Design Space Exploration of N Modular FPGA Structures

Joseph Abdilla
Bsc. Thesis
July 2019
abdillaj42@gmail.com

Abstract—The purpose of this paper was to provide a design space exploration of previously defined FPGA structures as defined by Robert Glein et al. Using their methods to analyse the reliability of such structures this exploration is undertaken using both a Genetic algorithm and exhaustive search. Effort was made to vary specification's of voters using in these structures, and show their effect on the structure's reliability. It was seen that artificially limiting the k metric, of the k of n voter, showed improvement of a given structure's reliability even when considering the possibility of correlation of errors overruling correct results.

1 Introduction

Redundancy is defined by Laprie as: "Redundancy is the provision of functional capabilities that would be unnecessary in a fault-free environment.".[JCL93] Modern systems, can be and are, expensive, safety critical, complex, exposed to hazardous environments. Any one of these factors require fault tolerance something which harsh operating conditions are only likely to reduce. One such area of interest is space, where devices are subject to large temperature fluctuations, chemical exposure, and radiation, among other things. An example being the Voyager space probes, utilizing redundant parts in order to maintain functionality as the craft aged and wear accumulated.[Sim00]

This paper will discuss and analyse the mitigation of transient effects in both a commercial and space-grade FPGA.

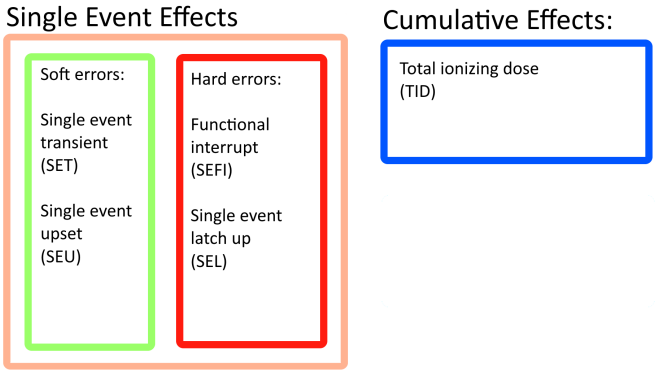

Figure 1. Diagram of the different radiation effects FPGA's are susceptible to.[Mes14]

2 Types of redundancy

The problem of providing redundancy will be divided into two types, that of software and hardware.

2.1 Software redundancy

Software redundancy is a far larger and open problem when compared to Hardware redundancy. [Dub] Simply paralyzing modules does not fix the problem as the erroneous state is present in all modules. It is difficult (if not impossible) to reduce software modules into defined types, frustrating attempts to create a systematic method for redundancy planning.

This being one of the reasons that software redundancy was not chosen as the focus for this paper.

2.2 Hardware redundancy

This is what comes to mind for many when the term ”redundancy” is thrown around. The method of supplying multiple physical copies of modules in order to ensure against the failure of any one.

2.2.1 Effects of a radiation environment on Integrated Circuits

When thinking about designs for a high radiation environment there are many factors and metrics to take into account. Effects can be split into types.[Tri16] Some examples are: Overall effects: Total Ionizing Dose, Enhanced Low Dose Rate Sensitivity. Transient effects: Single Event Effects, Single Event Functional Interrupts. Permanent destructive effects: Latch Up, Stuck Bits. All of these effects affect semiconductor components in different ways and to different degrees, permanent damage or downtime being the result. Fig: 1 shows this in diagram form. While permanent effects result in damage of the device or module, the affect of transient effects on device function can be mitigated or even masked when they occur.

2.2.2 Physical component duplication

A simple example of spacial redundancy being a system comprised of N units in parallel, with only one functional at any one time, a different one taking over the workload as soon as a failure occurs. This assuming perfect unit switching behaviour and perfect unit failure detection. The equation for which being rather trivial [Sch15]:

$$R(t) = e^{-\lambda t} \sum_{i=0}^{N-1} \frac{(\lambda t)^i}{i!} \tag{1}$$

Where λ is failures per hour, t is the system operating time, N is the number of parallel units, and $R(t)$ is the system reliability from 0 to 1. Where 1 is completely reliable and 0 is never reliable.

Fig. 2 shows that the decrease in reliability gain for each additional unit added is asymptotic, as is expected from the equation. While this model is certainly promising for the applications of parallelism it comes with some caveats: the units cannot fail while not in use; the is no loss of functionality during the switching (if there is any sequential processing occurring, it is preserved). Therefore this approach is more suited to devices such as power supplies, or devices running non critical processing/error tolerant processing, to components that are already reliable as their multiplication proportionally increases the mass and size of the finished system. Here the first hints of the law of diminishing returns is demonstrated.

