Towards An Optimal Multicore Processor Design for Cryptographic Algorithms – A Case Study on RSA

MUTAZ Al-TARAWNEH[1], ASHRAF ALKHRESHEH[2]

[1]Computer Engineering Department, [2]Mathematics and Computer Science

[1]Mu'tah University, [2]Tafila Technical University

[1]P.O.Box 7, Mu'tah 61710, [2]P. O. Box 179, Tafila 66110

JORDAN

[1]mutaz.altarawneh@gmail.com, [1]mutaz.altarawneh@mutah.edu.jo,[2]khresheh.ashraf@ttu.edu.jo

Abstract: - This paper aims at identifying the optimal Multicore processor configuration for cryptographic applications. The RSA encryption algorithm has been taken as a case study and a comprehensive design space exploration (DSE) has been performed to obtain the optimal processor configuration that can serve as either a standalone or a coprocessor for security applications. The DSE was based on four figures of merit that include: performance, power consumption, energy dissipation and lifetime reliability of the processor. A parallel version of the RSA algorithm has been implemented and used as an experimentation workload. Direct program execution and full-system simulation have been used to evaluate each candidate processor configuration based on the aforementioned figures of merit. Our analysis was based on commodity processors in order to come up with realistic optimal processor configuration in terms of its clock rate, number of cores, number of hardware threads, process technology and cache hierarchy. Our results indicate that the optimal Multicore processor for parallel cryptographic algorithms must have a large number of cores, a large number of hardware threads, small feature size and should support dynamic frequency scaling. The execution of our parallel RSA algorithm on the identified optimal configuration has revealed a set of observations. First, the parallel algorithm has achieved a 79% performance improvement as compared to the serial implementation of the same algorithm. Second, running the optimal configuration at the highest possible clock rate has achieved 40.13% energy saving as compared to the same configuration with the lowest clock rate. Third, running the optimal configuration at the lowest clock rate has achieved a 19.7 % power saving as compared to the same configuration with the highest clock rate. Fourth, the optimal configuration with low clock rate has achieved 109.85 % higher mean time to failure (MTTF), on average, as compared to the high-frequency configuration. Consequently, the optimal configuration has always the same number of cores, hardware threads, and process technology but the clock rate should be adjusted appropriately based on the design constraints and the system requirements.

Key-Words:- RSA, performance, power, energy, lifetime reliability, optimal configuration.

1. INTRODUCTION

Cloud computing has recently become an attractive computing infrastructure for both individual users and governmental organizations. It typically consists of networked resources and provides subscription-based services that allow users to obtain storage space and computing resources [1]. The cloud alleviates the need for users to be in the same physical location as the hardware resources they are using. Hence, its use via mobile and handheld devices has gained a widespread among computer users. Admittedly, data security is one of the major critiques against the services provided by the cloud; the ability of the cloud service providers to secure user's data is of questionable value from user's perspective. Therefore, users see an inevitable responsibility to protect their data before storing it on the cloud. This goal can be achieved by using cryptographic algorithms such that data can be encrypted before being stored, on the cloud, and decrypted when retrieved. The security level provided by any cryptographic algorithm is proportional to the amount of processing performed by that algorithm. On the other hand, energy efficiency and battery life of mobile and handheld devices have a great deal of importance for both users and hardware designers. This valuable requirement could potentially be overwhelmed by the high processing demand of cryptographic algorithms. Consequently, a new algorithm design and a careful hardware configuration should be attained in order for users to satisfy the requirement of high security and an extended battery lifetime of their mobile devices. From software perspective; cryptographic applications have a large amount of data-level parallelism. From hardware viewpoint, today's mobile devices have been equipped with Multicore processors that can execute multiple software threads simultaneously. This paper makes the case that a parallel cryptographic algorithm that runs on a Multicore processor with a particular power settings, such as clock frequency, would provide the same security level, achieve a higher performance and consume less energy as compared to a sequential version of the same algorithm running on a single core processor with the same or even higher power settings. This paper takes the RSA algorithm [2, 3, 4] as a case study and makes the following twofold contributions. First, a parallel version of the RSA algorithm has been implemented using the OpenMP parallel programming model [5]. Second, a comprehensive design space exploration has been performed to figure out the optimal Multicore processor configuration that best fits the processing requirements of cryptographic algorithms and maintains the optimum trade-off between four important figures of merit including performance, power, energy and lifetime reliability of the processor. Each processor configuration is a 3-tuple including the number of hardware threads, clock frequency and the process technology used to implement the underlying transistors. In order to obtain realistic processor configurations, the design space has been limited to those designs that resemble commercially available commodity processors. The rest of this paper is organized as follows: section 2 discusses some preliminaries, section 3 briefly summarizes the related work, section 4 describes our methodology, section 5 shows our results and analysis and section 6 summarizes and concludes this work.

