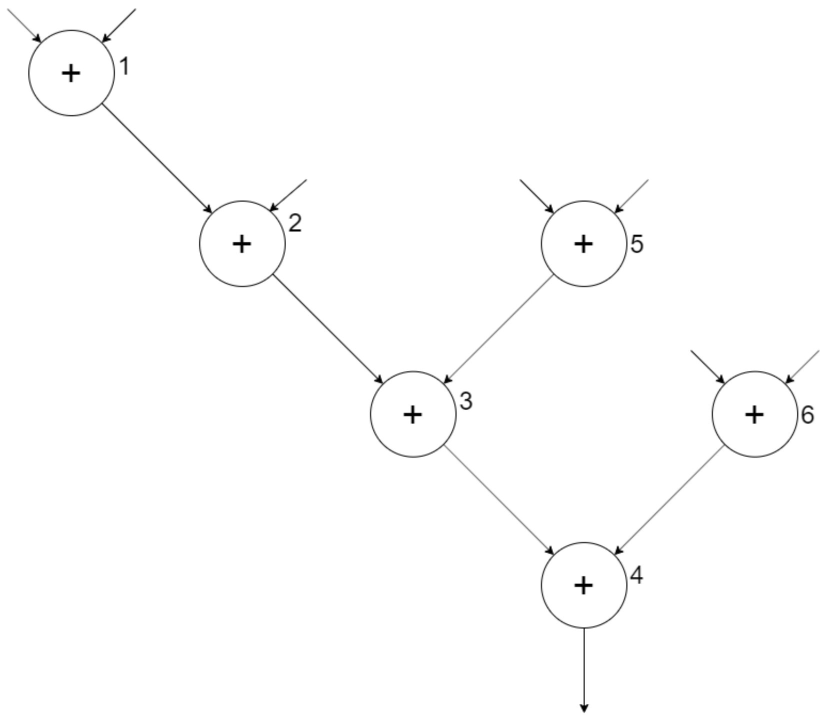


FIGURE 5.1: Customly designed data flow graph for the case study.

circuits, respectively. Column four gives the area of each implementation. Finally, the last column in the table lists the power consumption values. Since dynamic power consumption is proportional to the switching frequency, power results are collected without giving timing constraints to the synthesis tool. Therefore, power consumption values are generated under the maximum switching activity that can be handled by the synthesis tool. For this reason, power consumption values of the circuits give only the proportional relation among the circuits, not the absolute values.

When the results given in Table 5.11 are analyzed, circuits C2 and C3 are the fastest circuits since they use BKA in the critical path. On the other hand, C3 consumes less area than C2 as a result of replacing BKA with CLA for nodes five and six in the DFG. This replacement does not effect the total latency since nodes five and six are not on the critical path of the circuit. The power consumption of three implementations seems directly proportional to their areas as expected.

In Table 5.12, the error propagation results obtained from the calculations (i.e., estimation) by using the resource library and from the simulations are given. The columns two and three respectively show the essential bits and critical bits, which are determined by adding

