DPA to Rectify Transient Faulty Nodes in Effective Manner

P. S. Balamurugan
B. E., M. E, (Ph.D) Research Scholar
Anna University of Technology,
Coimbatore

Dr. K.Thanushkodi
Director
Akshaya College of Engineering and Technology,
Coimbatore

ABSTRACT
Wireless sensor networks faces a number of challenges; a wireless sensor network which includes a number of sensor nodes must provide reliability and fault tolerance against a number of odds such as scalability, hardware, environmental conditions, power and energy factors. In this paper, we address these two issues of Reliability and Fault Tolerance using mirror nodes. We demonstrate that increased reliability can be achieved by using mirror nodes and the costs could be maintained by implementing the Direct Processor Access(DPA). Experimental results on the benchmarks data set show that our proposed system based on Direct Processor Access outperforms the other well-known methods such as the Distributed Deviation Detection, Distributed anomaly detection, Intrusion detection for routing attacks, Statistical en route filtering and Abnormal Relationship Tests(ART). The improvement in performance using DPA is very high, particularly, for the graphical and network processes (6.8 percent improvement). Statistical Tests also demonstrate higher fault tolerance and improvement in performance for our method. Finally, we show that our system is robust and is able to handle faulty sensor nodes without compromising performance.

Keywords
Wireless Sensor Networks, Faulty Sensor Nodes, Fault Tolerance, Direct Processor Access, Mirror Nodes

1. INTRODUCTION
A wireless sensor network (WSN) is a collection of nodes organized in a network where each node consists of one or more microcontrollers, CPU's or DSP chips, a memory and a RF transceiver, a power source such as battery. It also accommodates various sensors and actuators. The nodes communicate without wire (wireless) and often organize itself after being deployed in an ad hoc fashion . The intrinsic properties of individual sensor nodes pose additional challenges to the communication protocols in terms of energy consumption.

The reliability or fault tolerance is yet another issue. Some sensor nodes may fail or be blocked due to lack of power, physical damage or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network.

1.1. Challenges of Wireless Sensor Networks
In monitoring sensor networks, data coming from various streams of the sensor nodes have to be examined dynamically and combined into normal patterns in order to detect potential anomalies. Due to the requirement for the support of mission critical applications in many cases, the sensors must possess mechanisms for securing communications and for validating the collected data. Several attack scenarios that exploit the weaknesses of WSNs has been identified and the scale of deployments of WSNs requires careful decisions and tradeoffs among various security measures. These issues are taken into consideration and mechanisms to achieve a higher level of security and reliability has been proposed in these networks.

2. WIRELESS SENSOR NETWORK WITH MIRROR NODES

In this investigation, we assign a mirror node for each master node. At a time only a single node will be activated, either master node or mirror node. The mirror node will be in active state only in the absence of the master node. Whenever master node is identified as faulty node, the primary node will activate the mirror node and isolate the master node from the sensor network. This process helps to improve the availability of the sensor networks during threats and disaster and its performance is shown in Fig 1.

Table 1. Performance of Sensor Network with Mirror Nodes

Data Transmission rate in sensor network with mirror nodes (MB/ s)	Data Transmission rate in sensor network without mirror nodes(MB/s)
0.2	0.1
0.4	0.22
0.45	0.33
0.52	0.34
0.56	0.41

0.59	0.42
0.61	0.43
0.67	0.46
0.69	0.51
0.74	0.52
0.78	0.54

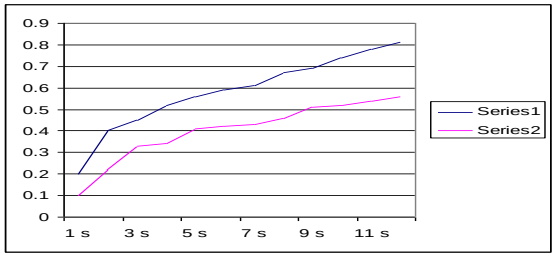

Fig 1:Performance of sensor network with mirror nodes

In the previous investigation, the availability of the sensor networks was increased to the optimum level. But the sensor networks consist of a large number of sensor nodes and implanting a mirror node for each individual sensor node will increase the overall cost of the networks. If the cost of the network is more expensive than deploying traditional sensors, then the sensor network is not cost justified.