2. Preliminaries

This section is an introductory framework that explains the basic operation of the RSA algorithm and the OpenMP programming model.

2.1 RSA Cryptography

The RSA algorithm follows the concept of public key cryptography (PKC) [6, 7, 8]. The PKC is a cryptographic paradigm in which two keys i.e. an encryption key and a decryption key are used to govern the encryption and decryption processes. The encryption and decryption in PKC are very compute-intensive and time-consuming processes due to the large number of mathematical operations involved. These mathematical operations include modular exponentiation and reduction techniques [9].The RSA include three main phases: Key Generation in which public and private keys are generated, Data Encryption which is used to hide a plain text in a cipher text and Data Decryption where plain text is retrieved from the cipher text.

2.1.1 Key Generation

The key generation phase is a very crucial part of the RSA algorithm. It consists of the following steps:

1. Pick two large random prime numbers *p* and *q* whose bit size is at least equal to 512 bits.
2. Find the modulus (*m*) = *p* × *q*.
3. Find *phi* = (*p-1*) × (*q-1*).
4. Pick an integer value e that falls in the range (1, phi) such that GCD (*e*, *phi*) =1 where GCD stands for the Greatest Common Divisor. In RSA, *e* is known as the public or encryption exponent.
5. Find an integer value d that falls in the range (1, phi) such that *e* × *d* = 1 mod *phi*. In RSA, *d* is referred to as the private or decryption exponent.

According to the RSA operation, the public key consists of *m* and *e* while the private key consists of *m* and *d*.

2.1.2 Data Encryption

The encryption process works according to the following equation:

$$C = M^e \bmod m \tag{1}$$

Such that:

- C is the cipher text
- M is the plain text.

Eq. (1) illustrates the basic operation of encryption in which the plain text (M) is raised to the power *e* and divided by the modulus *m*. The remainder of the division operation is taken as the cipher text (C). While a normal division operation can be used for small values of *p* and *q*, a more complex operation such as modular exponentiation and reduction [9] should be used for large numbers. The use of modular exponentiation makes Data Encryption a very time-consuming part of the RSA.

2.1.3 Data Decryption

The decryption process works according to the following equation:

$$M = C^d \bmod m \qquad (2)$$

Eq. (2) depicts the basic operation of decryption in which the cipher text (*C*) is raised to the power *d* and divided by the modulus *m*. The remainder of division yields the plain text (M). The Data Decryption phase with large values of *p* and *q* requires the use of modular exponentiation and reduction technique to perform the decryption process which means that Data encryption is also a time-consuming part of the RSA.

2.2 OpenMp

OpenMP is a portable Application Programming Interface (API) that can be used to create shared-memory parallel programs. It has recently gained a widespread among computer programmers; it allows programmers to write implicitly multithreaded applications that can efficiently utilize the ever-increasing abundance of processor cores on a Multicore processor. In general, Programs written using OpenMp depends on Thread-Level Parallelism (TLP), in which several execution threads are distributed among the available cores, to achieve high performance as compared to sequential programs which depend on Instruction-Level Parallelism (ILP) only. OpenMP consists of a group of compiler directives that tells the complier that a particular region of a program (typically the most time-consuming loops) should be divided into a number of simultaneous threads. The number of created threads in by default equal to the number of cores, however, the user can set the number of threads to any particular value via the numerous interfaces provided by the OpenMP. Moreover, OpenMP provides the user with the ability to choose the thread's scheduling policy that will be used during program execution.

3. Related Work

This section summarizes previous research efforts that directly relates to our work in both parallel algorithm implementation and optimal hardware design.

3.1 Processor Design

Optimal hardware configuration of both general and special purpose architectures is an active area of research. In [10], an exhaustive design space exploration (DSE) has been performed to identify the optimal parameters i.e. branch prediction, instruction-window size and cache size of a general purpose superscalar processor. Their work targets a single-core processor design and has focused on processor performance without taking into account other parameters such as energy and reliability. On the other hand, DSE has been widely used to figure out the optimal design of embedded or special purpose processors. In [11-19], DSE has been used to study performance/energy tradeoffs in the design of embedded/special purpose processors. However; their work differs from ours in two main aspects. First, their work targets single core processor design only without considering Multicore designs. Second, cache memory design was the only parameter of interest without taking into account other parameters such clock frequency and process technology. In [20], the impact of TLP on the performance of optimum single-core embedded processor design has been investigated. However, their work did not consider Multicore processors and did not take into account other parameters such as energy and reliability.

