Journal of Universal Computer Science, vol. 24, no. 12 (2018), 1776-1799
submitted: 24/1/18, accepted: 15/11/18, appeared: 28/12/18 © J.UCS

Dynamic Estimation of Temporary Failure in SoC FPGAs for Heterogeneous Applications

J. Kokila
(Department of Computer Science and Engineering
National Institute of Technology, Tiruchirappalli, Tamil Nadu, India
406114002@nitt.edu)

N. Ramasubramanian
(Department of Computer Science and Engineering
National Institute of Technology, Tiruchirappalli, Tamil Nadu, India
nrs@nitt.edu)

Ravindra Thamma
(Manufacturing and Construction Management Department
School of Engineering, Science and Technology, Central Connecticut State University
New Britain, CT, USA
thammarav@ccsu.edu)

Abstract: Recent processors are shrinking in size due to the advancement of technology. Reliability is an important design parameter along with power, cost, and performance. The processors need to be fault tolerant to counter reliability challenges. This work proposes a dynamic thermal and voltage management (DTVM) system which ensures a reasonable level of fault tolerance. The fault tolerance system (FTS) identifies and subsequently can forecast temporary failures at run-time. The temporary failures are dynamically estimated on SoC FPGAs for a class of heterogeneous applications. The dynamic priority scheduling based on absolute deadline is adopted to improve the nature of FTS. Experimental results indicate that the failure rate reduces by 7.2% with the variation of 2% and 12% in temperature and voltage respectively.

Keywords: Fault tolerance system, System on chip design, Priority dynamic scheduling algorithm, Physics of failure and Modified Voltage Lifetime Model
Categories: C.4, C.5, B.8.1, B.6.3, J.6

1 Introduction

Technology has developed at a phenomenal pace over the years, leading to systems with lots of predictable and unpredictable defects. The importance of fault tolerance is increasing. It includes detecting, correcting and preventing multiple failures in terms of performance, power and area. An embedded system plays a major role in daily life, starting from the morning alarm in the mobile phone to listening to music at bedtime. The upcoming co-design of complex embedded system is directed by different authors which may be mainly due to shrinking technology, heterogeneity, dependability, additivity and verification [Teich, 2012]. System on a chip (SoC) is a complete embedded system on a single chip. In the era of nanotechnology, single SoC is of no use; multiple systems on a single chip is the latest development. There are

different variants of multiple SoC such as multiprocessor systems on chips (MPSoCs), chip-level multiprocessing (CMP) and network-on-a-chip (NOC).

The very-large-scale integration (VLSI) system is a Big Data-embedded system that resulted in significant inventions over the years. This led to some new developments in technology such as MPSoCs and CMP. The MPSoC puts the full system or all the essential components together for computation, which may be necessary for multiple programmable processors and complex applications. MPSoCs are widely used in digital signal processing, multimedia, communication, networking and other applications too. A multicore system has multiple CPU cores. This can mean different things depending on the strict architecture, but it principally means that a firm split of the CPU's components is duplicated, hence multiple cores can work in parallel on each operation. This is called CMP. The reliability and security issues in the SoC design are still present in the systems based on latest technologies, hence the reliability is considered in this study. Embedding the function of resistance to an unexpected behavior of a system or multicore to face the challenging issues of dependability [Xia, 2010]. As the latest technology is not limited to performance and size of transistors, we need a CMP or an MPSoC for high performance and superscalar processing.

The field of designing and developing fault-tolerant system (FTS) is vast, which has a major role in all areas starting from academic, business, research, IT, and so on. The main factors an FTS has to consider are dependability and redundancy. As computing becomes omnipresent and influences our everyday lives to a great extent, dependability becomes essential not only for conventional safety, security, and business applications but also for our society as a whole [Henkel, 2012]. There are different types of failures that occur in various systems and redundancy is a common approach to tolerate these. Redundancy is the application of additional hardware or functions that are not strictly necessary to functioning of a system, although they are used in case of failure of other components. Dependability and redundancy play a very important role in the FTS. Technology scaling is not in millions but in trillions hence in all recent mission's fault have to be kept within acceptable ranges. Some common failure mechanisms with associated causes and stresses are listed in [Abraham, 2006]. The proposed dynamic thermal and voltage management system is used to limit the failure rate and hence enhance the life time of the system.

The contributions of the work are as follows:

1. Designing a dynamic thermal and voltage management (DTVM) for heterogeneous applications.
2. The DTVM uses hardware, software, support along with the real-time scheduler to detect and recover from failures at run time.
3. The failure rate has been calculated based on modified voltage life time model and scheduler on an SoC architecture. The TDDB, NBTI, HCI and EM failure rate are used and forecasted at regular intervals to overcome the temporary failures.
4. The constants involved in the formulas have been modified, which are appropriate for the proposed architecture.
5. The time and information redundancy are considered in the FTS design to achieve reliability.

This organization of the article is as follows: Section 2 deals with some related works based on design issue and how to build an FTS with existing system and techniques. Section 3 focuses on the temporary failure mechanism followed by Section 4, which emphasizes proposed system architecture to build an FTS. Section 5 deals with the experimental setup with the heterogeneous application details of the proposal. In section 6 the results are analysed and ending with a conclusion and future work.

2 Related Works

This section describes the challenging issues in the SoC design. Basic requirements for developing the FTS have been discussed using some recent research proposals. The design issues for SoC have been considered for a survey paper which deals with performance, power, time-to-market, reliability and security [Kokila, 2016], which is the route of this proposal. The main base article deals with a distributed resource manager to gather information about a CMP component's strength, and a control mechanism to improve from permanent failures. This architecture can function as long as at least one general-purpose processor is functioning [Pellegrini, 2016]. Heterogeneous computing has been proposed as one way to successfully modify transistors in order to increase performance, energy efficiency, and even reliability within power and thermal constraints. There are numerous critical challenges that are yet to be addressed, including automated synthesis, design space exploration, and run-time management of the heterogeneous dark silicon processors [Shafique, 2014]. The multiprocessor-embedded system uses MicroBlaze and Zynq architecture to implement cardiac monitoring. Those systems consists of an important set of IP cores that are necessary and appropriate for many applications such as UART, GPIO, timer and interrupt [Karim, 2014].

Monitoring and controlling the chip temperature and power utilization are the basic and challenging problems faced for designing an SoC. Dynamic thermal management techniques involving hardware or software or combination of both are used for such systems. The technique used in chip design addresses power dissipation, the modifying task using scheduling, and temperature is regulated using low cost cooling solutions [Yang, 2008]. The performance demands of the current workload are identified in this technology. By using job scheduling and voltage or frequency scaling dynamically, desired performance is obtained while minimizing the energy consumption and thermal imbalance [Sharifi, 2010]. The another method with model prediction control, which is able to follow the precise temperature bound in thermal management. The Dynamic Voltage and Frequency Scaling (DVFS) and task migration are combined to achieve the limited performance degradation and temperature at run-time for different cores [Ma, 2014].

In this real world, fault injection is important because a fault can cause severe damages to the whole system. A survey was made on fault injection techniques and environment, which mainly deals with complex microprocessor-based systems running on software [Kooli, 2014].The fault injection module has been designed using VHDL and tested in FPGA which is consider efficient to study the features of fault tolerance system proposed by [Reddy, 2013] and [Shanthi, 2014].

Intermittent hardware errors are a burst of errors that occurs in the same physical location. In real-world systems, processor failures rates due to non-deterministic intermittent error are 40%. The percentage for inter-level hardware errors that activates a hardware lock is 80%, and that for software and memory, which may lead to data corruption or loss, is 20%. Mechanisms at the software level to mitigate the impact of intermittent faults using detection, diagnosis, and recovery are needed. Intermittent faults lie between the extremes of transient and permanent faults [Rashid, 2015]. A comparison of intermittent and transient faults have been made on a software application to study the impact [Feng, 2013]. Compared to previous ones, this approach requires up to 9.6 times less area overhead while offering 57.6% higher reliability to mask multiple unidirectional SEUs.