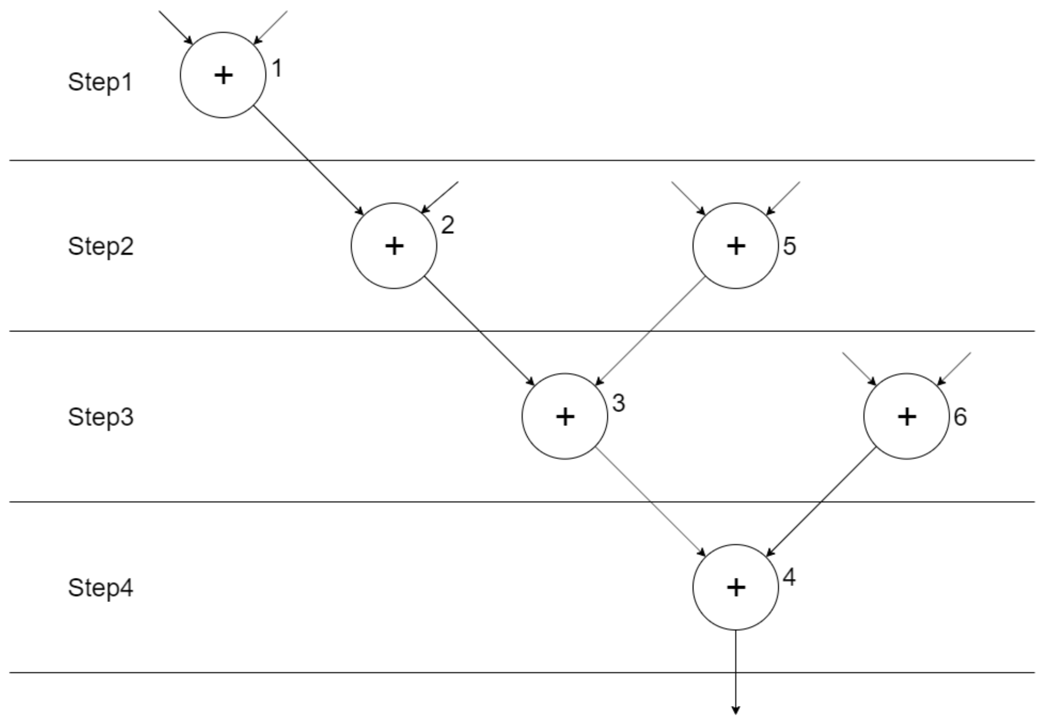


FIGURE 5.2: Schedule for only CLA and BKA implementations. Note that clock rates are different as a result of adder delays.

TABLE 5.11: Latency, area, and power consumption of three different implementations.

Circuit	***Adders***	***Clock Cycle(ns)***	***Latency (ns)***	***Area(LUT)***	***Power(W)***
C1	CLA	17	67	282	2.952
C2	BKA	11	44	671	6.224
C3	CLA-BKA	11	44	564	5.940

the essential and critical bits of each adder. The column four of Table 5.12 is the ratio of the critical bits to the essential bits, which gives the vulnerability of the estimation. Similarly, the columns five, six, and seven give the essential bits, critical bits, and their ratios obtained by the simulation. The last column shows the difference (i.e., $Diff$) in percentage between the vulnerabilities of the estimation (R_c) and simulation (R_s). This value is determined using the percentage reduction formula given in Equation (2). This column indicates that the resource library parameters are very accurate in estimating the total vulnerability (or reliability) with a very small deviation from the results obtained by simulations. Therefore, characterized resource library is very useful source for designing better circuits in FPGAs in terms of area, latency, power, and reliability. While some of these parameters can be used as design