In [21, 22], the efficiency of the DSE process has been addressed and some mechanisms were proposed to improve its speed. In the context of Multicore processors, [23, 24, 25, 26, 27] have used DSE to study performance/energy tradeoffs of Multicore special purpose processors. However; our work differs from theirs in four main aspects. First, their analysis was cache-based only without taking into account other parameters such as clock frequency and the number of hardware threads. Second, they have focused on dynamic energy only without considering leakage energy. Third, they have overlooked the impact of technology scaling on processor energy. Fourth, their analysis was based on multiprogramming workloads while our analysis is based on a multithreaded workload. In [28], a regression-based analysis has been performed to identify the most important factors that affect the power consumption of Multicore and multithreads processor chips.

3.2 Algorithm Design

With the advent of Multicore processors, parallel algorithm implementation has become an appealing programming paradigm due to its performance advantages over sequential implementations. However, only few research efforts have focused on cryptographic algorithms. In [29], the parallelization of the AES algorithm has been presented and its performance has been evaluated. However, their study was performance-centric without taking into account the energy and reliability implications of the algorithm on Multicore processor design .On the other hand, [30] has studied the performance advantages of parallel RSA implementation. However, they did not take into account other parameters such as power consumption and processor reliability. Moreover, they did not shed light on which hardware configuration was used to run the algorithm. Hence, their work did not focus on workload-architecture interactions. Moreover, [31] has investigated an FPGA-based implementation of the RSA. While FPGA-based implementations can provide comparable performance levels as Multicore processors, its power consumption is very high that it cannot be employed in battery-operated devices such as mobile and handheld devices. In summary, our work represents a comprehensive study in which a novel parallel algorithm design has been presented and its performance, energy and lifetime reliability implications have been investigated based on some figures of merit that captures the current trends in processor design.

4. Methodology

The experimental work conducted towards this research consists of three main steps including: RSA parallelization and performance evaluation, power consumption and energy dissipation estimation and lifetime reliability analysis. Each step will be thoroughly explained in the following sub-sections.

4.1 Automatic parallelization of RSA

This step was mainly inspired by Amdahl's law which is commonly used to measure the overall speedup that can be achieved by parallelizing a particular portion of an algorithm [32]. Amdahl's law is given by equation 3.

$$Speedup = \frac{1}{f + (1 - f)/n} \tag{3}$$

In Eq. 3, f is the time spent executing the serial portion of the parallelized algorithm and n is the number of processor cores. The first corollary of Amdahl's law states that: decreasing the serialized portion of an algorithm by increasing the parallelized portion is of greater importance than adding more processor cores [33]. For example, if 30-percent of an algorithm can be parallelized on a dual-core system, doubling the number of processor cores reduces the execution time from 85% of the serial time to 77.5%, whereas doubling the amount of parallelized code reduces the execution time from 85% to 70%. Consequently, implementing a parallel version of RSA has been adopted as an extremely important part of this work. Given a number of messages to be encrypted (NUM_MSGS), the RSA encryption algorithm is straight forward and can be written as shown in Fig.1. Where MSGS and EMSGS are the arrays in which plain and cipher messages are stored respectively. The RSA main loop does not have any true data dependences. On the other hand, anti and output dependences can be avoided by the privatization of loop induction variables.

```
Algorithm 4.1.1 : Serial RSA(SeRSA)
Input : NUM_MSGS;
Input : MSGS;
Output : EMSGS;
Begin
   for i ← 0 to NUM_MESGS
   do {EMSGS[i] ← (MSGS[i] ^ e ) mod m;}
End
```

Fig.1: Serial RSA (SeRSA) pseudo-code.

The OpenMP parallelization technique was based on the following two steps:

i. Creating a parallel region using the ***#pragma*** directive. The appropriate clauses (i.e. shared and private clauses) should also be set accordingly.

ii. Using the ***#pragma omp for*** directive to specify the loop whose iterations should be executed in parallel.