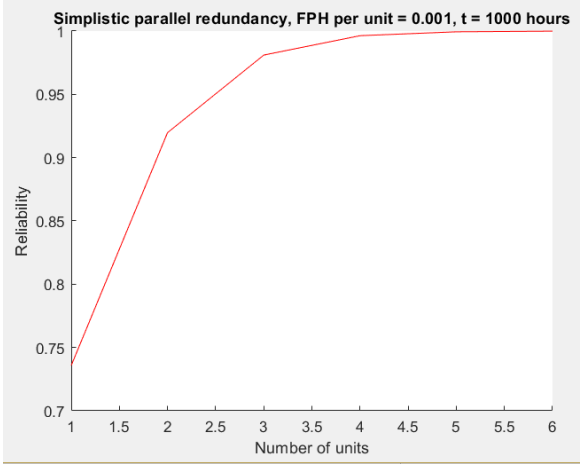

Figure 2. Simple parallel implementation of Hardware redundancy

3 The FPGA as Hardware redundancy

The methods discussed before each require physical duplication at the component level, this adds mass and cost proportional to the degree of redundancy chosen. It would therefore be beneficial to add redundancy in a different form factor.

FPGA’s are increasingly used in signal processing applications due to their flexibility and high throughput due to their potential for concurrent processing. They also offer an opportunity for on chip modular redundancy, where computation streams are duplicated to help guard against erroneous results. All things that make for a good fault tolerant design.

3.1 Not all FPGAs are created equal

Not merely a homogeneous mass of transistors on a substrate the modern FPGA contains

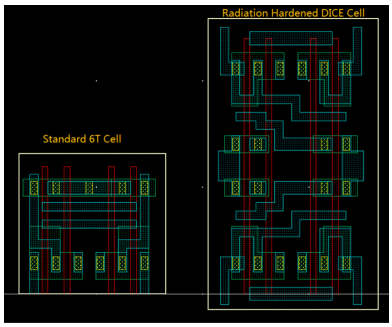

Figure 3. Layout example of a commercial 6T RAM cell shown with a 12T Rad-Hard cell. [Cai+19]

many specialist executions blocks and sections, each affected by radiation to different degrees.

In order to do this properly the differentiation between the designs of Commercial Off The Shelf (COTS) parts and Radiation Hardened (Rad-Hard) ones must be made clear. In order to facilitate this comparison two devices were compared. Xilinx Rad-Hard Virtex-5QV, and Xilinx Kintex-7.

Fig: 4 clearly shows the differences between the weakness to radiation displayed by the commercial part. These differences are embedded in the technology used for each of these primitives, and as such the function of the design informs the primitives it uses and therefore it's reliability.

COTS devices are consumer/industrial grade in nature and are not designed with radiation tolerance in mind, whereas Rad-Hard are. They employ methods including (but not limited to) 12 transistor static RAM, and inbuilt error correcting coding. The drawbacks for a Rad-Hard FPGA include a much higher price, higher power usage, lower working frequency, smaller possible design size, and being in a larger transistor node among others. [Cor15]

3.2 Module and system reliability

Leading on from this one can start to get an idea of just how unreliable a module is based on the primitives it uses. In order to do this first knowledge of the amount of each primitive used is needed. This is most easily done by passing a single FPGA design module through the relevant tool-chain, this will output the primitives used. This is the starting point. Appendix A lists the resources available to the two FPGAs under consideration.

$$U_{pri,mod} = U_{pri} \cdot \frac{n_{,mod}}{n_{device,total}} \tag{2}$$

$U_{pri,mod}$ is the module upset rate for a specified primitive. To get the complete modules upset rate each primitive must be summed up.

$$U_{tot,mod} = \sum_{pri}^{all,pri} U_{pri,mod} \tag{3}$$

This has now given the module upset rate according to the amount and type of primitives used. This, however, is only a rate of failure. Reliability will be calculated by assuming that it decreases exponentially with time: [Rel]

$$R_{mod,t} = e^{-U_{tot,mod} \cdot t} \tag{4}$$

$R_{mod,t}$ is the probability that the module does not suffer any failures during the interval $[0, t]$. With $t = 1s$, and setting $T = 3600$ it becomes simple to calculate the probability of failure per hour (PFH) [Gle+15].

$$PFH_{mod} = 1 - R_{mod,T} \tag{5}$$

Now an equation for the reliability of a module has been found based on it's used primitives and their upset rate.

3.3 Voting

It is clear that a singular processing unit would not provides a fault tolerant or reliable system. Duplication on it's own also may not provide this. Therefore, a voting system is to be introduced. The voter will be define by two metrics, firstly n, this is the number of inputs it is taking into consideration. Secondly k, this is the minimum number of values which must be the same for the value to be considered correct and passed to the next stage of the signal chain.

Assuming a structure similar to Fig: 5a, a value can be calculated for the reliability of the system using $k/2 + 1$ voter. Using perfect voter reliability (for now), Module reliability of $R(m)$, and the total system reliability R_{sys}.