There were some fault mitigation methods that use dynamic partial reconfiguration and run-time self-recovery in FPGA platform. These methods concentrate on recovering permanent and transient faults using component relocation [Kim, 2016]. An electrical diagnosis method for wear-out has been proposed with a built-in-self-test (BIST) with statistical analysis. The methodology detects defects, identifies causes of faults due to wear-out in static random access memory (SRAM) cells, and determines the relative frequency of each wear-out mechanism [Dumitriu, 2014].

3 Temporary Failure Mechanisms

Transient and intermittent faults are classified as temporary fault and they are non-recurring and recurring errors respectively. Intermittent faults are non-deterministic errors that occur at different intervals in software and hardware. They are caused by marginal design parameters such as timing problems that result from races, hazards, skew or signal integrity problems caused by crosstalk, ground bounce, and other factors. Transient faults are non-periodic faults and after some time the system comes to its normal state. This type of errors is caused by a disturbance external from the fault site such as radiation, noise, or power disturbance. The main failures dealt in this study are temporary failures [Specification, 2010]: they are time-dependent dielectric breakdown (TDDB), negative-bias temperature instability (NBTI), hot carrier injection (HCI), and soft-error effect (SEE). Table 1 shows the causes, description, variables, and remedies for temporary failures considered in this work.

3.1 Requirement of Redundancy

Fault tolerance requires some form of redundancy in terms of time, hardware, and information. In this work all the three redundancies have been used to design a fault-tolerant SoC. The main aim is to identify the intermittent fault, transient fault, forecast and recover it.

Failure mechanisms	Descriptions	Perimeter cause	Remedies
Time-dependent gate oxide breakdown (TDDB) [Mahmoodi, 2012]	Strong electric field applied to a thin gate oxide film may cross over the withstanding voltage leading to a breakdown	Infant mortality Thin gate oxide High voltage	Constant stress tests
Negative-bias temperature instability (NBTI) [Kothawade, 2011]	High electric fields and high power dissipation due to compact device integration, Both strong fields as well as increased temperatures enhance the NBTI .	Thin gate oxide High Temperature	Input vector control Flipping bit cell data in SRAM Power gating schemes for NBTI
Hot carrier injection (HCI)	The carriers that have gained high kinetic energy due to intense electric fields are injected into the gate oxide of the CMOS device.	Short channel length High voltage	Optimum design and low temperature
Electromigration (EM)	The conducting electrons and dispersed metals together may cause transfer of electrons to an high range which may cause Electromigration.	High temperature	Widen the wire to reduce current density Reduce the frequency and supply voltage Keep the wire length short Reduce buffer size in clock lines.
Soft-error Effect(SEE)	A soft error occurs when an external event causes enough charge disturbance to either change or flip the data state of a device.	Single Event Upset	LogiCORE™ IP Soft Error Mitigation (SEM) IP core

Table 1: Failure mechanisms with description, cause and remedies

1. Time Redundancy: Giving same input at different time should lead to same output, otherwise fault will reoccur. At different time intervals the functional modules are re-executed many times to compare and analyze the outcomes of each one individually. These may allow one to notice the transient or intermittent faults; the permanent faults cannot be detected as they always produce the same wrong output.
2. Hardware redundancy: It is also called structural redundancy because it is accomplished via adding extra equipment to the system in order to tolerate the loss or malfunctioning of some components. Comparing replicated hardware or modules should result in same output, if not an error is detected. This method is used to detect both permanent and temporary faults.
3. Information redundancy: The error correction, and detection are carried out using encoding methods. It is based on coding the data so that the errors can be detected. The Extra bits is attached to the information to allow recovery mechanisms.

In this proposal, the timing redundancy is taken care by executing the different application at the different order and the results have been plotted. For hardware redundancy, the module is replicated and checked.

4 Dynamic Thermal and Voltage management (DTVM)

The proposed system can be used to find fault in different IP cores and correct the appropriate fault in each system by a dynamic thermal and voltage management (DTVM) method. Latest technology is dealing with merely one application is of no use. So in this study, four IP Cores – arithmetic IP, audio controller IP, image compression IP and Strassen matrix multiplication IP – are considered. All these applications play a predominant role in the cluster, cloud and grid computing, big data, and many more areas. The implementation details of each IP cores are explained in the next section.

Dynamic thermal management methods are important to control the temperature and voltage of the SoC board at runtime, improve the reliability and performance of the chip. DTVM is used to manage temperature, voltages and it sustains a high execution time overhead. Examples for this technique are dynamic voltage scaling (DVS), clock gating, and fetch toggling. Software dynamic thermal management (SDTVM) uses an OS with different scheduling and managing techniques. The advantage of this method is lower performance impact, but it cannot guarantee thermal safety. DTVM makes use of both hardware and software mechanisms in a synergistic manner to improve thermal crisis with minimal performance impact. The overall view of the entire system is illustrated in Figure 1.

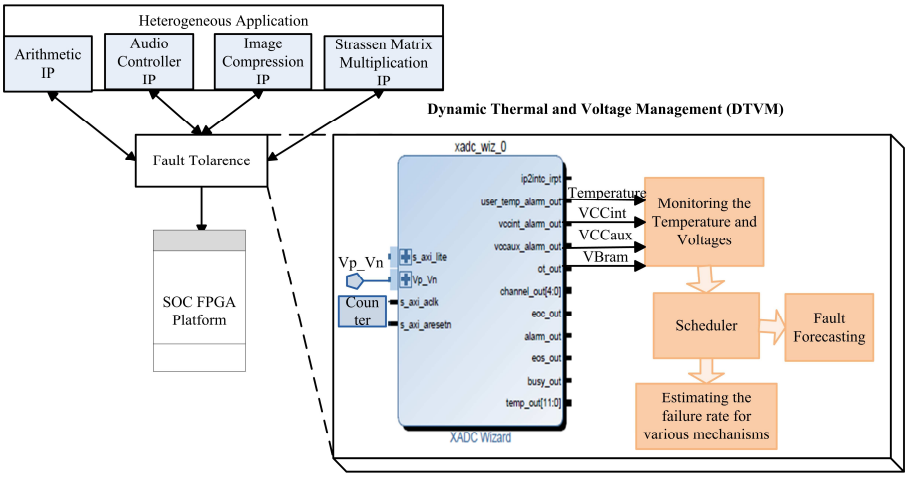

Figure 1: Overall view of the system architecture.

4.1 Hardware Dynamic Thermal and Voltage Management

The hardware uses an XADC block that monitors the temperature and voltage at regular intervals and forecasts the values periodically. The PL includes another hard IP component: the XADC block [Reference manual, 2016]. This is a dedicated set of analog-to-digital converter (ADC) mixed-signal hardware. It has two separate 12-bit ADCs, both competent of sampling external analog input signals at 1 MIPS. The XADC is controlled by the achievement of using the PS_XADC interface block located within the PS. The PS_XADC control block can program itself from software executing on the APU. An initial file with temperature, voltage internal, voltage for auxiliary, and voltage for BRAM will be stored initially with an intermittent fault type in DDR (Double Data Rate) memory. A log file with similar content but with updated values will be created each time when any new application is running on the SoC architecture. If the deadline is detected, then a software scheduling will be involved to recover from the failure. A block diagram for the PS_XADC interface is shown in Figure 2 [Reference manual, 2016].