The new automatically parallelized RSA algorithm is given by pseudo-code depicted by Fig.2.

```
Algorithm 4.1.2 : Parallel RSA(PaRSA)
Input : NUM_MSGS;
Input : MSGS;
Output : EMSGS;
Begin
 # pragma omp parallel shared (EMSGS,MSGS) private(i)
        {
          # pragma omp for
           {
             for i ← 0 to NUM_MSGS - 1
             do {EMSGS[i] = (MSGS[i] ^ e ) mod m;}
           }
        }
End
```

Fig.2: Parallel RSA (PaRSA) pseudo-code.

The same parallelization procedure can be applied on the decryption part of the RSA. For the rest of this paper, the serial and parallel versions of the RSA will be denoted as SeRSA and PaRSA respectively. Both SeRSA and PaRSA have been implemented using the C++ programming language and compiled using the GCC compiler on Fedora 19 [34]. They have been run on four different Intel Multicore processors [35]. Each processor configuration is defined in terms of four different parameters including: number of cores, number of threads per core, clock frequency and on-chip cache hierarchy. Processor configurations are summarized in table 1. In case of PaRSA, OpenMP allows the programmer to set the number of threads that the complier can create. From performance perspective, the number of created threads should be less than or equal to the number of hardware threads per processor chip. Table 2 illustrates the number of threads supported by each processor. A PaRSA with 1 thread is equivalent to SeRSA. The used processors (Table 1) support CPU frequency scaling. Frequency scaling can be done either dynamically or manually. Dynamic frequency scaling is done by the Operating System based on workload demands while manual scaling can be done by userspace applications based on performance or power constraints. In this paper, CPU frequency scaling has been done manually using the *CPUFreqUtils* package provided by the Linux operating system [36].For both SeRSA and PaRSA, several working set sizes i.e. NUM_MSGS have been tested. The working set include {100, 200, 300, 400 and 500} million messages. For each possible working set size all possible combinations of the 2-tuple (no. of threads, clock frequency) have been experienced. In case of SeRSA, the no. of threads was always set to 1. Therefore, each experiment in this part represents one possible combination of the 3-tuple (working set size, no. of threads, clock frequency).

Table 1: Processor Configurations.

Processor	**Cores per Processor**	**Threads per Core**	**Available Frequencies in GHz**
Core2Duo	2	1	0.8,1.6,2.13,2.8
Core i3	2	2	1.6, 2.1, 2.7, 3.3
Core i5	4	1	1.2,1.5,1.8,2.1,2.4
Core i7	4	2	1.2,1.5,1.8,2.1,2.4

Cache Hierarchy			
	Level-1	**Level-2**	**Level-3**
Core2Duo	IL1: 32KB DL1:32 KB	6 MB shared	N/A
Core i3	IL1: 32KB DL1:32 KB	256 KB Private	3 MB Shared
Core i5	IL1: 32KB DL1:32 KB	256 KB Private	3 MB Shared
Core i7	IL1: 32KB DL1:32 KB	256 KB Private	3 MB Shared

Table 2: Possible number of Threads per Processor.

Processor	**Supported no. of Threads**
Core 2 Duo	1,2
Core i3	1,2,4
Core i5	1,2,4
Core i7	1,2,4,8

4.2 Power Consumption Estimation.

Power consumption is an extremely important parameter in the design space of modern processors. Fig.3 illustrates our power analysis framework. McPAT [37] has been used to obtain the power consumption of the RSA on the processor configurations shown in Table 1. McPAT is an integrated power, area and timing modelling framework that supports extensive design space exploration for Multicore processors ranging from 90nm to 22 nm and beyond. Power readings include both dynamic and leakage power. It models power, area and timing for the device types forecast in the ITRS roadmap [38].