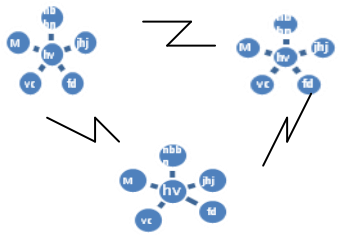

Fig 2: Structure of Sensor node with asynchronous mirror node

Table 2: Performance of sensor network with asynchronous mirror nodes

Data Transmission rate in sensor network with Asynchronous mirror (MB/s)	Data Transmission rate in sensor network without Mirror nodes (MB/ s)
0.19	0.1
0.35	0.22
0.412	0.33
0.452	0.34
0.516	0.41
0.58	0.42
0.59	0.43
0.64	0.46
0.656	0.51
0.714	0.52
0.728	0.54
0.781	0.56

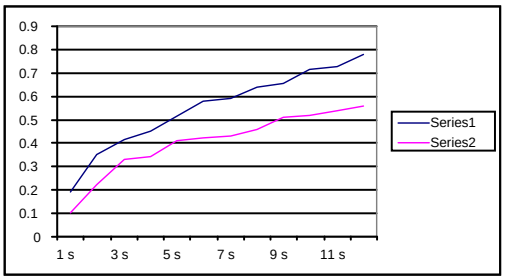

Fig 3: Performance graph for asymmetric mirror nodes in sensor network

Table 3: Comparison of time delay for a sensor network with asynchronous mirror node and without mirror node

Time delay for a sensor network without mirror in sensor network (ns)	Time delay for a sensor network with asynchronous mirror in sensor network (ns)
0.2	0.1
0.31	0.23
0.42	0.28
0.46	0.31
0.51	0.344
0.62	0.38
0.74	0.51
0.89	0.64
0.916	0.7

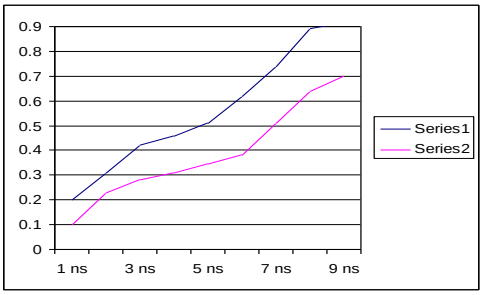

Fig 4: Time delay for a sensor network having mirror node and without mirror node

From these comparisons we could conclude that introducing mirror nodes will obviously improve the performance of sensor networks. Hence sensor networks with mirror nodes can be implemented in real time systems where time constraints are strictly followed and cost factor is not an issue.

3. COST EFFECTIVE ARRAY BASED DPA

To increase the accessing speed and to attain an efficient memory access, a new methodology is employed in the proposed system. The methodology is termed as Array Methodology. By this methodology the processor will interact with the RAM device in an array fashion by which the RAM will be divided into arrays and each array will be allotted for a default program to be utilized. Thus the processor can access the data and codes easily, by searching in the specified memory location. By this procedure the value n will not represent the total cache memory space but it will represent only the value of an array. When a program needs more memory space than the allotted memory by using artificial intelligence we can combine the memory and utilize it to execute the program. The process of combining memory can be done by calculating the frequently used program or FIFO method. This method is highly applicable when we need to run a program which needs memory space less than n/11 in a high RAM capacity machine. In this methodology the work of the processor is simplified by allowing it to search in the allotted array.

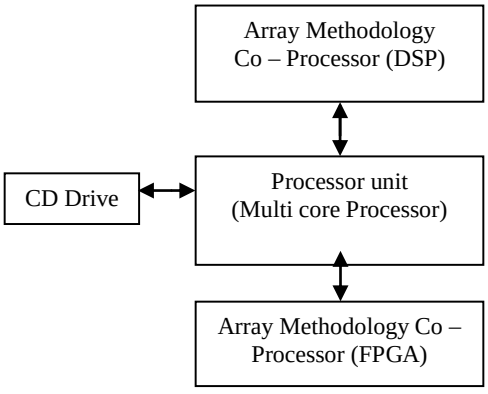

Fig 5: Block Diagram of Array methodology Based Co - Processor Design

Another important concern that influences the processor performance is the heat sink designed for the IC and the design has to be chosen between its size and performance. The impact of CMOS technologies on substrate and metal line temperatures have resulted in improved reliability and better performance of the devices and interconnections. 74 % of processor failures are due to thermal factors and high power sources such as power dissipation, temperature relation, a method for full chip temperature calculation and implications on the design of high performance low power VLSI circuits. By spacing the memory in an array manner, the cache port's accessibility could be improved. This can be known from the percentage of hit ratio tabulated for the different programs.