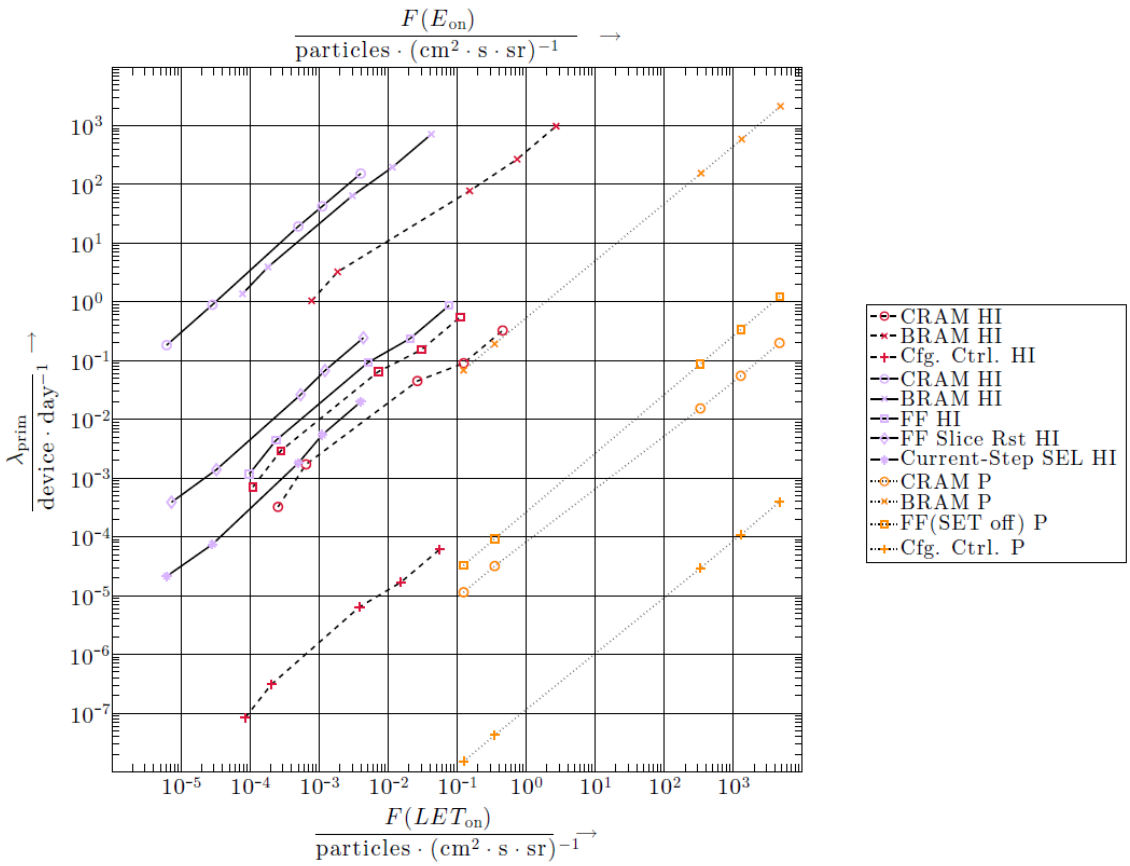

Figure 4. Comparing the Heavy ion (HI) upset rate as function of the integral flux F(LETon) for Kintex-7 (XC7K325T, purple marker and solid line) and Virtex-5QV (red marker) as well as proton (P) upset rate as function of the integral flux F(Eon) Virtex-5QV only (orange marker) for selected FPGA primitives in GEO and 7mm Al shielding. The markers of all curves from left to right represents the upset rate at the onset of the Solar Maximum, Solar Minimum, Worst Week, Worst Day and Peak 5 Minutes condition. [Gle+15]

$$R_{sys} = \sum_{i=k}^{n} \binom{n}{i} R(m)^i (1 - R(m))^{n-i} \qquad (6)$$

In order for a voter system to be effective, the number of values being compared must be greater than two (as with two values there is no possibility to form a consensus as to which is correct when an erroneous value occurs).

The design of the voter is also important to the success of the design. Consider a 1 bit voter of input vector size n, which requires at least k correct inputs for the output to be deemed "correct" and passed to the output, where $k \geq 3$. The value of k in this case is:

$$k = \left\lfloor \frac{n}{2} + 1 \right\rfloor \qquad (7)$$

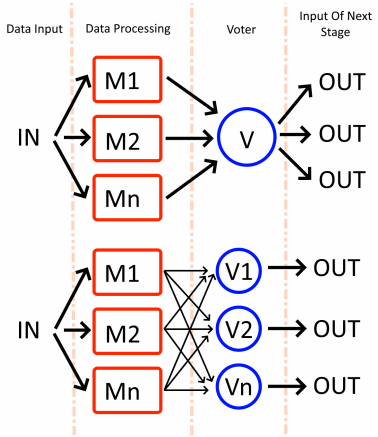

Figure 5. Simple graphic showing possible voter used for this analysis, where the value of 'n' can be as high as required. [Mes14]