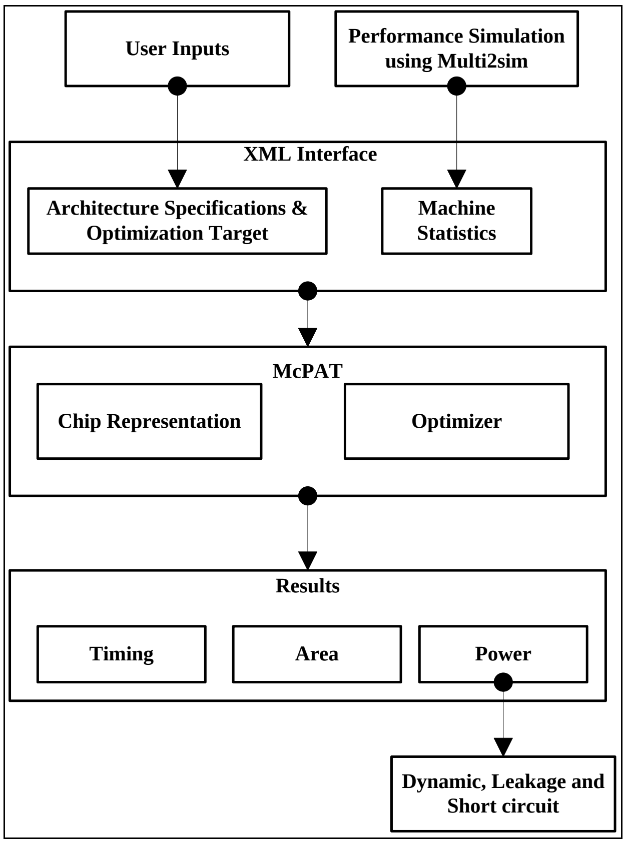
User Inputs
Performance Simulation using Multi2sim
XML Interface
Architecture Specifications & Optimization Target
Machine Statistics
McPAT
Chip Representation
Optimizer
Results
Timing
Area
Power
Dynamic, Leakage and Short circuit

Fig.3: Power Analysis Framework

McPAT has a flexible XML-based interface that allows it to accept both user inputs and performance simulator's statistics. McPAT uses user inputs and performance simulator's statistics in order to report power, area and timing of the modelled processor architecture. All these inputs can be passed via the XML interface. User inputs include architecture parameters, circuit's parameters, technology parameters and optimization target. Simulation statistics include hardware utilization, activity factors and the usage of power management techniques such as P- and C- states. On the other hand, multi2sim [39] has been used to obtain the necessary resource utilization information that McPAT requires in order to estimate the total power consumption of the processor. Multi2sim is a full-system simulation environment that allows hardware designers to simulate superscalar, multithreaded and Multicore processor based on the x86 instruction set architecture (ISA). The RSA has been simulated on multi2sim using processor models the closely resembles the configurations shown in Table 1. Multi2sim has been chosen since it supports the X86 ISA used by the processor configuration in Table 1. Therefore, the same binary can be used for both direct execution (section 4.1) and performance simulation without any modifications which allows hardware designer to obtain identical results from the two evaluation methods i.e. direct execution and performance simulation. The power consumption of each processor configuration (Table 1) has been estimated at all possible clock frequencies and process technologies.

4.3 Lifetime Reliability Analysis

Admittedly, integrating a large number of transistors on a single chip and running them on a very high clock rates have resulted into an incredible power density levels. Increasing power densities has a negative effect on the long-term life-time reliability of the processor [40-42].

Therefore, designing efficient processor chips should take into account lifetime reliability besides other metrics such as performance and power consumption. In this work, the RAMP model [40-42] has been used to estimate the lifetime reliability of processor chips based on the RSA workload. RAMP is an analytical model that provides reliability estimates for a workload running on a processor chip implemented using a specific process technology. In order to choose an appropriate reliability metric, the proposed processor architecture is assumed to work according to a particular Service Level Agreement (SLA) [43]. With respect to an SLA, the processor alternates between two states of service:

a. Service accomplishment in which the service is delivered as specified.
b. Service interruption in which the delivered service is different from the SLA.

Failures cause transitions between these two states (from state 1 to 2) while restorations leads to transition from state 2 to 1. This work takes into account irreparable faults only; only failures are taken into account. In other words, the used reliability metric should take failures into account while overlooking system restorations. Therefore, system reliability becomes a measure of the continuous service accomplishment (or equivalently, of the time to failure) from a reference initial instant. Thus, the mean time to failure (MTTF) is an appropriate reliability measure.

RAMP estimates long-term processor MTTF of a given workload based on four failure mechanisms including electromigration (EM), stress migration (SM), time-dependent dielectric breakdown (TDDB) and thermal cycling (TC). It provides a micro-architectural structure level implementation (e.g. caches, ALUs, branch predictor, etc) of the failure mechanisms for a particular technology generation. The following paragraphs summarize each of these failure mechanisms assuming that the processor operates in the steady state at a fixed operating point.

Electromigration (EM) is a well-understood failure mechanism in the field of material science and semiconductor technology. It appears in processor interconnects and is caused by the mass transport of the conductor metal atoms in those interconnects [44, 45]. Two destructive scenarios can be caused by EM. First, increased resistance and open circuits appear at the sites of metal atoms depletion. Second, extrusions can form at the sites of metal atom pile up which leads to shorts between adjacent metal lines [44, 45].