4. EXPERIMENTS

The experiments comparing the performance of the various processors to be used in place of the mirror nodes and their time consumption are analyzed using the Benchmarks such as the Graphics Benchmarks, Network Benchmarks, Desktop Benchmark, etc.

4.1 Performance of multi core processors with DSP co processors

The performances of a multi core processor in improving the time factor for executing the graphical bench mark program, where a co processor is used along with a DSP processor and is shown in the following figure.

Table 4:Performance of Multi core processor with a DSP co processor in executing Graphical Benchmark programs

Program Specification	Physical capacity of program	Time consumption in seconds
Graphical Benchmark	1.2GB	0.58
Graphical Benchmark	1.14 GB	0.52
Graphical Benchmark	1.10 GB	0.52
Graphical Benchmark	1.04 GB	0.51
Graphical Benchmark	1. GB	0.5
Graphical Benchmark	850 MB	0.62
Graphical Benchmark	780MB	0.65

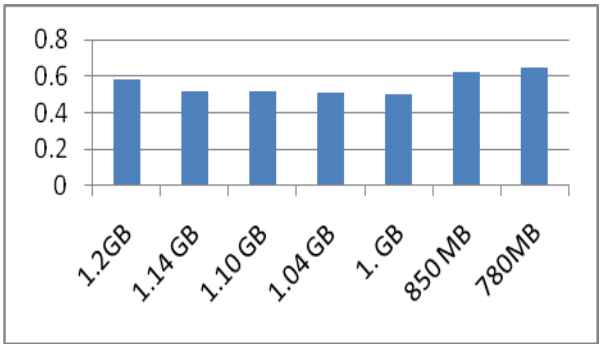

Fig 6: Performance of Multi core processor with a DSP co processor in executing Graphical Benchmark programs

Table 5: Performance of Multi core processor with a DSP co processor in executing Network Benchmark programs

Program Specification	Physical capacity of the program	Time consumption in seconds
Network Benchmark	790 MB	0.65
Network Benchmark	1.05 GB	0.51
Network Benchmark	1.08 GB	0.52
Network Benchmark	875 MB	0.62
Network Benchmark	805 MB	0.62

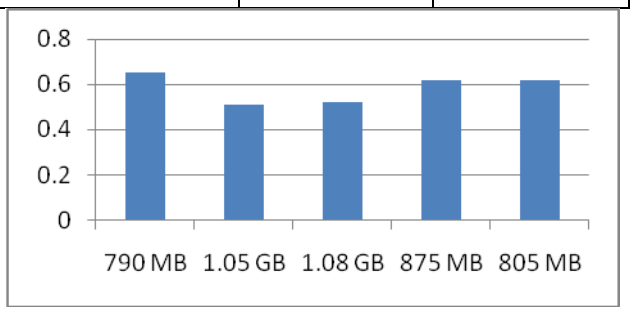

Fig 7:Performance of Multi core processor with a DSP co processor in executing Network Benchmark programs

The DSP co processor has to be designed in such a way that the sum of the clock speed of all the co processors should be equal to the clock speed of the processor which is used to enhance synchronization. The number of co processor depends on the speed of the processor. For instance, if P represents the speed of the processor having four co processors and each of the co processor's clock speed is Cp1, Cp 2, Cp 3 and Cp4 respectively, then the speed P of the DSP co processor is derived from the equation

$$P = Cp1 + Cp2 + Cp3 + \ldots\ldots + Cpn \qquad (1)$$

The same programs were executed in a multi core processor in which the performance was measured initially. The clock speed of the processor is observed and accordingly the speed of the co processor is designed. When a single multi core processor has to be designed with a clock speed of 2 GHz, then two co processors are implemented, each with a clock speed of 1 GHz.