The hardware uses an XADC block that monitors the temperature and voltage at regular intervals and forecasts the values periodically. The PL includes another hard IP component: the XADC block [Reference manual, 2016]. This is a dedicated set of analog-to-digital converter (ADC) mixed-signal hardware. It has two separate 12-bit ADCs, both competent of sampling external analog input signals at 1 MIPS. The XADC is controlled by the achievement of using the PS_XADC interface block

The PL power domain consists of an implemented XADC that is a hard logic. The functional module Xadc_function can access the PS_XADC interface which is part of the PS without the PL being programmed. To operate XADC and configure PS_XADC, PL must be powered on using PL_JTAG or dynamic reconfiguration port (DRP). The software that is executed in the SDK will control the PS_XADC interface using the XADC wizards. The function will write the no operation and commands to the first in first out (FIFO), which is 32-bits each. The output of FIFO is a 32-bit,

which is serialized by the parallel-to-serial converter for the XADC. There is a programmable idle gap (IGAP) time between the packets to allow time for the XADC to load read data in response to the previous packet. The read data FIFO will shift the corresponding word for every shift output from the command FIFO. If the DRP sends the read command which means shifting out of FIFO that will lead the old content of XADC_DRP's DR register to be shifted out. The XADC DRP's DR register is having the output of the current DRP read after the IGAP period. The next command from word XADC will shift out the current read of the DR register with RDFIFO.

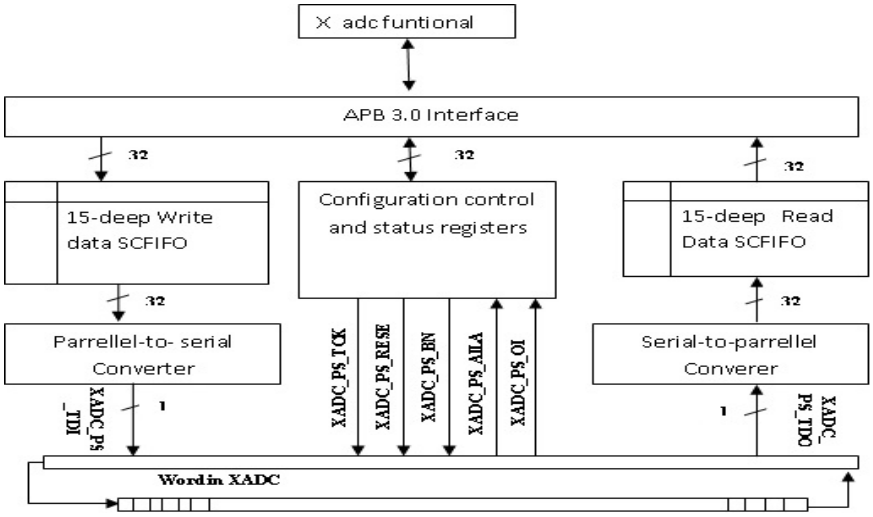

Figure 2: PS-XADC interface.

4.2 Software Dynamic Thermal and Voltage Management

Figure 3 shows a flow graph depicting the overall functional modules in the proposed work. At each scheduling instant, the temperature and voltage are estimated at run-time. Given the outputs of the DTVM and failure mechanism, the module determines a set of thermally safe power states that are able to provide application as close as possible to the workload's requirements. The output of this module is used to set the correct variables and also in assigning application to cores. The log file is stored with some pre-processed data, which will be directed to make decisions. There are four main applications running on the core and the estimation of temperature, and voltage will not be the same each time. At runtime, high temperature and voltage means the failure mechanisms involved are TDDB and NBTI. An application with low temperature and voltage has to be assigned, which is predicted based on the results stored in the log file or pre-processed data. The EM mechanism may be due to low temperature and high voltage for which the application that matches from the log file will be given priority. The fault mechanism HCI depends only on voltage, so temperature-dependent applications may be selected and can be recovered. Therefore, the condition is that if any application is showing wrong output the parity bit for each

application is stored in the memory. The SEU processes will be carried out so that bit flip will be corrected.

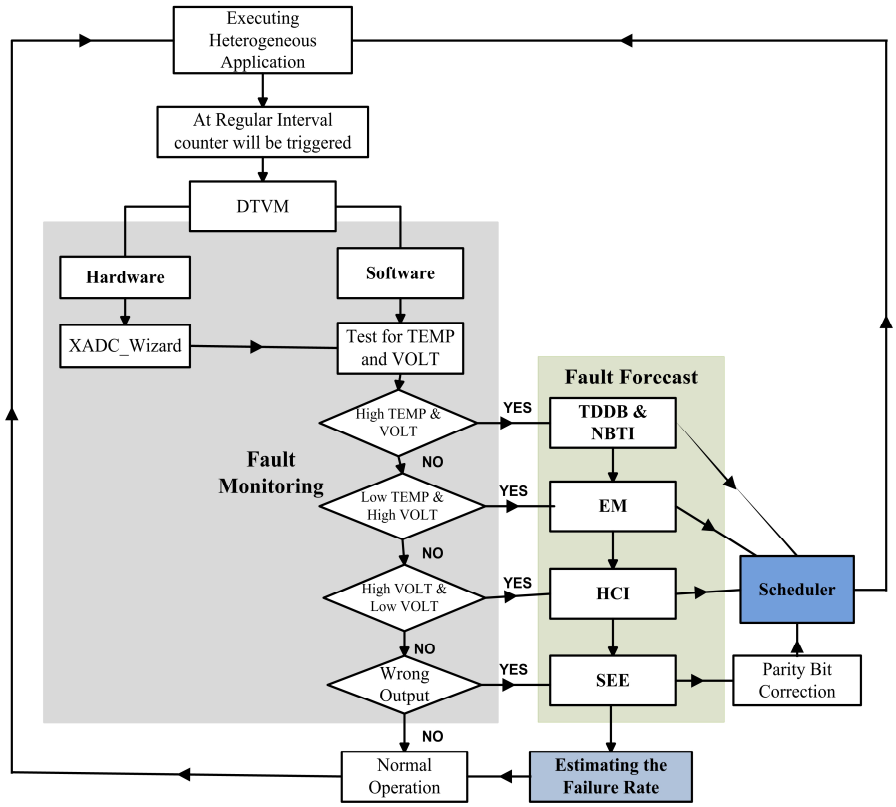

Figure 3: Implementation Flow graph.

4.3 Scheduler

The hardware implementation is the first level followed by scheduler operation at the next level. An optimal pre-emptive scheduling with dynamic priority is used for scheduling the given application. The Absolute deadline based priority dynamic scheduling is used to schedule the application with the earliest deadline first. Each application in an EDF scheduler is assigned a deadline [Hong, 2011]. Every time the application is inserted in the system or completed, the scheduler looks for the task that has the closest deadline and selects it for execution. A monitor must assess to check the scheduler is still able to meet each condition. To implement the EDF-based algorithm, two things should be considered: the deadline of the task (voltage and temperature limit) and the expected time needed to perform the task (regular interval). The scheduler manages the application information such as its current temperature and voltage at regular intervals. The primary role of the scheduler is the conversion of IP cores status dynamically, which is implemented using Algorithm 1. If the temperature and voltage are higher, the threshold limit of the application is rejected: it

will be blocked and not attempted until a task is removed from the chip. Finished tasks remain allocated on-chip, ready to be re-executed until a new task occupies its space [Verber, 2011]. The time complexity of this algorithms is O(nlogn), where n is the number of IP cores.