Evaluating to:

$$(n,k) = (3,2),(4,3),(5,3),(6,4),(7,4),(8,5),... \tag{8}$$

This voter types seems robust in that it always requires a complete majority of answers for the result to be considered correct. This brings up several questions on how to get more out of the voter. A complete majority system automatically assumes that there is only two states for an answer 1 or 0, this provides clear delineation between the states but simplifies the types of incorrect answers that could be produced. As FPGAs are often used with wide data widths it is possible to introduce more states. In this method the collisions between erroneous values is sought, to determine if the frequency of their occurrence outweighs the frequency of the stated "correct" output.

The probability that i incorrect values consist of the same bit pattern is described by [Gle+15]:

$$\frac{2b}{b^i} \tag{9}$$

Combining this with Eq. 6 the reliability of a an N modular system that includes these error duplication's can be expressed [Gle+15].

$$R_{sys} = \sum_{i=k}^{n} \binom{n}{i} R(m)^i (1-R(m))^{n-i} - \sum_{i=k}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{i} R(m)^i (1-R(m))^{n-i} \cdot \frac{2^b - 1}{2^{b \cdot i}}$$

As seen in Fig: 6, the probability of any specific number of erroneous values sharing the same bit string decreases exponentially. This of course requires that there be multiple errors distributed among different computing modules within the same computation time (time in which the result is calculated and output, before the next calculation begins).

It should also be noted here that every SEU does not ultimately become a SEFI, without such the error is not visible to the outside world and has not caused a computation error [Ste08]. This effect reduces the probability of an error being asserted as "correct" by a voter. However this will not be discussed here for reasons of simplicity.

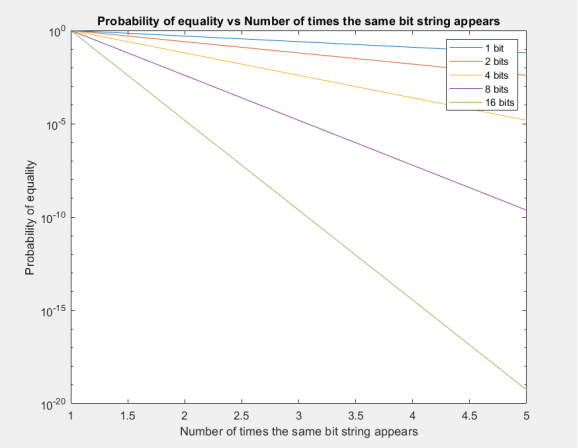

Figure 6. The probability of error collision. [Mes14]

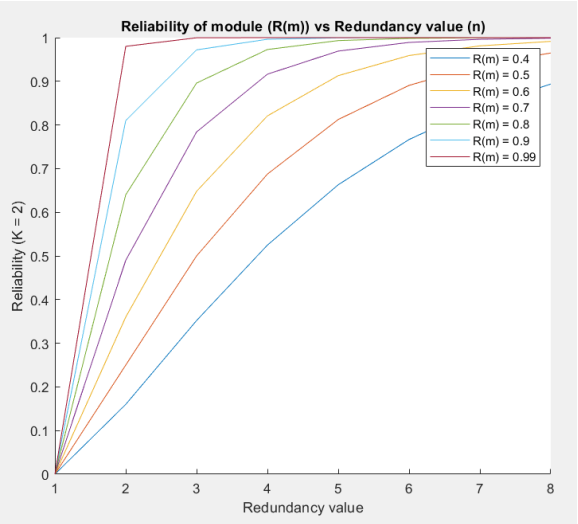

Figure 7. Reliability of system made of n modules with K = 2. [Mes14]

To determine which of the two voting strategies it is helpful to analyse the system reliability when using the two voter types. For this the standard floor type will be used along with one with majority voting (with this value set to $K = 2$. This low K value is justified by relatively low redundancy factors and the low probability of an error collision. These two strategies, when compared, show large differences as the order of redundancy increases. Seen in Fig: 7 and Fig: 8

The lower the k value can be kept the lower the redundancy factor need be for a certain

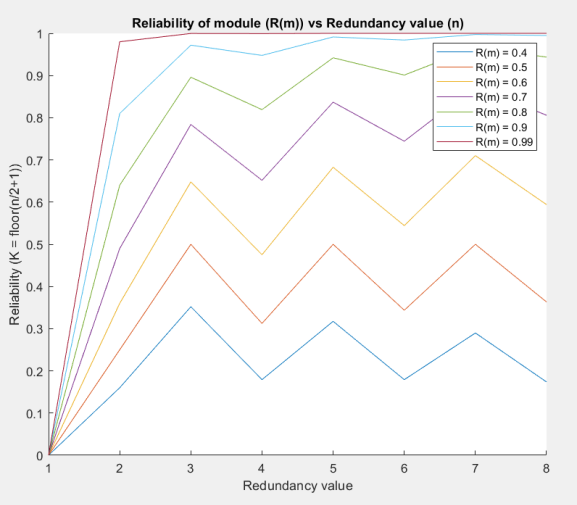

Figure 8. Reliability of system made of n modules with K = floor((n/2)+1). [Mes14]

level of reliability. This of course means fewer FPGA resources used, meaning more resources for other things and a lower upset rate overall.