Table 6:Performance of processors without co processors in executing Document Benchmark Programs

Program Specification	Physical capacity of program	Time consumption in seconds
Document Benchmark	300 MB	0.64
Document Benchmark	180MB	0.95
Document Benchmark	150 MB	0.95
Document Benchmark	144 MB	1.11
Document Benchmark	135 MB	1.1
Document Benchmark	80 MB	1.31
Document Benchmark	75 MB	1.31

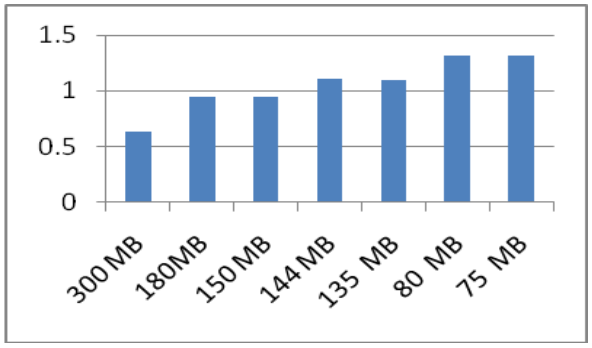

Fig 8: Performance of processors without co processors in executing Document Benchmark Programs

Table 7: Performance of processors with co processors in executing Document Benchmark Programs

Program Specification	Physical capacity of program	Time consumption in seconds when co processor is used
Document Benchmark	300 MB	0.63
Document Benchmark	180MB	0.96
Document Benchmark	150 MB	0.96
Document Benchmark	144 MB	1.12
Document Benchmark	135 MB	1.1

Document Benchmark	80 MB	1.30
Document Benchmark	75 MB	1.32

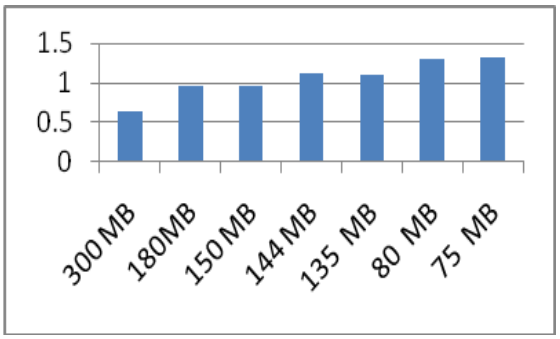

Fig 9: Performance of processors with co processors in executing Document Benchmark Programs

Table 8: Performance Comparisons between a processor without co processor and processors with co processor in executing Graphical Benchmark Programs

Program Specification	Physical capacity of program	Time consumption in seconds	Time consumption in seconds by using co processor
Graphical Benchmark	1.2GB	0.58	0.50
Graphical Benchmark	1.14 GB	0.52	0.45
Graphical Benchmark	1.10 GB	0.52	0.45
Graphical Benchmark	1.04 GB	0.51	0.51
Graphical Benchmark	1. GB	0.5	0.45
Graphical Benchmark	850 MB	0.62	0.58
Graphical Benchmark	780MB	0.65	0.58

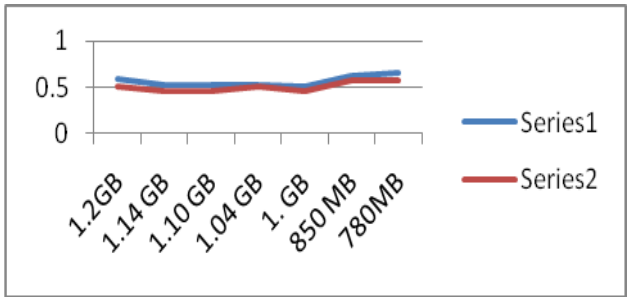

Fig 10:Performance of processors without co processors and processors with co processors in executing Graphical Benchmark program

Network Benchmark programs are also processed in the same manner and tabulated as below. Fig 11. shows the performance comparisons between a multi core processor without co processors and a multi core processor with DSP co processor in executing Graphical Benchmark and Network Benchmark Programs

Table 9: Graphical Benchmark and Network Benchmark programs executed by multi core processor and DSP Based Co Processor