Algorithm 1. **Preemptive scheduling with Dynamic Priority**

```
1: Assign a First IP Core to the execution unit;
2: Calculate Absolute Deadline (AD);
3: Create a input file(IF) based on Temporary faults of past Voltage (V) and
   Temperature (T) ;
4: AD= Current V and T;
5: while (Traverse through the IF to check priority) do
6:     Assign current priority = priority of new IP core;
7:      if (AD of new IP core less than AD of first IP core) then
8:        Add the new IP core information to the top of the IP;
9:        Update the IF to point to the new IP core and create output File(OF);
10:       if (AD of new IP core less than AD of Last IP core) then
11:        Add the new IP core information to the bottom of the IF
12:       end if
13:      end if
14:     while (At regular intervals Check for IP core priority) do
15:       if (AD of New IP core less than AD of current IP core) then
16:       Allow the current Application to complete;
17:       Add New IP cores application above the current task in the IF;
18:       end if
19:      end while
20:      while (Deadline Monitoring) do
21:        Get the Full execution time of each IP core;
22:        Traverse through the list of IP cores under the priority level;
23:       if (AD of Application less than Execution Time) then
24:         Stop the task using the stop function;
25:       end if
26:     end while
27: end while
```

4.4 Estimating the Failure Rate

The failure rate is the predictable number of failures in the type of device, module, component, or system per given time period. It is the speed at which components are likely to fail. To model FTS, all these intrinsic failure mechanisms have to be considered as any one of them may cause system failure. The intermittent fault occurs repeatedly at the same location by a specific cause of an underlying hardware. However, intermittent faults appear non-deterministically and only under extreme

operating conditions. The individual failure rate model for a TDDB, NBTI, HCI, and EM were proposed in various study [Sunderland, 2014], [Vigrass, 2010], [Bernstein, 2007] and [Bernstein, 2016] which has been used in this work. Table 2 lists the formula and activation energy for each failure mechanisms.

Failure Mechanisms	**Failure Rate Model**	**Activation Energy**
TDDB	$\lambda_{TDDB} = \exp[\gamma_{TDDB} * V_G] * \exp\left[\frac{-E_a TDDB}{\kappa T}\right]$	$E_a TDDB$ = 0.6eV to 0.9eV
NBTI	$\lambda_{NBTI} = \exp[\gamma_{NBTI} * V_G] * \exp\left[\frac{-E_a NBTI}{\kappa T}\right]$	$E_a NBTI$ = 0.1eV to 0.8eV
HCI	$\lambda_{HCI} = \exp[-\gamma HCI / V_D] * \exp\left[\frac{-E_a HCI}{\kappa T}\right]$	$E_a HCI$ = -0.1eV to -0.2eV
EM	$\lambda_{TDDB} = (V_D)^n * \exp\left[\frac{-E_a EM}{\kappa T}\right]$	$E_a EM$ = 0.7eV to 1.1eV

Table 2: Failure rate formula for TDDB,NBTI,HCI and EM

To calculate the failure rate the following steps are listed as follows:

- Temperature and voltage were measured at regular intervals for different applications.
- Forecasting of faults and scheduling in terms of failure mechanisms.
- Failure rate (FIT) was measured at different time intervals for TDDB and HCI.
- Failure rate (time) was determined for NBTI and EM.
- SEUs were taken care using parity bit generation.

The modified failure rate calculation will yield some range of values because the activation energy for each failure mechanism will have a minimum and a maximum range. To calculate the constant the following attributes are considered:

- The CPU frequency is 2.5 MHz.
- The high-range and high-performance input or output support the voltage range of 1.2v to 3.3v.
- The regular intervals of HDTVM is measured in 1000 clock cycles.

The constant is the ratio of regular intervals to the product of CPU frequency and voltage. The above formula yields 0.208 seconds, which is given as input to the γ_{TDDB} , γ_{NBTI} and γHCI constants. The constant n of EM is related to metals line, so the ceramic copper value is taken as 2. The constants involved in each failure

mechanisms are different from each other, but the root causes for all of these intermittent faults are voltage and temperature. The overall system failure can be measured by summing all the failure mechanism rates. As each of them is measured in different range like FITs and seconds, it is not possible to show the system failure output.

5 Experimental evaluation

The proposed work has been implemented in an Zynq®-7000 All Programmable SoC, which offers a new ultimate platform for realizing flexible SoCs. The Xilinx is a vendor which markets the device as an 'All-Programmable SoC' (APSoC) and completely captures its capabilities. The Zynq encompasses two main parts: a Processing System (PS) formed around a dual-core ARM Cortex-A9 processor, and Programmable Logic (PL), which is comparable to that of an FPGA.

5.1 Experimental setup

A custom IP cores for three different applications has been created that executes the simple function for heterogeneous computing. The Arithmetic, Audio and image compression IP has been developed with the IP packager capability of Vivado 2014.4 suite. The core has additional ports so that stimuli can be brought in and the response can be monitored. This way the core can be tested independently without using the PS or software application. The image compression needs the HP port of the PS and CDMA controller in the PL, which is added for our use. Next, an AXI–BRAM controller is added to access the second port of the BRAM through the processor. One can connect the interrupt request line from the CDMA to the input of the GIC of the PS. In this module the experimental setup along with designing the heterogeneous application is explained.

The diagram in Figure 4 represents the completed design. The FTS in ZedBoard with the internal modules involved in PS and PL is shown in orange and dark blue color filling respectively. The three main processes are fault detection, forecasting, and recovery, which are mainly guided by an approach called DTVM.

The DTVM will monitor the application flow and finds the voltage and temperature details at regular intervals. Based on application, the period of detection will change because the heterogeneous core is using different intellectual properties. Fault forecasting is performed by analysing the behaviour of the system based on the physical fault or injection of fault for some experimental needs. The valuation has two aspects: qualitative and quantitative. The qualitative aspect will identify, classify, and rank the failure modes or the event that would lead to system failures. The quantitative aspect will be calculated in terms of probabilities and the extent to which some of the attributes of dependability are satisfied. In our experiment, the qualitative aspect was involved for classifying the intermittent faults.

The fault forecasting based on qualitative mechanism is done for different variations of voltage and temperature. The four conditions are checked and their mean is displayed in the OLED of the ZedBoard and the processes enter into software dynamic thermal and voltage management.

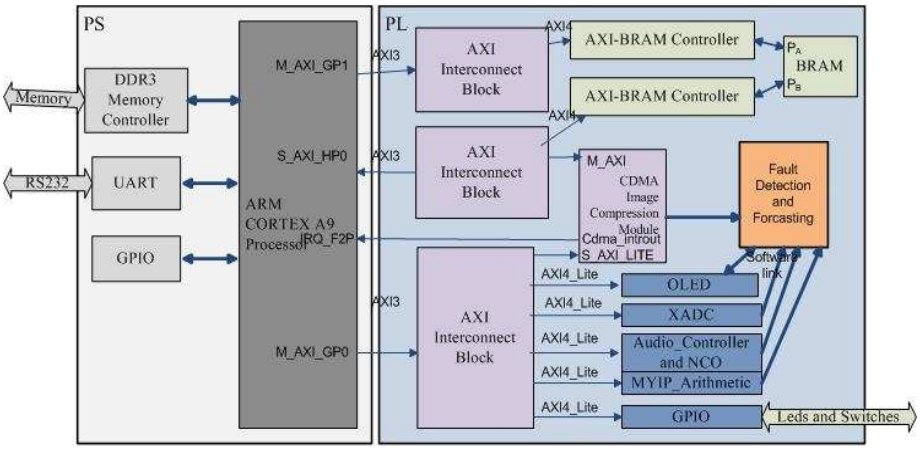

Figure 4: Complete infrastructural design in Zedboard.