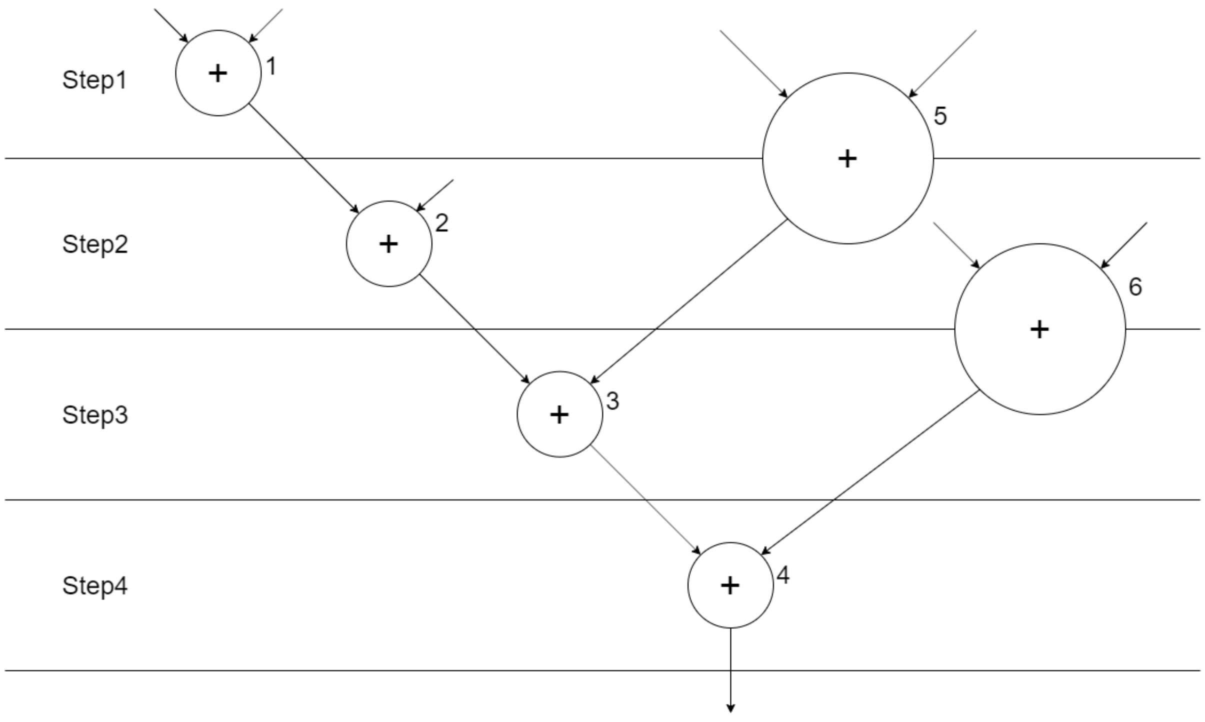


FIGURE 5.3: Schedule for the mixed adder implementation

TABLE 5.12: Error propagation values (critical bits in all essential bits) from the calculation (i.e., estimation) using resource library and from the simulation. Last column gives the error of estimation.

Circuit	***Bits from calculation***		Ratio	***Bits from simulation***		Ratio	$Diff$***(%) in***
	Essential (**1**)	*Critical* (**2**)	R_c (**2/1**)	*Essential* ((**3**)	*Critical* (**4**)	R_s (**4/3**)	R_c and R_s
C1	14640	7344	0.501	13417	6518	0.485	3.19
C2	24510	9936	0.405	23050	8737	0.379	6.40
C3	17930	8208	0.457	18924	7969	0.421	7.87

constraints, others can be the objective function parameters.

$$Diff = \frac{R_c - R_s}{R_c} \times 100 \tag{2}$$

6. CONCLUSION

Different arithmetic circuits may have different area, latency, and power consumption values on FPGAs. They can even have different response to the transient errors on the circuits. Having a characterized resource library for high level synthesis eases the design process and helps meeting the design constraints while optimizing the selected parameters. In this study, the needs of such a resource library are fulfilled and a resource library with four commonly used adders and two multipliers are characterized. A methodology for the error propagation simulations to test the vulnerability and reliability of the circuits is presented. Presented method can easily be applied to different arithmetic and logic circuits. Also the effectiveness of the resource library is tested on a custom-generated application by comparing the estimation and simulation results. The estimation results are in agreement with the results obtained by the simulations.

Results show that there is a trade-off between the analyzed characteristics. While a circuit gets faster its power and area requirements increase. As the area increases the number of critical bits increases which also inversely affects reliability. For these reasons, one should choose the circuits deeply analyzing their own topology, requirements, and constraints. As shown in the case study substituting some of the fast adders that are not on the critical path with slower adders may have a positive effect in terms of reliability, power consumption, and area while not decreasing the speed of the circuit.

In terms of multipliers, the DSP based multiplier is far more ahead of the CLA multiplier in terms of every measured aspect. Since the CLA multiplier is not a complex and fast multiplier one can expect faster multipliers will require more area and power than CLA multiplier while gaining on speed. However, DSP based multiplier is even faster than the fastest adder characterized in the library. Since DSP based multiplier requires little amount of configuration its number of critical bits is also very little. Therefore one should prefer DSP based multiplier as a choice of multiplier especially in reliability oriented designs. However, the number of DSP units is limited on the FPGA and it is completely dependant on the choice of FPGA platform. Therefore a designer should predict the number of multiplier requirement for the project and prefer the FPGA platform that satisfies their needs. Resource sharing can also be used to save on the required number of DSP units.

The library which is characterized in this study consists of four adders and two multipliers. Test results belong to 28nm Artix-7 architecture. However, one can also utilize the library as an insight even though the choice of the FPGA platform differs, especially if the design of CLB is similar between the FPGA platforms. In this study, the effects of the CLB design on reliability are not discussed and can be the work of the future.

REFERENCES

[1] Zynq-7000 soc. https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. Accessed: 2020-02-05.