Program Specification	Physical capacity of program	Time consumption in seconds	Time consumption in seconds for multi core processor with DSP Co processor
Graphical Benchmark	1.2GB	0.58	0.57
Graphical Benchmark	1.14 GB	0.52	0.52
Graphical Benchmark	1.10 GB	0.52	0.52
Graphical Benchmark	1.04 GB	0.51	0.50
Graphical Benchmark	1. GB	0.5	0.5
Graphical Benchmark	850 MB	0.62	0.62
Network Benchmark	790 MB	0.65	0.67
Network	1.05 GB	0.51	0.51

Benchmark			
Network Benchmark	1.08 GB	0.52	0.54
Network Benchmark	875 MB	0.62	0.62
Network Benchmark	805 MB	0.62	0.62

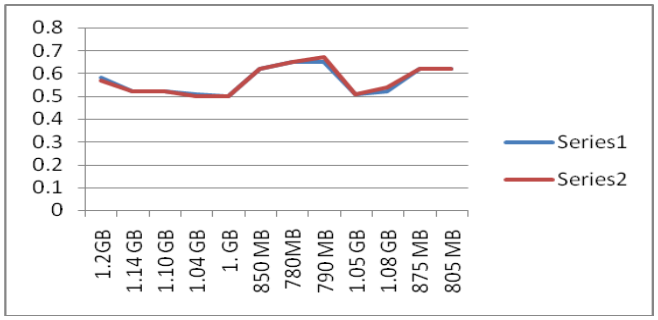

Fig 11:.Performance of the multi core processor without co processors and multi core processors with DSP Based Co Processor in executing Graphical benchmark programs and network Benchmark programs

4.2. Performance of FPGA processors

To calculate the performance of different processors with graphical benchmark programs, initially the program is made to execute in an FPGA processor and the time to execute the program is observed. It is then compared with that of the benchmarks.

Table 10: Performance of Multi core Processors and FPGA Processors in executing Graphical Benchmark programs

Program Specification	Program Size	Time consumption in seconds in a Multi core processor	Time consumption in seconds in an FPGA processor
Graphical Benchmark	1.2GB	0.54	0.59
Graphical Benchmark	1.14 GB	0.42	0.52
Graphical Benchmark	1.10 GB	0.42	0.53
Graphical Benchmark	1.04 GB	0.42	0.51
Graphical Benchmark	1. GB	0.39	0.45
Graphical Benchmark	850 MB	0.38	0.45
Graphical Benchmark	780MB	0.38	0.43

From the above values, graphs are constructed for the program size and the corresponding time consumed, while executing in a multi core processor and an FPGA respectively.

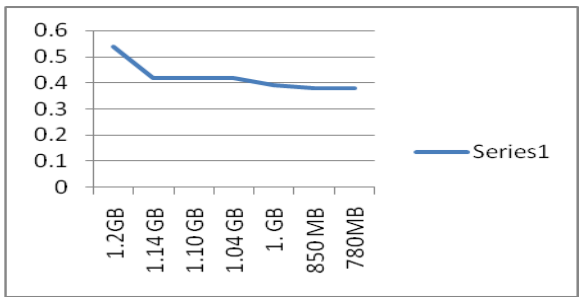

Fig 12: Performance in executing Graphical Benchmark program using multi core processor

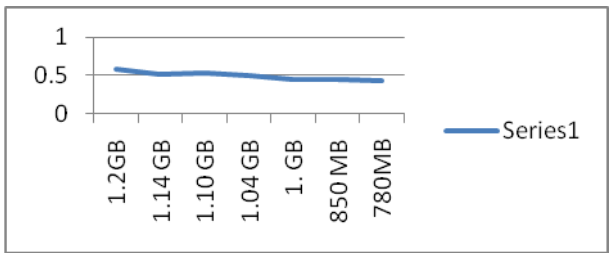

Fig 13: Performance in executing Graphical Benchmark programs using FPGA processor

From the graphs, it can be concluded that the multicore processor processes graphical data in an average time of 0.422 seconds and the Field Programmable Gate Array has consumed an average time of 0.49 seconds. Now the time line graph for both the multi core processor and FPGA is compared, to have a clear view of the processor preformance and is shown in Fig 14.