Also of note here is that with traditional voting there is no reason to consider using even values for n because the resulting reliability due to k is always lower than the reliability bought by a system of $n-1$. Adding modules to a voting system does not always mean a more reliable system. However, with a fixed artificially limited value of k the reliability increases whenever another module is added.

3.4 The cases

The analysis is split into two Cases, continuing from the work of Robert Glein et al. [Gle+15] using the three logic blocks De, Di, and Ld for modules. The functions of these logical and computation blocks is only important insofar as they define which FPGA primitives the module uses. Case 1 will be following on from Fig: 5 (top), with only the level of redundancy for the modules changing. Case 2 will build on this to add changes to the voter redundancy. The whole module chain will be defined as $[A, B, C]$. Where each letter represents the value of n used for a particular module. The structure under test will also be defined as either SV (single voter) for Case 1, or NV (nth order voter) for Case 2. Therefore also defining the type of structure under test. The specific value of k in use will be defined nearer the time of computation.

Method	Speed	Search completeness
Exhaustive search	Slow	Complete
Genetic Alg.	Medium	Partial

Table 1
Comparison of analysis methods

3.5 Calculating primitives

A somewhat true mythos surrounds the realm of Hardware Description Languages and FPGA usage, that they take inordinate amounts of time to software complete a compile-debug-edit cycle. [Lav+10] With this in mind it was decided to try and keep the process of reliability calculation simple. There is a an argument (in the interest of supreme accuracy) to generate a test structure, have the tool analyse it and determine the number of resource primitives that are required, and whether it is viable size-wise. From this a calculation can be made into how reliable the structure is, the result informing the creation of the next test structure. Resource primitive values could taken from a single version of the module under consideration and then extrapolated linearly. While this method is not as accurate as running the module system through the relevant tool-chain, it does provide a large time saving (1 hour per system) [Gle+15].

Let M be a module using primitives p, it is then duplicated by a redundancy order of n.

$$nM = np \tag{10}$$

3.6 Problem analysis methods

Now that the method of Reliability calculation has been established the analysis method can be discussed. It is clear that the problem is one of design space exploration. The most obvious search method being a brute force exhaustive search, however this is assumed to be the slowest option.

4 ANALYSIS

4.1 Genetic: n modular with one voter

In order to walk before we run the initial test will be done using the top structure in Fig: 5.

First the amounts used for each block type in terms of FPGA resources is calculated, then the reliability for each module is determined. This is then used to calculate the reliability of the redundant module structure.

In order to implement a Genetic Algorithm a species must first be defined. The species being the overall container of genes, with genes being the variables that the Algorithm can edit and then compare results to establish the fitness of the individual in question.

As for fitness criteria the area being utalised will be the only metric, to ensure that the evolved design will fit within the FPGA fabric (as per SubSection 3.5). Other possible metrics included: minimal power usage, area minimisation, least amount of a particular resource. However, these are more esoteric and require more innate knowledge of the design than is understood here.

For this first version Case 1 will be analysed therefore the species has only three genes, one for each of the N values of the modules. A population is created randomly with upward of 20 members, they are then passed through the fitness function to determine their PFH, and whether the structure fits within the FPGA's resources. If it does fit it can move the the next stage, if not it is marked for removal from the population. The population is then ranked, with the lowest PFH as best specimen down to the largest PFH, with preference given to the smallest area if two are ofund with the same PFH. The top most members are bred via crossing of gene values, a random member is then mutated, this makes up the next generation. The cycle then continues as many times as specified.

This was done in Python 2.7 in cooperation with Matlab due it's ease of use and large number of libraries available. These libraries are optimised and provide many specific functions to aid and speed up these simulations. [19]

An initial population size of 20 individuals was used, with 10 generations of evolution. This was deemed enough as a steady state was reaches after approximately 6 generations, and so 10 was considered sufficient to ensure the evolution completed.

4.2 Exhaustive

The simplest method to find the lowest PFH of all the variations. To do this all module redundancy variations are calculated and their PFH determined, then the best is taken as the result, again with preference given to the smallest when equal PFH values are found.