[2] Umer Farooq, Zied Marrakchi, and Habib Mehrez. *FPGA Architectures: An Overview*, pages 7–48. Springer New York, New York, NY, **2012**. ISBN 978-1-4614-3594-5. doi:10.1007/978-1-4614-3594-5_2.

[3] Xilinx Inc. 7 series fpgas configurable logic block user guide. **2016**.

[4] Fpga design flow overview. https://www.fpgacentral.com/docs/fpga-tutorial/fpga-design-flow-overview. Accessed: 2020-02-05.

[5] Ripple carry adder module in vhdl and verilog. https://www.nandland.com/vhdl/modules/module-ripple-carry-adder.html. Accessed: 2020-02-05.

[6] Carry lookahead adder in vhdl and verilog. https://www.nandland.com/vhdl/modules/carry-lookahead-adder-vhdl.html. Accessed: 2020-02-05.

[7] Adders. https://web.stanford.edu/class/archive/ee/ee371/ee371.1066/lectures/lect_04.2up.pdf. Accessed: 2020-02-05.

[8] U. Penchalaiah and S. K. VG. Design of high-speed and energy-efficient parallel prefix kogge stone adder. In *2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA)*, pages 1–7. **2018**. ISSN null. doi:10.1109/ICSCAN.2018.8541143.

[9] Rami Akeela and Behnam Dezfouli. Software-defined radios: Architecture, state-of-the-art, and challenges. *Computer Communications*, 128, **2018**. doi:10.1016/j.comcom.2018.07.012.

[10] Xilinx Inc. 7 series dsp48e1 slice user guide (ug479). **2018**.

[11] Andrew Mark Keller. Using on-chip error detection to estimate fpga design sensitivity to configuration upsets. **2017**.

[12] Gamma decay. https://energyeducation.ca/encyclopedia/Gamma_decay. Accessed: 2020-02-05.

[13] Uses of beta radiation. http://allusesof.com/energy/uses-of-beta-radiation. Accessed: 2020-02-05.

[14] Neutrino map. http://neutrino.xyz/neutrino-map, **2020**. Accessed: 2020-02-05.

[15] Magnetic fields. https://www.astronomynotes.com/solarsys/s7.htm. Accessed: 2020-02-05.

[16] Nanjundappan Devarajan and V. Rukkumani. Design and analysis of static random access memory by schmitt trigger topology for low voltage applications. *Journal of Engineering Science and Technology*, 11, **2016**.

[17] Soft Error Mitigation Controller. v4. 1 logicore ip product guide. *Xilinx inc., San Jose, CA*, **2015**.

[18] Robert Le. Soft error mitigation using prioritized essential bits. *Xilinx XAPP538 (v1. 0)*, **2012**.

[19] Series FPGAs Configuration User Guide. Ug470 (v. 1.3). *Feb*, 14:1–136, **2012**.

[20] David Ratter. Fpgas on mars. *Xcell J*, 50:8–11, **2004**.

[21] M. Alderighi, F. Casini, S. D'Angelo, S. Pastore, G. R. Sechi, and R. Weigand. Evaluation of single event upset mitigation schemes for sram based fpgas using the flipper fault injection platform. In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, pages 105–113. **2007**. ISSN 2377-7966. doi:10.1109/DFT.2007.45.

[22] Paul S Graham, Nathaniel Rollins, Michael J Wirthlin, and Michael P Caffrey. Evaluating tmr techniques in the presence of single event upsets. *Faculty Publications*, **2003**.

[23] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin. Improving fpga design robustness with partial tmr. In *2006 IEEE International Reliability Physics Symposium Proceedings*, pages 226–232. **2006**. ISSN 1938-1891. doi:10.1109/RELPHY.2006.251221.

[24] A. Sari and M. Psarakis. Scrubbing-based seu mitigation approach for systems-on-programmable-chips. In *2011 International Conference on Field-Programmable Technology*, pages 1–8. **2011**. ISSN null. doi:10.1109/FPT.2011.6132703.

[25] P. S. Ostler, M. P. Caffrey, D. S. Gibelyou, P. S. Graham, K. S. Morgan, B. H. Pratt, H. M. Quinn, and M. J. Wirthlin. Sram fpga reliability analysis for harsh radiation environments. *IEEE Transactions on Nuclear Science*, 56(6):3519–3526, **2009**. ISSN 1558-1578. doi:10.1109/TNS.2009.2033381.