5.2 Designing a Heterogeneous Application

Designing heterogeneous application in Zynq board by using all its feature is a challenging task. An FTSoC (Fault Tolerance SoC) is designed over the heterogeneous application to coordinate and control. Because the main concern about the proposal is fault tolerance, the application designed in this project is light and controllable. The implementation is done on Zynq, which is an SoC board that uses hardware-software co-design. There are many applications for Zynq, which include wired and wireless communications, image and video processing, automotive, and high-performance computing. The hardware is designed and implemented in Vivado design suite and software code is implemented in SDK (software development kit). The design issues along with intellectual property involved in each application are as follows:

5.2.1 Arithmetic operation

In all basic technologies, simple operation is very much required. This is a user defined IP, which uses Verilog code to perform simple addition, subtraction, multiplication, division and modulo operations. In Zynq processor the AXI interconnects and interfaces are the main connection between Processing system (PS) and Programmable logic (PL). The defined My_arith IP is connected through the M_AXI_GPIO of the processor to the AXI peripheral. A simple c code has been written to drive the My_arith IP. The input and output are stored in registers. The C code used in this module has build in and user defined functions to make the overall IP to work.

5.2.2 Audio controlling

One of the core will have the digital Audio Processing Functionality. ADAU1761 audio codec mounted on the Zedboard is used for audio processing. It takes audio

input from Line In and provides the data to PS for processing. The ADAU1761 is a low-power, the stereo audio codec with integrated digital audio processing that supports stereo 48 kHz record and playback. The I2C Bus is used to configure the codec by writing to its control registers. The I2S bus(Inter IC Sound) is a de-facto standard for moving audio data from digital systems of SOC to the host computer. The signals in the I2S protocol are two clocks and data lines known as serial audio data in/out lines. The NCO (numerically controlled oscillator) is used to generate a sine wave at the desired frequency. The NCO is implanted in hardware as an AXI core to provide accurate timing.

5.2.3 Image processing

In this core image, the compression technique has been addressed. This is a different functional unit, so we need a separate memory and a high-performance interconnect in PL. In Zynq, multiple interconnections are available between the PS and PL sections with different performance levels for data transfer between the two subsystems. The general purpose (GP) master and slave AXI interconnect is used for peripherals that do not have high bandwidth requirements, for example, switches, LEDs, keyboard, and mouse. There are four high-performance PS slaves to PL master AXI interfaces available for peripherals that need higher bandwidth, for example video and image processing applications. In this module, the process of enabling a high-performance AXI slave port in the PS, adding an AXI central direct memory access (CDMA) controller, and performing direct memory access (DMA) operations between various memories is performed. Choosing an image file from a desktop at runtime compresses it and stores in block RAM (BRAM).

6 Results and Discussion

The results are discussed in terms of reliability and failure mechanisms. The results are analyzed in the following order first resource utilization of overall system along with each application is reported followed by the runtime monitoring of temperature and voltage. The failure estimation for each failure mechanism is measured and finally System failure rate with and without FTS is analyzed.

6.1 Resource utilization

Table 3 lists the need and resource utilization for various applications. Each application is designed individually for different applications, and its resource utilization on the board is reported. The arithmetic IP was designed for 32-bit operands and performs five arithmetic operation by using the LEDs and switches of the zedboard. The audio controller block uses DSP resources for its implementation. The image compression block uses BRAM and direct memory access for processing images, and hence LUT and BRAM utilization is more. The last module is for Strassen matrix multiplication which uses 16-bit IO for 16-bit in and out data. All the IP cores as mentioned above are heterogeneous because their application, utilization, and benchmarks are entirely different.

Heterogeneous Applications	Use of Application	FF	LUTs	MEM LUTs	I/O	B RAM	DSP 48	BUFG
Arithmetic Ips	General purpose Systems	3	5	1	37	0	0	3
Audio Controller IP	Multimedia systems and mobile computing	3	6	1	5	2	3	6
Image Compression Ips	Multimedia systems and Image processing	9	19	3	16	36	1	19
Strassen Matirx	Image processing	3	8	1	16	6	5	7

Table 3: Use of the application and resource utilization of each application

Hardware is designed using Vivado suite and Figure 5, shows the overall utilization details along with the applications. The BRAM and IO utilization are higher since we are using image compression technique for storing images and IO for each IP is different respectively.

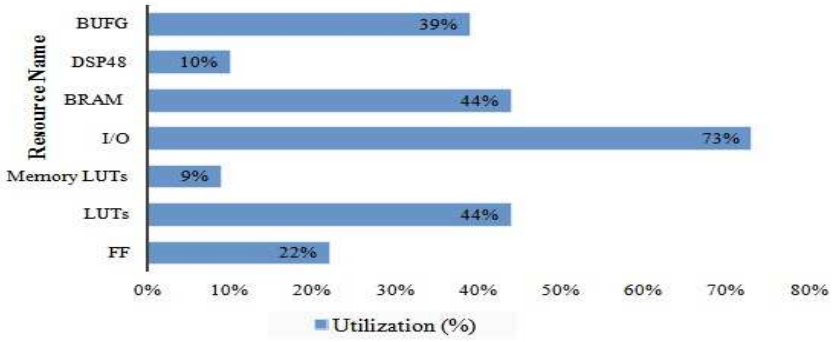

Figure 5: Utilization of overall hardware.

6.2 Run-time temperature and voltage measure

In Table 4, the estimation of voltage and temperature for all iteration is listed and the voltage and temperature are measured in millivolts and Celsius respectively. The difference is too small to determine, but finding failure rate for the following is very important for reliability estimation. The absolute voltage rating for the three voltages is 1.1, 2.0, and 1.1 V respectively. The influence of the temperature and voltage measures for heterogeneous IP cores at different trails is random. The VCCint,

VCCAux, and VCCBram are the PL internal supply voltage, auxiliary supply voltage, and supply voltage for the BRAM memories, respectively.

Heterogenous Application	Temperature (celsius)	VCCint (MilliVolts)	VCCaux (MilliVolts)	Vbram (MilliVolts)
T1	40	996.002	1801.254	997.559
T2	43	995.544	1801.346	995.636
T3	43	997.055	1801.254	995.819
T4	43	995.728	1800.888	995.865
T5	44	995.911	1801.3	995.682
T6	39	992.477	1804.047	994.308
T7	43	994.904	1804.138	994.354
T8	44	994.08	1805.695	994.4
T9	44	994.72	1803.68	993.942
T10	44	993.164	1804.047	994.125
T11	41	993.576	1804.047	994.583
T12	43	994.537	1805.74	993.622
T13	43	993.988	1803.589	994.217
T14	44	995.544	1804.276	995.224
T15	44	994.72	1804.276	995.224

Table 4: DTVM output for heterogeneous application

In Figure 6, the temperature has been measured at regular intervals for all the application with and without DTVM. The hardware module will report the temperature in terms of Celsius, which is converted to Kelvin to estimating the system failure. The temperature measure for DTVM is having gradual increase and decrease based on the application, but without DTVM the temperature is facing random and sudden changes. In trial 3, 11 and 15 the high-performance application is running, hence temperature and voltage increase. In trial 4, and 12 the temperature decrease suddenly which indicates that the low priority application is executing.