4.3 N modular redundancy with singular voters

Running the GA and the exhaustive search on Case 1, using the De, Di, Ld blocks is dependent on the highest value of redundancy allowed, this was set at 8 to allow extensive exploration of the design space. It is seen from earlier there is very much a law of diminishing returns when it comes to redundant modules, where adding more only serves to use up ones limited resources. It was found that after the $5th$ order of redundancy there was no reduction in the reported PFH.

The GA was run several times and produced many best individuals a recurring theme among them was the limitation of high resource usage units such as Ld.

Table: 2 shows the results of the GA for single voter structures for both $k = floor((n/2) + 1)$ and $k = 2$ specifications. $k = 2$ was chosen due to the low likelihood of error collision for higher data widths. The effects of altering these widths will be discussed later.

These results show firstly that the GA was a suitable method of finding the best individual to satisfy the criteria of low PFH, they tally very closely (if not exactly) to the results found via an exhaustive search.

4.4 N modular redundancy with N voters

Results from the single voter proved the viability of the GA approach, therefore the Nth order voters were not implemented. These can be seen in Table: 3. The GA again producing very similar outputs to the Exhaustive search.

Test situation	Best ind. (GA)	Solar min. PFH	Best ind. (EXH)	Solar min. PFH
5Q SV k=flr(x)	[5,3,3]	5.11e-08	[3,3,3]	5.11e-08
5Q SV k=2	[4,3,3]	5.11e-08	[3,3,3]	5.11e-08
K7 SV k=flr(x)	[5,5,3]	1.13e-05	[3,3,3]	1.13e-05
K7 SV k=2	[4,4,4]	1.10e-05	[4,3,4]	1.10e-05

Table 2
Single voter structure analysis, data width is 16 bits

Test situation	Best ind. (GA)	Solar min. PFH	Best ind. (EXH)	Solar min. PFH
5Q NV k=flr(x)	[5,3,3]	7.33e-12	[5,3,3]	7.33e-12
5Q NV k=2	[4,3,3]	7.33e-12	[4,3,3]	7.33e-12
K7 NV k=flr(x)	[5,5,3]	2.37e-07	[5,3,3]	2.37e-07
K7 NV k=2	[5,4,4]	1.87e-11	[5,4,4]	1.87e-11

Table 3
Nth order voter structure analysis, data width is 16 bits

4.5 Analysing results

Comparing the PFH results in this paper to the ones found through both the GA and Exhaustive search it is clear that there has been no substantial reduction. While this is disappointing, upon looking back at SubSection 3.3 a reason for this is seen. Adding more modules has diminishing returns, the asymptotes of Fig: 7 are reached, and there is no further reduction in the PFH.

With this upper limit now firmly established it provides limits on the design space to be analysed by either search method. The search space open to the Exhaustive method being exponentially proportional to the Nth order tested, with there being $(N-2)^3$ structures to compute.

4.6 Effects of data width

Here the value of b from SubSection 3.3 will be altered to see it's effects on PFH values. The problem the arises when using a set value for K, under consideration here as $k = 2$, is that there is the opportunity for erroneous values to be asserted as correct. With $k = 2$ an erroneous value need only be produced at least twice due to upsets and it could be considered "correct". This problem becomes worse the lower the value of K. The width of the data and FPGA primitive usage being used within the implementation has been assumed to scale linearly. Fig: 9 shows that the larger the data width in use the lower the PFH, the lower the data width the more likely the same error string occurs multiple times during one calculation cycle.

This will be done for system $[5, 4, 4]$ for the Kintex 7 in nth order voter setup. To calculate the primitives for this calculation, the values for $[5, 4, 4]$ were taken. Seeing as these values are for $b = 16$, this value was scaled linearly for the other tested b vlaues. When comparing the the results of $k = 2$ and $k = floor((n/2)+1)$ for the structure, it is possible to see how useful this limiting of K is. For the sake of graphical clarity only 3 of the 5 radiation metrics have been draw, but they are typical of the results. Firstly it is seen that altering the data width for $k = floor((n/2) + 1)$ does alter the PFH, errors are never considered in such number as to override a correct answer, therefore the only reason for change is the increasing primitive usage. The resulting PFH increases with used primitives (as to be expected). While with $k = 2$ the PFH reduces as the width increases, this is attributed to both the reducing likelihood of error collision as width increases and the value of k providing impetus to reduce the PFH. Despite the two converging toward similar PFH values, there is never a time where the fixed k system becomes less reliable than the traditional system. This is the main interesting result of this work.