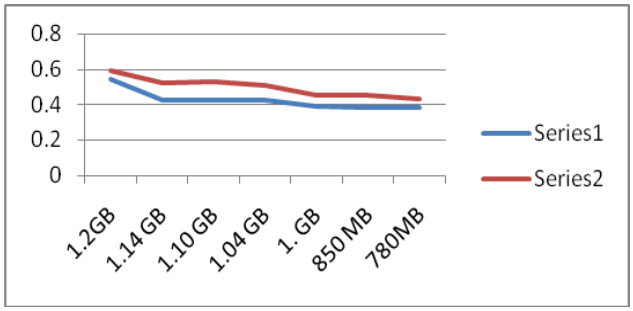

Fig 14: Performance of multi core processor and FPGA processor in executing Graphical Benchmark programs

The comparison graph gives a conclusion that for the graphical benchmark programs, the multicore processor has a better efficiency than the FPGA processor.

The same process of testing can be carried out with networking benchmark programs to analyze the performance of FPGA processor with networking Benchmark programs.

Table 11:Performance of Multi core Processor and FPGA Processor in executing Network Benchmark Programs

Program Specification	Program Size	Time consumption in seconds with Multi-core Processor	Time consumption in seconds with FPGA processor
Network Benchmark	790 MB	0.64	0.62
Network Benchmark	1.05 GB	0.52	0.47
Network Benchmark	1.08 GB	0.53	0.50
Network Benchmark	875 MB	0.60	0.57
Network Benchmark	805 MB	0.62	0.60

To analyze the performance of FPGA with network Benchmark programs, different programs are made to run in multi-core processor and FPGA processor and the time consumed to process is tabulated and its pictorial view is shown in Fig .7.10.

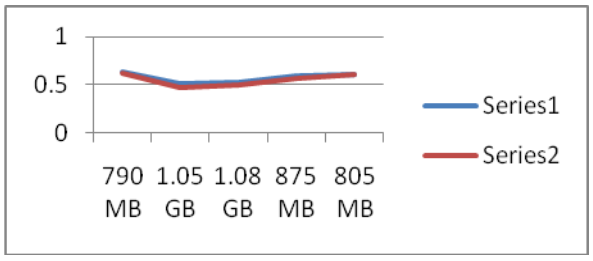

Fig 15: Performance of multi core processor and FPGA processor in executing Network Benchmark programs

Using the same procedure, both the processors has been tested for document benchmark programs and the results are tabulated and a graph is drawn to analyze the performance.

Table 12: Performance of Multi core Processor and FPGA Processor in executing Document Benchmark Programs

Program Specification	Program Size	Time consumption in seconds with Multi-core processor	Time consumption in seconds with FPGA processor
Document Bench	300 MB	0.99	0.84
Document Bench	180MB	0.95	0.81
Document Bench	150 MB	0.95	0.8
Document Bench	100 MB	0.86	0.74
Document Bench	80MB	0.85	0.74

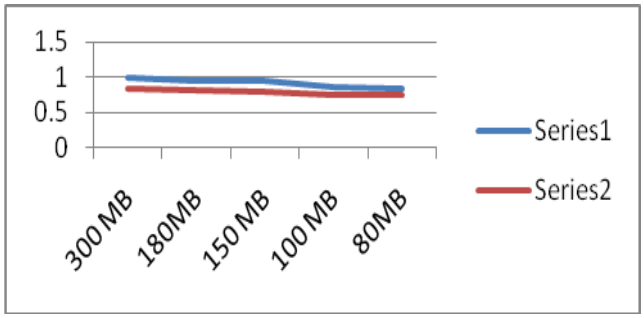

Fig 16:. Performance of multi core processor and FPGA processor in executing Document Benchmark programs

Analysing the perfomances of both the processors in executing graphical and desktop benchmark programs, the FPGA processor performs better than multi core processor in terms of time cosumption.