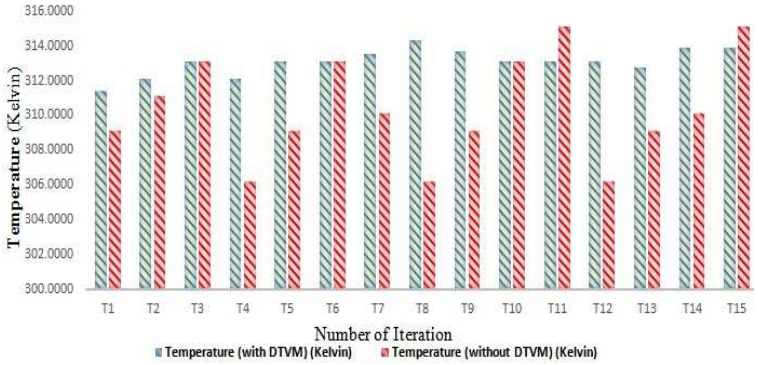

Figure 6: Temperature variation with and without DTVM

In Figure 7, the source and drain voltages are determined for different trials with and without DTVM. The VCCint, VCCAux, and VCCBram were reported as the output of the hardware module which is converted as source and drain voltage for a software module. The voltages are gradually rising and falling for trial from 1 to 15. Without the DTVM the voltages are showing odd behavior in trails 5,6,7,8,9,10, and 14. These changes have to be monitored and managed to get a failure-free system.

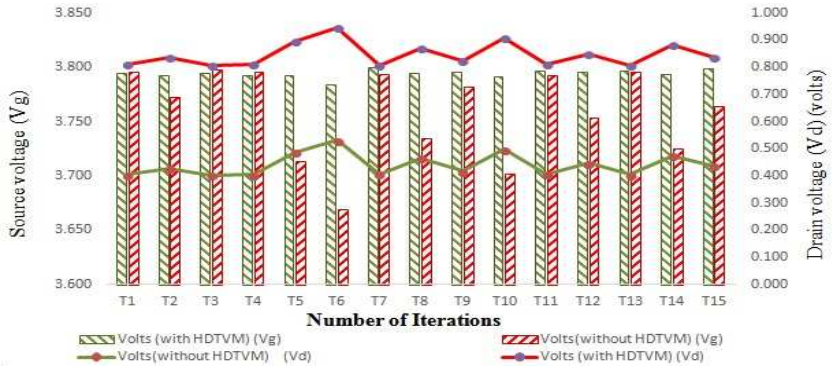

Figure 7: Voltage variation with and without DTVM

6.3 Estimating the failure Rate for each failure mechanisms

Each application will create stress on the board, which measures in term of many failure modes. Figure 8, and 9 illustrate how the temperature and source voltage cause changes on TDDB and NBTI with and without FTS. In Figure 10 and11 shows the temperature and drain voltage changes on EM and HCI. The graphs shown above examines the failure rate for the minimum activation energy of the heterogeneous system in sync. The green and red color shading indicate the DTVM and without DTVM respectively.

Dynamic values of temperature and voltage have to be determined at regular intervals because sometimes the board will be in normal temperature and sometime air condition will be on. Each time application run on ZedBoard may yield random values so the FTS should be flexible and distributed. In the real world, the failure rate based on the temperature and voltage calculation is challenging one, but this proposal has made an attempt to overcome. High temperature and high voltage will lead to the two failure mechanism NBTI and for TDDB that can be seen in trial 5,8 and 14. Because we cannot exceed a particular limit, a threshold limit of 45 degree Celsius can be used and that voltage is limited to 4.2 volts. The TDDB may cause an increase in leakage and short circuit problems, which can be avoided by this mechanism. The change in voltage test is adopted for EM. Examining the Low temperature and high voltage will lead to HCI. A comparative analysis of EM and HCI to TDDB and NBTI shows that voltage will mainly affect the failure rate.

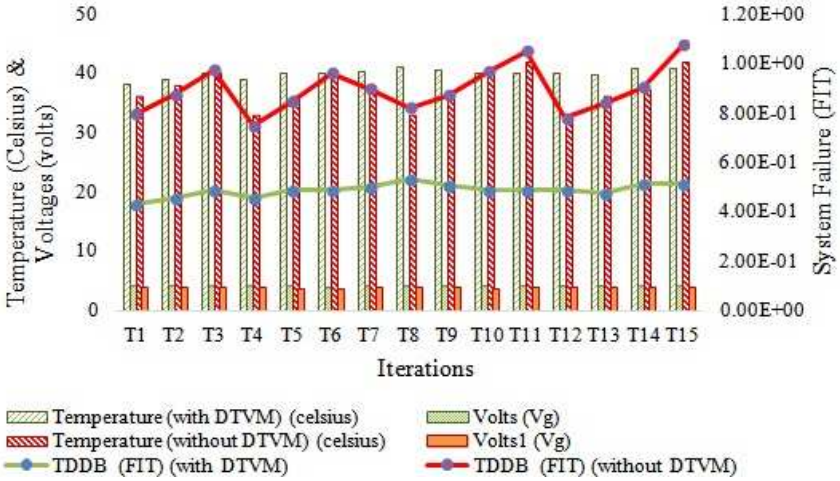

Figure 8: Failure rate for the TDDB with and without DTVM

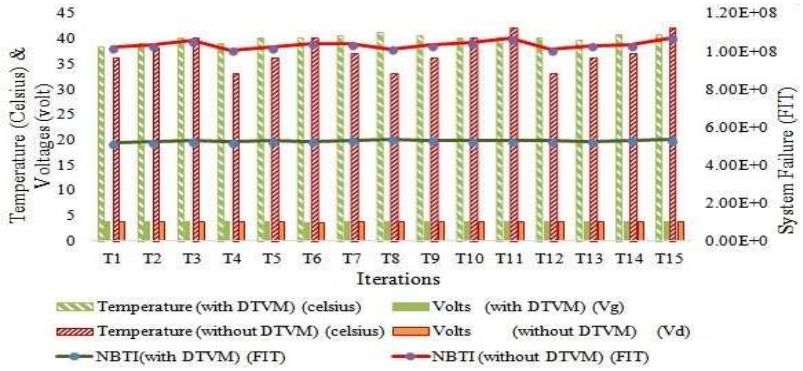

Figure 9: Failure rate for the NBTI with and without DTVM

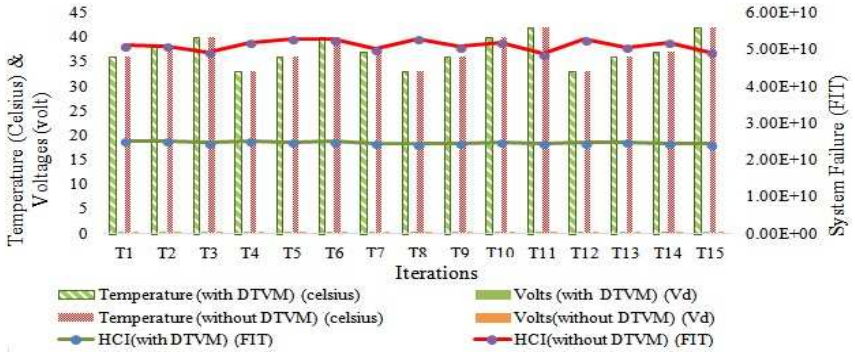

Figure 10: Failure rate for the HCI with and without DTVM

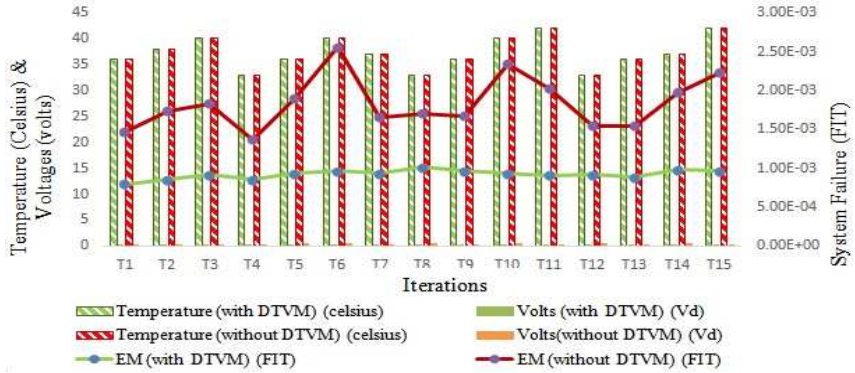

Figure 11: Failure rate for the EM with and without DTVM

6.1 Estimating system failure rate

Table 5 and 6 list the failure rate of different failure mechanism based on the modified voltage life time model. Finally to analyze the accuracy of the systems the application with and without DTVM is adopted.