5 Conclusion

During the course of this work it has become clear just how limited adding more than 3

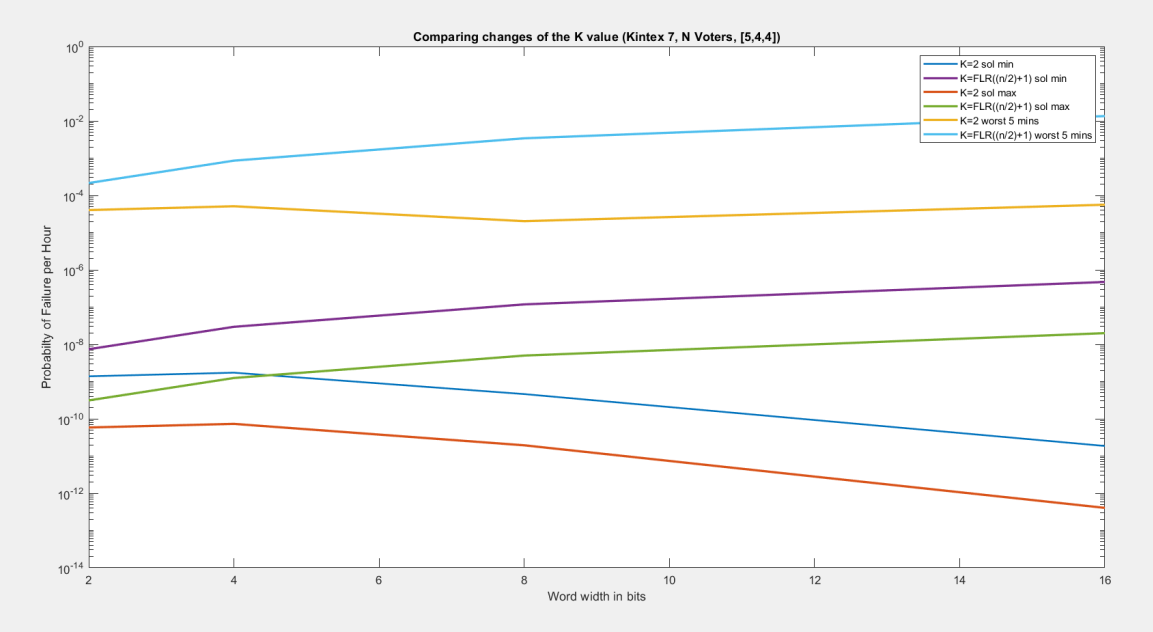

Figure 9. Effects of changing the data width. [Mes14]

modules in parallel is for increasing the reliability of a system. When this limit is reached further gains may be had by changing the voter structure as per Case 2. Improvements may also be had by fixing the K value of the voters, however this must be done thoughtfully so as to not reduce the reliability through the increased likelihood of error collision.

In summary it has been confirmed that through use of redundancy and voter structures it is possible to utalise a COTS FPGA to gain a similar reliability to one which has been Radiation Hardened. It has also been shown that even with the prospect of error collision and reducing data width the $k = 2$ implementation may produce a more reliable system.

5.1 Dependability of results

It will be pointed out here again that these results are simulated and there has been minor concessions to perfect accuracy to ensure within time. These concessions centre around the calculation of the used FPGA primitives for each of the tested structures, as noted in Section 3.5. While this would effect the resulting PFH value produced for a given structure; it is not believed that the difference would be a good trade for the hour (on average) it takes to get exact primitive usage results for each structure under test.

As for future developments it has been shown that different voter structures can produce very different and sometimes more reliable results than either the single or N voter structure. [ALA07]

The other important future development needed is proper validation of these results, via proper calculation of the used primitives in each stage of the structures.

Appendix A
Upset rate comparison of two FPGAs

See next page.

Appendix B
Thanks

The author wishes to thank Dr Daniel Ziener for his help through the supplying of source code to aid this work, in addition to his help as a supervisor. Gratitude is also extended to Dr Leila Bagheriye and Dr Gijs Krijnen for their roles as supervisors.

References

[JCL93] J.C.Laprie. *Dependable Computing and Fault Tolerance: Concepts and Terminology*. 1993.

[Sim00] R. Simpson. *Voyager 1 Instrument Host Information*. 2000. URL: https://pds-rings.seti.org/voyager/spacecraft/vg1host.html.

[ALA07] B. Baykant ALAGOZ. *Hierarchical Triple-Modular Redundancy (H-TMR) Network For Digital Systems*. 2007. URL: https://arxiv.org/ftp/arxiv/papers/0902/0902.0241.pdf.

[Ste08] L. Sterpone. *Electronics System Design Techniques for Safety Critical Applications*. 2008. URL: https://link.springer.com/content/pdf/10.1007%5C%2F978-1-4020-8979-4_2.pdf.

[Lav+10] Christopher Lavin et al. 2010. URL: http://rapidsmith.sourceforge.net/papers/LavinFPL10.pdf.

[Mes14] G. C. Messenger. *Radiation hardening. In AccessScience. McGraw-Hill Education.* 2014. URL: https://doi.org/10.1036/1097-8542.566850.

[Cor15] Xilinx Corp. *Radiation-Hardened, Space-Grade Virtex-5QV FPGA Data Sheet*. 2015. URL: https://www.xilinx.com/support/documentation/data_sheets/ds692_V5QV_Data_Sheet.pdf.