The EM-induced MTTF, $\mathrm{MTTF}_{\mathrm{EM}}$, is given by the following formula [39-41]:

$$\mathrm{MTTF}_{\mathrm{EM}} = (J)^{-n} e^{\frac{E_{a_{EM}}}{kT}} \tag{4}$$

Such that J is the current density in the interconnects of the processor, $E_{a_{EM}}$ is EM's activation energy, k is Boltzmann's constant and T is the processor temperature in Kelvin. $E_{a_{EM}}$ and n are interconnect-dependent constants. For the copper interconnects modelled in RAMP [40-42], the values of 0.9 and 1.1 have been used respectively.

Stress Migration (SM) is a failure mechanism in which the metal atoms in the processor interconnects migrate due to mechanical stress. SM is mainly caused by thermo-mechanical stresses which are caused by differing thermal expansion rates of different materials in the device [44].

The SM-induced MTTF, $\mathrm{MTTF}_{\mathrm{SM}}$, is given by the following formula [40-42, 44]:

$$MTTF_{SM} = (T - T_0)^{-m} e^{\frac{E_{a_{SM}}}{kT}} \tag{5}$$

Where T_0 is the stress-free temperature in Kelvin, T is the absolute processor temperature in Kelvin, m and E_{aSM} are material-dependent constants and have been set to 2.5 and 0.9 for the copper interconnects modelled in RAMP [40-42].

Time-Dependent Dielectric Breakdown (TDDB) is another failure mechanism that can be encountered in the semiconductor devices. It is also known as gate oxide breakdown. The dielectric (or the gate oxide) breaks down with time and fails when a conductive path is formed in the dielectric [40, 46]. The TDDB-induced MTTF modelled by RAMP [40-42] is based on the experimental work performed by Wu et al. at IBM [47] and is given by the following formula:

$$MTTF_{TDDB} = \left(\frac{1}{V}\right)^{a-bT} e^{\frac{x+\frac{y}{T}+zT}{kt}} \tag{6}$$

Such that T is the absolute processor temperature, V is the supply voltage, a, b, x, y and z are fitting parameters. Table 3 shows the fitting parameters used by RAMP Based on the experimental data published in [47].

Table 3: RAMP TDDB parameters [47]

Parameter	Value
A	78
B	-0.081
X	0.759 *ev*
Y	-66.8 *evk*
z	-8.37 x 10^{-4} *ev/k*

Thermal Cycling (TC) is a well-known failure mechanism in the semiconductor industry. It is mainly caused by thermal cycling in the processor.

Thermal cycles can be classified into two main types: large cycles which occur at a low frequency (powering up and down) and small cycles which happen at a much higher frequency due to changes in application behaviour. RAMP [40-42] takes into account large thermal cycles only due to the lack of validated models that captures the impact of small thermal cycles on device reliability [44]. The TC-induced MTTF modelled in RAMP [40-42] is based on the Coffin-Manson equation [44] and is given by the following formula:

$$MTTF_{TC} = \left(\frac{1}{T_{average} - T_{ambient}}\right)^{q} \qquad (7)$$

Where $T_{ambient}$ is the ambient temperature and ($T_{average}$ - $T_{ambient}$) is the average large thermal cycle a processor chip encounters during workload execution and q is the Coffin-Manson exponent which is a material-dependent, empirically determined constant. RAMP uses a q of 2.35 [40-42].

As shown in the formulas given by equations 4-7, the extent to which a particular workload can influence the lifetime reliability of the processor depends on the amount of heat dissipated during program execution. In other words, since the switching activity and the total power consumed by the processor depends on the running workload, the actual operating temperature and current densities in the processor interconnects also depends on the running workload [42]. Consequently, in order estimate the impact of the RSA workload on the MTTF of the processors shown in Table 1, processor temperature under the RSA workload should be determined. The flow chart shown in Fig.4 summarizes our reliability analysis procedure. As shown in Fig.4, clock frequency was the only parameter of interest during our reliability analysis. All frequencies supported by a particular processor configuration have been tested and its corresponding temperature and MTTF values have been observed. The reliability analysis consists of five main steps. First, the clock frequency of the processor is changed using the CPUFreqUtils. Second, the RSA workload is run using appropriate input parameters that can aggressively stress the processor chip. Third, the processor temperature is observed using the psensor utility [48]. Psensor is a graphical hardware temperature monitor for Linux. It provides temperature measurements by accessing the sensors installed on the processor chip. Fourth, processor temperature, obtained by psensor, is used as inputs for the RAMP model. Fifth, the MTTF values from the RAMP model are used to judge the RSA impact on the lifetime reliability of the processor taking into account the four different failure mechanisms modelled by RAMP [40-42].