[26] M. Radu. Reliability and fault tolerance analysis of fpga platforms. In *IEEE Long Island Systems, Applications and Technology (LISAT) Conference 2014*, pages 1–4. **2014**. ISSN null. doi:10.1109/LISAT.2014.6845211.

[27] BASHIER MACHMUR. Reliability calculation of hdl-designs for fpga-based safety related systems.

[28] Chiraz Khedhiri, Mouna Karmani, Belgacem Hamdi, and Ka Lok Man. Concurrent error detection adder based on two paths output computation. In *2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications Workshops*, pages 27–32. IEEE, **2011**.

[29] P. Kumar and R. K. Sharma. Double fault tolerant full adder design using fault localization. In *2017 3rd International Conference on Computational Intelligence Communication Technology (CICT)*, pages 1–6. **2017**. ISSN null. doi:10.1109/CIACT.2017.7977345.

[30] Swaroop Ghosh, Patrick Ndai, and Kaushik Roy. A novel low overhead fault tolerant kogge-stone adder using adaptive clocking. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '08, page 366–371. Association for Computing Machinery, New York, NY, USA, **2008**. ISBN 9783981080131. doi:10.1145/1403375.1403462.

[31] K. Vitoroulis and A. J. Al-Khalili. Performance of parallel prefix adders implemented with fpga technology. In *2007 IEEE Northeast Workshop on Circuits and Systems*, pages 498–501. **2007**. ISSN null. doi:10.1109/NEWCAS.2007.4487969.

[32] S. Daphni and K. S. V. Grace. A review analysis of parallel prefix adders for better performnce in vlsi applications. In *2017 IEEE International Conference on Circuits and Systems (ICCS)*, pages 103–106. **2017**. ISSN null. doi:10.1109/ICCS1.2017.8325971.

[33] A. N. Jayanthi and C. S. Ravichandran. Comparison of performance of high speed vlsi adders. In *2013 International Conference on Current Trends in Engineering and Technology (ICCTET)*, pages 99–104. **2013**. ISSN null. doi:10.1109/ICCTET.2013.6675920.

[34] Dr P Samundiswary Maroju SaiKumar. Design and performance analysis of various adders using verilog. *International journal of computer science and mobile computing*, 2(9):128–138, **2013**.

[35] D Mohanapriya, Dr N Saravanakumar, and Erode BIT. A comparative analysis of different 32-bit adder topologies with multiplexer based full adder. *International Journal of Engineering Science*, 4850, **2016**.

[36] Mostafa Salehi, Ali Azarpeyvand, and Armin Hajaboutalebi Aboutalebi. Vulnerability analysis of adder architectures considering design and synthesis constraints. *Journal of Electronic Testing*, 34(1):7–14, **2018**.

[37] Louise H Crockett, Ross A Elliot, Martin A Enderwitz, and Robert W Stewart. *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. Strathclyde Academic Media, **2014**.

[38] Xilinx Inc. Zynq-7000 soc technical reference manual. **2018**.

[39] Mohamed Elhady Keshk and Kenichi Asami. Fault injection in dynamic partial reconfiguration design based on essential bits. *Journal of Aeronautics and Space Technologies*, 11(2):25–34, **2018**.

[40] J. L. Barth, C. S. Dyer, and E. G. Stassinopoulos. Space, atmospheric, and terrestrial radiation environments. *IEEE Transactions on Nuclear Science*, 50(3):466–482, **2003**. ISSN 1558-1578. doi:10.1109/TNS.2003.813131.

[41] M Morris Mano and MD Ciletti. Digital design. *New Jersey, Pearson Education Inc*, pages 158–164, **2015**.

[42] Brent and Kung. A regular layout for parallel adders. *IEEE Transactions on Computers*, C-31(3):260–264, **1982**. ISSN 2326-3814. doi:10.1109/TC.1982.1675982.

[43] Peter M Kogge and Harold S Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE transactions on computers*, 100(8):786–793, **1973**.

[44] Xilinx Constraints Guide. Ug625 (v. 14.5) ed. *Xilinx, April*, **2013**.

[45] Xilinx UG116. Device reliability report. *First Half*, **2015**.