[Gle+15] Robert Glein et al. *Reliability of Space-Grade vs. COTS SRAM-Based FPGA in N-Modular Redundancy*. 2015. URL: https://ieeexplore.ieee.org/document/7231159.

[Sch15] Fred Schenkelberg. 2015. URL: https://accendoreliability.com/standby-redundancy-with-equal-failure-rates-and-perfect-switching/.

[Tri16] Rakesh Trivedi. *A survey of radiation hardening by design (rhbd) techniques for electronic systems for space application*. 2016.

[Cai+19] Chang Cai et al. *Heavy-Ion Induced Single Event Upsets in Advanced 65 nm Radiation Hardened FPGAs*. 2019.

[19] *DEAP Project*. 2019. URL: https://deap.readthedocs.io/en/master/.

[Dub] Elena Dubrova. URL: https://people.kth.se/~dubrova/FTCcourse/LECTURES/lecture7.pdf.

[Rel] ReliWiki. *Life Data Analysis Reference Book by ReliaSoft Corporation*. URL: http://reliawiki.com/index.php/Time-Dependent_System_Reliability_(Analytical).

Device: Primitive (prim)	Count	*Solar Min.*	*Solar Max.*	*Worst Week*	*Worst Day*	*Peak 5 Minutes*
	$n_{\text{device,prim}}$	λ_{prim} in $upsets/day/device$				
V-5QV: CRAM	34,087,072 Bit	$1.75 \cdot 10^{-03}$	$3.35 \cdot 10^{-04}$	$6.04 \cdot 10^{-02}$	$1.46 \cdot 10^{-01}$	$5.27 \cdot 10^{-01}$
V-5QV: BRAM Primitive 512 × 72 Bit	298	$3.44 \cdot 10^{+00}$	$1.11 \cdot 10^{+00}$	$2.33 \cdot 10^{+02}$	$8.58 \cdot 10^{+02}$	$3.12 \cdot 10^{+03}$
V-5QV: FF SET Filter off	81,920	$3.02 \cdot 10^{-03}$	$7.42 \cdot 10^{-04}$	$1.53 \cdot 10^{-01}$	$4.92 \cdot 10^{-01}$	$1.79 \cdot 10^{+00}$
V-5QV: FF SET Filter on	81,920	$1.06 \cdot 10^{-04}$	$2.06 \cdot 10^{-05}$	$9.91 \cdot 10^{-03}$	$3.35 \cdot 10^{-02}$	$1.22 \cdot 10^{-01}$
V-5QV: DSP M-Register	320	$1.22 \cdot 10^{-01}$	$4.34 \cdot 10^{-02}$	$8.75 \cdot 10^{+01}$	$3.26 \cdot 10^{+02}$	$1.18 \cdot 10^{+03}$
V-5QV: DSP other Register	1,280	$2.70 \cdot 10^{-01}$	$9.60 \cdot 10^{-02}$	$1.92 \cdot 10^{+02}$	$7.13 \cdot 10^{+02}$	$2.59 \cdot 10^{+03}$
V-5QV: Configuration Controller	4	$3.51 \cdot 10^{-07}$	$9.89 \cdot 10^{-08}$	$3.57 \cdot 10^{-05}$	$1.25 \cdot 10^{-04}$	$4.56 \cdot 10^{-04}$
K-7 HI: CRAM	75,202,176 Bit	$8.96 \cdot 10^{-01}$	$1.84 \cdot 10^{-01}$	$1.92 \cdot 10^{+01}$	$4.23 \cdot 10^{+01}$	$1.53 \cdot 10^{+02}$
K-7 HI: BRAM Primitive 512 × 72 Bit	445	$3.93 \cdot 10^{+00}$	$1.37 \cdot 10^{+00}$	$6.43 \cdot 10^{+01}$	$1.96 \cdot 10^{+02}$	$7.15 \cdot 10^{+02}$
K-7 HI: FF	407,600	$4.40 \cdot 10^{-03}$	$1.18 \cdot 10^{-03}$	$9.31 \cdot 10^{-02}$	$2.38 \cdot 10^{-01}$	$8.66 \cdot 10^{-01}$
K-7 HI: FF Slice Rst	50,950	$1.39 \cdot 10^{-03}$	$3.89 \cdot 10^{-04}$	$2.63 \cdot 10^{-02}$	$6.76 \cdot 10^{-02}$	$2.45 \cdot 10^{-01}$
K-7 HI: Misc. - Current-Step SEL	1	$7.48 \cdot 10^{-05}$	$2.14 \cdot 10^{-05}$	$1.80 \cdot 10^{-03}$	$5.46 \cdot 10^{-03}$	$1.99 \cdot 10^{-02}$

Figure 10. Data for the upset rate for the two FPGAs, data from CREME96. [Gle+15]