5. CONCLUSION

From the above results, we arrive at a conclusion that the introduction of mirror node either to a cluster of nodes or to each sensor node will result in increasing the performance of the sensor network and also the availability is increased to a maximum level. The method of introducing mirror node for each sensor node is not suitable for commercial application due to high cost factor. In systems where cost factor is an issue the later can be opted to improve the performance of the system. To overcome the difficulty that incurred due to high cost, we proposed a system, where a single node (DPA) is allocated for monitoring the remaining nodes in the cluster. If any of the monitored nodes produced anomalous data, then

the faulty node would be isolated from the network and the monitoring node will take charge performing the tasks that needs to be done by the faulty node. Thus implementing substitute node for each cluster will improve the reliability of the sensor network and makes the sensor to work more efficiently in real time systems where strict constraints are followed to make the system work correctly.

This proposed system is designed for a static sensor network. It makes use of substitution nodes to ensure continuous flow of data even in case of base node failure. It can be extended to mobile adhoc networks. Future work involves increasing the number of nodes which increases the efficiency and effectiveness of metrological data collected from distributed set of sensor nodes. Also work can be done by using a scalable network to increase the performance. Cloud computing could also be made use of, which offers many benefits, such as flexibility and instant access to the latest data and applications. But there are also risks, such as the dependency on high availability, high performance network connections, and not to forget about the least security and privacy.

6. REFERENCES

[1] E. Rohou and M. Smith, (1999), "Dynamically managing processor temperature and power," *Proc. FDDO-2*

[2] S. I. Souri, K. Banerjee, A. Mehrotra, and K. C. Saraswat,(2000) "Multiple Si layer ICs: Motivation, performance analysis, and design implications," *Proc Design Automation Conf.*, pp. 873-880.

[3] S. Das, A. Chadrakasan, (2003) "Three-dimensional integrated circuits: performance, design methodology, and CAD tools," *Proc. IEEE Annual Symp. on VLSI*,pp. 13-18.

[4] M.B. Kleiner, S. A. Kuhn, P. Ramm and W. Weber,(1995) "Thermal analysis of vertically integrated circuits." *Tech. Dig. Int'l Electron Devices Meeting*, pp.487-490.

[5] A. Rahman, R. Reif, (2001) "Thermal analysis of three dimensional (3-D) integrated circuits (ICs)," *Proc.Int'l Interconnect Technology Conf.*, pp. 157-159.

[6] F. Fallah and M. Pedram,(2005) "Standby and active leakage current control and minimization in CMOS VLSI circuits." *IEICE Trans. on Electronics*, Special Section on Low-Power LSI and Low-Power IP, Vol. E88-C, No. 4, pp. 509-519.

[7] B. Chatterjee, M. Sachdev, S. Hsu, R. Krishnamurthy, and S. Borkar, (2003) "Effectiveness and scaling trends of leakage control techniques for sub-100nm CMOS technologies," *Proc. Int'l Symp. Low-power Electronics*, pp. 122-127.

[8] M.L. Mui, K. Banerjee, A. Mehrota, (2004) "Power supply optimization in sub-130nm leakage dominant technologies," *Proc. Int'l Symp. Quality Electronic Design*, pp. 409-414.

[9] Int'l Technology Roadmap for Semiconductors (ITRS), 2004.

[10] K. Banerjee, A. Mehrotra, A. Sangiovanni- Vincentelli, and C.Hu (1999) "On Thermal Effects in Deep Sub-Micron VLSl Interconnects," *Proc. Design Automation Conf.* , pp. 885-891.

[11] H. J. M. Veendrick (1984) "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. 19, pp. 468–473.

[12] A. Alvandpour, P.L. Edefors, C. Svensson, (1998) "Separation and extraction of short-circuit power consumption in digital CMOS VLSI circuits," *Proc. Low Power Electronics and Design*, pp. 245-249.

[13] S. Turgis, N. Azemard, and D. Auvergne, (1996) "Explicit evaluation of short-circuit power dissipation and its influence on propagating delay for static CMOS gates," *Proc. IEEE Int. Symp. on Circuits and Systems*,vol. 4, pp. 751-754.

[14] S.H. Jung, J.H. Baek, S.Y. Kim, (2001) "Short circuit power estimation of static CMOS circuits," *Proc. Asia-Pacific Design Automation Conf.*, pp. 545-549.

[15] P.S.Balamurugan, K.Thanushkodi, (2009) "Array Cache An Efficient Memory For Multi Core Processor", Kyoto Journal of Engineering Research, Vol XI, 0024-609X