Figure. 12 depicts the system failure decrease in the proposed work with no Fault tolerance systems. In different order heterogeneous IP core implementation is compared and analyzed. The system failure is very random without DTVM and constant with DTVM.

The CHStone is a real-world application benchmark suite which consists of the 12 programs. Each program has been developed in C-based high-level synthesis (HLS) and suitable for the arithmetic, media processing, security, and microprocessor. The DFADD, ADPCM, JPEG and DFMUL are the programs selected from the CHStone benchmark suit and scaled down for 16-bit, 16-bit, 8-bit and 16-bit input and output respectively. The benchmark suite is analyzed and implemented in Xilinx Vivado HLS [Dubey, 2015]. The detail discussion of source description along with resource utilization, line of code, latency for all the benchmark program is referred in [Hara, 2009].

Trails	Tem (K)	Volts (Vg)	Volts (Vd)	TDDB (FIT)	NBTI (FIT)	HCI (FIT)	EM (FIT)	System failure (without DTVM)
T1	309	3.80	0.404	3.65E-01	5.03E+07	2.59E+10	6.59E-04	2.59E+10
T2	311	3.77	0.427	4.20E-01	5.13E+07	2.60E+10	8.72E-04	2.60E+10
T3	313	3.80	0.402	4.87E-01	5.28E+07	2.46E+10	9.10E-04	2.46E+10
T4	306	3.80	0.405	2.93E-01	4.85E+07	2.69E+10	5.10E-04	2.69E+10
T5	309	3.71	0.487	3.59E-01	4.95E+07	2.82E+10	9.56E-04	2.82E+10
T6	313	3.67	0.530	4.75E-01	5.15E+07	2.78E+10	1.59E-03	2.78E+10
T7	310	3.79	0.407	3.93E-01	5.09E+07	2.56E+10	7.25E-04	2.57E+10
T8	306	3.73	0.466	2.90E-01	4.79E+07	2.87E+10	6.76E-04	2.87E+10
T9	309	3.78	0.418	3.64E-01	5.02E+07	2.63E+10	7.03E-04	2.63E+10
T10	313	3.70	0.499	4.78E-01	5.18E+07	2.71E+10	1.40E-03	2.71E+10
T11	315	3.79	0.408	5.60E-01	5.40E+07	2.42E+10	1.10E-03	2.43E+10
T12	306	3.75	0.446	2.91E-01	4.81E+07	2.81E+10	6.20E-04	2.82E+10
T13	309	3.80	0.404	3.65E-01	5.03E+07	2.59E+10	6.58E-04	2.59E+10
T14	310	3.73	0.475	3.87E-01	5.02E+07	2.75E+10	9.89E-04	2.76E+10
T15	315	3.76	0.436	5.57E-01	5.37E+07	2.50E+10	1.26E-03	2.50E+10

Table 5: System Failure without DTVM in which Tem(temperature in Kelvin)

The heterogeneous application used in this work is compared with the selected benchmark program, priorities and allow to run for 15 trails. Figure 12 shows the system failure rate for heterogeneous application with and without DTVM and CHStone benchmarks with DTVM respectively. The heterogeneous applications

specify more failure rate without DTVM, but with DTVM the used application and benchmark programs were seem to be same.

Trails	Tem (K)	Volts (Vg)	Volts (Vd)	TDDB (FIT)	NBTI (FIT)	HCI (FIT)	EM (FIT)	System failure (with DTVM)
T1	311	3.795	0.405	4.31E-01	5.17E+07	2.52E+10	8.02E-04	2.53E+10
T2	312	3.793	0.407	4.53E-01	5.21E+07	2.51E+10	8.61E-04	2.51E+10
T3	313	3.795	0.405	4.87E-01	5.28E+07	2.47E+10	9.24E-04	2.47E+10
T4	312	3.792	0.408	4.53E-01	5.21E+07	2.51E+10	8.61E-04	2.51E+10
T5	313	3.793	0.407	4.87E-01	5.28E+07	2.48E+10	9.34E-04	2.48E+10
T6	313	3.784	0.416	4.86E-01	5.27E+07	2.50E+10	9.74E-04	2.51E+10
T7	314	3.799	0.401	5.01E-01	5.31E+07	2.44E+10	9.33E-04	2.45E+10
T8	314	3.794	0.406	5.29E-01	5.35E+07	2.44E+10	1.02E-03	2.44E+10
T9	314	3.795	0.405	5.07E-01	5.32E+07	2.45E+10	9.68E-04	2.46E+10
T10	313	3.791	0.409	4.87E-01	5.28E+07	2.48E+10	9.41E-04	2.49E+10
T11	313	3.797	0.403	4.87E-01	5.28E+07	2.46E+10	9.14E-04	2.47E+10
T12	313	3.796	0.404	4.87E-01	5.28E+07	2.47E+10	9.20E-04	2.47E+10
T13	313	3.797	0.403	4.76E-01	5.26E+07	2.47E+10	8.91E-04	2.48E+10
T14	314	3.794	0.406	5.14E-01	5.33E+07	2.45E+10	9.89E-04	2.46E+10
T15	314	3.799	0.401	5.15E-01	5.33E+07	2.43E+10	9.64E-04	2.44E+10

Table 6: System Failure with DTVM in which Tem(temperature in Kelvin)

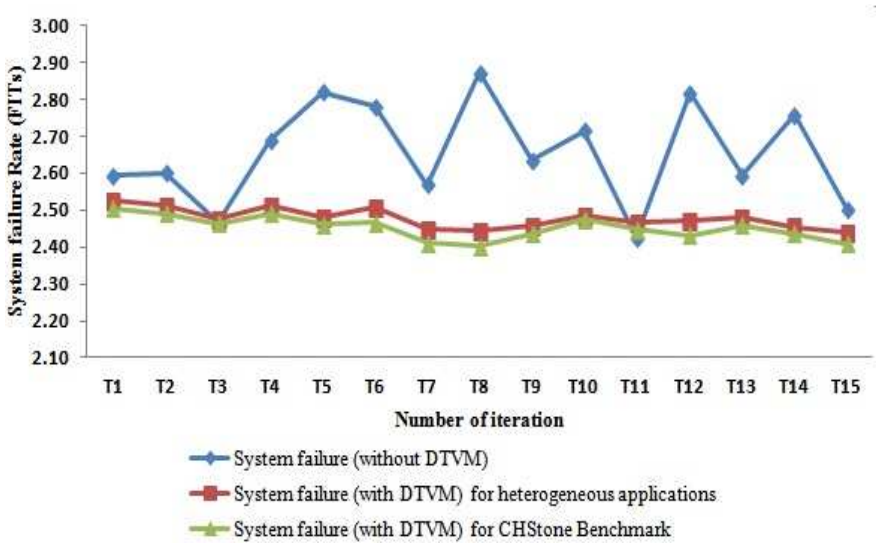

Figure 12: System failure rate with DTVM and without DTVM

In Table 7. the area , power, temperature, voltage, execution time and system failure is reported for the system with and without FTS. The utilization of resources measures the area in which the overhead is 8.0%. The power is noted in the vivado environment, which is indicated in Watt. The Temperature and voltage variations is measured and is limited by the proposed approach. The execution time for DTVM is 19,000 cycles more than the system without DTVM. The System faiure rate is decreased by 7.20% for the entire system. The system failure is reduced from 2.66E+10 to 2.48E+10, by compromising area, power and execution time by 5% to 10%.