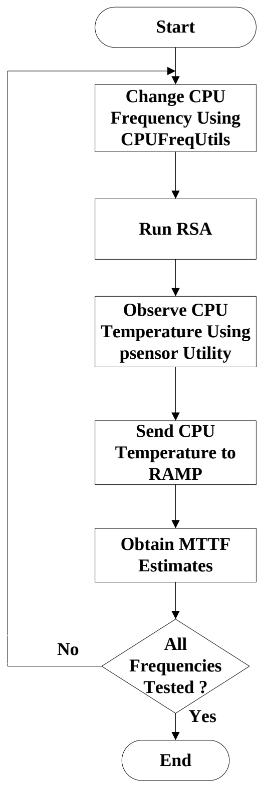
Start
Change CPU Frequency Using CPUFreqUtils
Run RSA
Observe CPU Temperature Using psensor Utility
Send CPU Temperature to RAMP
Obtain MTTF Estimates
All Frequencies Tested ?
No
Yes
End

Fig.4: Reliability Analysis Flow Chart

5. Results and Analysis

This section summarizes the results that have been obtained based on the research methodology outlined in the previous section. Each subsection provides a comparison between the processor configurations shown in Table 1 in terms of one particular parameter or figure of merit. The optimal processor configuration is determined based on its performance, power consumption, energy dissipation and lifetime reliability. In other words, the processor that achieves the best tradeoffs between the aforementioned parameters is said to be the optimal processor configuration for the cryptographic algorithms.

5.1 Performance Analysis

In this part, an extensive performance-centric analysis has been performed in order to study the impact of software parallelization on RSA's performance and identify the best processor configuration to run such a parallel workload. The best processor configuration is the one on which the RSA achieves its highest performance as compared to other processor configurations. Two main steps have to be performed as a prerequisite to the aforementioned goals:

a. *Workload selection:* the goal of this step is to pick an appropriate working-set size for performance evaluation.
b. *Scheduling policy section:* this step aims at identifying the best threads scheduling policy that minimizes contention between threads and leads to the highest possible performance using OpenMP.

Fig.5 depicts the impact of working-set size (i.e. the number of RSA input messages) on the performance of the parallelized RSA. It shows the speedup that parallel RSA (PaRSA) can achieve over serial RSA (SeRSA) at all possible working-set sizes. PaRSA-*n* indicates a PaRSA running with *n* threads. The values on the x-axis indicate the working-set size while the y-axis shows the speedup the PaRSA-*n* can achieve as compared to SeRSA. It can be observed that PaRSA-*n* almost always achieves the same speedup, as compared to SeRSA, at all working-set sizes. In other words, it can be concluded that the working-set size has a negligible impact on PaRSA's performance. Therefore, one particular working-set size can be used for the sake of performance evaluation. For the rest of this work, a working-set size of 200 million messages will be used. The other intermediate step was to evaluate the impact of different scheduling techniques on the performance of the PaRSA. OpenMP provides three main scheduling schemes: *static*, *dynamic* and *guided* [33]. *Static scheduling* partitions the loop iterations into equal-sized chunks or as nearly equal as possible in the case where the number of loop iterations is not evenly divisible by the number of threads multiplied by the chunk size. When chunk size is not specified, the iterations are divided as evenly as possible, with one chunk per thread [33]. *Dynamic scheduling* uses an internal work queue to give a chunk-sized block of loop iterations to each thread as it becomes available. When a thread is finished with its current block, it retrieves the next block of loop iterations from the top of the work queue. By default, chunk size is set to 1. It has extra overhead as compared to other scheduling schemes [33]. *Guided scheduling* is similar in spirit to dynamic scheduling, but the chunk size starts off large and shrinks in an effort to reduce the amount of time threads have to go to the work queue to get more work. When guided scheduling is used, the optional chunk parameter specifies the minimum size chunk to use, which, by default is 1 [33].