Resource Utilization	**DTVM**	
	With	**Without**
Area	62%	54%
Total On chip-Power	1.415 W	1.815 W
Temperature variation	0.9%	2.8%
Voltage variation	2%	13.8%
Execution time	~ 152000	~133000
Average System Failure	2.48E+10	2.66E+10

Table 7: Comparison of Hardware analysis in system with and without DTVM.

7 Conclusion

The proposed work mainly concentrates on the reliability of an SoC design. The physics of failure is a very important issue in chip designing. The estimated failure rate for the failure mechanism shows less fraction of values; it is optional in case of small application, but in heterogeneous type it is more essential. The operating temperature and voltages are measured for heterogeneous applications at regular intervals along with frequency to concentrate on device wear-out. The DTVM is a hardware and software-based approach to forecast the physical type of failure mechanism due to variables involved and to schedule the application to recover from the failure. Different traces of output have been measured for latest CHStone benchmark to illustrate how the parameters affect the system failure.This model has the unique feature of designing an FTS for the temporary failures on the SoC design. The future enhancement is to use a higher-end board such as Zynq® UltraScale+™ MPSoC to monitor and estimating the failure rate of NOC, cloud, and big data. The life time of the chip will be enhanced by detecting and recovering from temporary failures.

References

[Abraham, 2006] Abraham, Ajith, and Crina Grosan. "Automatic programming methodologies for electronic hardware fault monitoring." Journal of Universal Computer Science, vol. 12, 408-431 (2006).

[Bernstein, 2007] J. B. Bernstein, "Physics based reliability qualification", IEEE International Reliability Physics Symposium (2007).

[Bernstein, 2016] Bernstein, Joseph B., Alain Bensoussan, and Emmanuel Bender. "Reliability prediction with MTOL." Microelectronics Reliability (2016).

[Dubey, 2015] Dubey, Anuj, Ashish Mishra, and Shrikant Bhutada. "Comparative Study of CH Stone Benchmarks on Xilinx Vivado High Level Synthesis Tool."

[Dumitriu, 2014] Dumitriu, Victor, Lev Kirischian, and Valeri Kirischian. "Run-Time Recovery Mechanism for Transient and Permanent Hardware Faults Based on Distributed, Self-organized Dynamic Partially Reconfigurable Systems."

[Feng, 2013] Feng, Chaochao, et al. "Addressing transient and permanent faults in NoC with efficient fault-tolerant deflection router." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 21.6, 1053-1066(2013).

[Hara, 2009] Hara, Yuko, et al. "Proposal and quantitative analysis of the CHStone benchmark program suite for practical C-based high-level synthesis." Journal of Information Processing 17 (2009): 242-254.

[Henkel, 2012] Henkel, Jörg, et al. "Design and architectures for dependable embedded systems." Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2011 Proceedings of the 9th International Conference on. IEEE, (2011).

[Hong, 2011] Hong, Chuan, et al. "Efficient on-chip task scheduler and allocator for reconfigurable operating systems." IEEE Embedded Systems Letters 3.3, 85-88, (2011).

[Karim, 2014] Karim, Mohammed. "A MicroBlaze-based Multiprocessor System on Chip for real-time cardiac monitoring." Multimedia Computing and Systems (ICMCS), 2014 International Conference on. IEEE, (2014).

[Kim, 2016] Kim, Woongrae, et al. "Built-in self-test methodology with statistical analysis for electrical diagnosis of wearout in a static random access memory array." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 7 (2016): 2521-2534.

[Kokila, 2016] Kokila, J., N. Ramasubramanian, and S. Indrajeet. "A Survey of Hardware and Software Co-design Issues for System on Chip Design." Advanced Computing and Communication Technologies. Springer Singapore, 41-49, (2016).

[Kooli, 2014] Kooli, Maha, and Giorgio Di Natale. "A survey on simulation-based fault injection tools for complex systems." Design and Technology of Integrated Systems In Nanoscale Era (DTIS), 2014 9th IEEE International Conference On. IEEE, (2014).

[Kothawade, 2011] Kothawade, Saurabh. Design of Negative Bias Temperature Instability (NBTI) Tolerant Register File. Diss. Utah State University, (2011).

[Ma, 2014] Ma, Jian, et al. "Hybrid dynamic thermal management method with model predictive control." Circuits and Systems (APCCAS), 2014 IEEE Asia Pacific Conference on. IEEE, (2014).

[Mahmoodi, 2012] Mahmoodi, Hamid. "Reliability enhancement of power gating transistor under time dependent dielectric breakdown." VLSI and System-on-Chip, 2012 (VLSI-SoC), IEEE/IFIP 20th International Conference on. IEEE, (2012).

[Pellegrini, 2016] Pellegrini, Andrea, and Valeria Bertacco. "Cardio: CMP adaptation for reliability through dynamic introspective operation." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 33.2,265-278 (2014).

[Rashid, 2015] Rashid, Layali, Karthik Pattabiraman, and Sathish Gopalakrishnan. "Characterizing the impact of intermittent hardware faults on programs." IEEE Transactions on Reliability 64.1,297-310 (2015).

[Reddy, 2013] Reddy, G. Gopinath, A. Rajasekhar Yadav, and Y. Mahesh. "High Speed Fault Injection Tool Implemented With Verilog HDL on FPGA for Testing Fault Tolerance Designs." G. Gopinath Reddy et al Int. Journal of Engineering Research and Application 3. 96-99 (2013).

[Reference manual, 2016] Zynq-7000 All programmable SOC technical reference manual (2012) http://www.cl.cam.ac.uk/research/srg/han/ACS-P35/zynq/Zynq-7000-TRM.pdf

[Shafique, 2014] Shafique, Muhammad, et al. "Dark silicon as a challenge for hardware/software co-design: invited special session paper." Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis. ACM, (2014).

[Shanthi, 2014] Shanthi, G. Roja, et al. "VHDL Implementation of High Speed Fault Injection Tool for testing FPGA based designs." IJRCCT 3.11, 1495-1501 (2014).

[Sharifi, 2010] Sharifi, Shervin, Ayse Kivilcim Coskun, and Tajana Simunic Rosing. "Hybrid dynamic energy and thermal management in heterogeneous embedded multiprocessor SoCs." Proceedings of the 2010 Asia and South Pacific Design Automation Conference. IEEE Press,(2010).

[Specification, 2010] Specification, J. E. D. E. C. "Failure Mechanisms and Models for Semiconductor Devices." Paper No. JEP 122 (2010).

[Sunderland, 2014] Sunderland, David A., et al. "Collaborative approach for practical modeling of microcircuit failures in high-reliability applications." 2014 IEEE International Reliability Physics Symposium. IEEE,(2014).

[Teich, 2012] Teich, Jürgen. "Hardware/software codesign: The past, the present, and predicting the future." Proceedings of the IEEE 100.Special Centennial Issue,1411-1430 (2012).

[Verber, 2011] Verber, Domen. "Hardware implementation of an earliest deadline first task scheduling algorithm." Informacije MIDEM 41.4,257-263, (2011).

[Vigrass, 2010] Vigrass, William J. "Calculation of semiconductor failure rates." Harris Semiconductor (2010).

[Xia, 2010] Xia, Bingbing, et al. "A fault-tolerant structure for reliable multi-core systems based on hardware-software co-design." Quality Electronic Design (ISQED), 2010 11th International Symposium on. IEEE, (2010).

[Yang, 2008] Yang, Jun, et al. "Dynamic thermal management through task scheduling." Performance Analysis of Systems and software, 2008. ISPASS 2008. IEEE International Symposium on. IEEE, (2008).