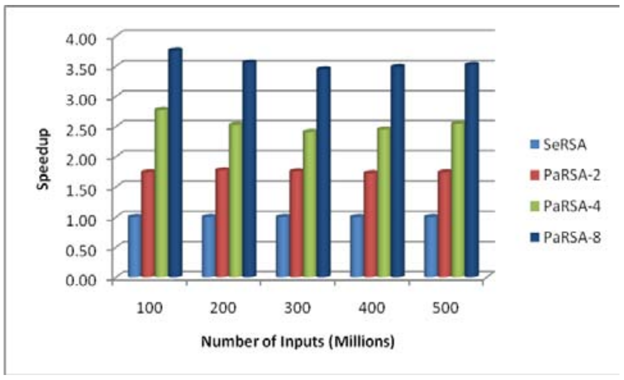

Fig.5: Impact of working-set size on performance.

In this work, the three scheduling schemes have been evaluated on PaRSA with eight threads (PaRSA-8). PaRSA-8 has been run on a data-set size of 200 million messages. Fig.6 shows the execution time of PRSA-8 with different scheduling techniques as the chunk size increases from 1 up to 100. As illustrated in Fig.6, PaRSA-8 with guided scheduling and PaRSA-8 with static scheduling have maintained an almost constant execution time regardless of the chunk size. On the other hand, the execution time of PARSA-8 with dynamic scheduling was higher than that of other scheduling techniques when the chunk size is less than 20; as the chunk size decreases, the overhead associated with retrieving work from the shared work queue will increase which in turn can degrade the performance of PARSA-8 with dynamic scheduling. It can be observed from Fig.6 that the guided scheduling policy always outperforms the other scheduling policies due to its minimal thread contention. Consequently, the guided scheduling policy has been used throughout this paper. Having obtained the optimal OpenMp settings, both SeRSA and PaRSA have been directly executed on the processor configurations shown in Table 1 and their performance has been observed at different clock frequencies and thread numbers. Fig.7 shows the execution time of SeRSA and PaRSA on Intel Core2Duo [35]. Intel Core2Duo supports two hardware threads at a time; therefore, only PaRSA-2 has been evaluated on this processor. It supports four different frequency levels that include 800, 1600, 2130 and 2800 MHz. Both SeRSA and PaRSA have been executed at all frequency levels and their execution time has been observed.

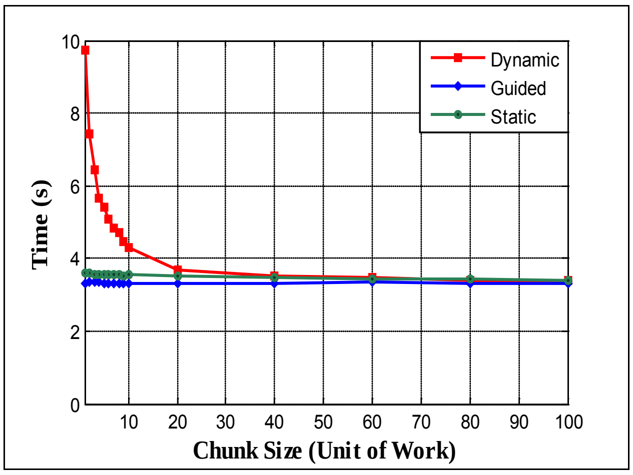

Fig.6: the impact of OpenMP scheduling policy on performance.

As depicted in Fig.7, increasing the clock frequency can always reduce the execution time of SeRSA while increasing the number of threads and/or clock frequency can always reduce the execution time of PaRSA-2; both SeRSA and PaRSA-2 have achieved their optimal performance at 2800 MHz. PaRSA-2 achieves approximately 47% performance improvement, as compared to SeRSA, at all clock frequencies. On the other hand, doubling the clock frequency (from 800 MHz to 1600 MHz) has achieved a 50 % performance improvement for both SeRSA and PaRSA-2. In other words, doubling the clock frequency on Intel Core2Duo can yield more performance improvement than doubling the number of threads. However, doubling the number of threads at low clock frequencies is more beneficial from power consumption perspective as will be shown in the next subsection. Fig.8 illustrates the performance results of SeRSA and PaRSA on Intel Core i3 [35]. Intel Core i3 has two cores with each core supporting two hardware threads. In other words, it supports four concurrent software threads. Consequently, both PaRSA-2 and PaRSA-4 have been evaluated on this processor. It can run at 1600, 2100, 2700 and 3300 MHz. Several observations can be made based on Fig.8. First, the execution time of both SeRSA and PaRSA is inversely proportional to the clock rate of the processor. Second, the execution time of PaRSA decreases as the number of threads increases at all clock rates.

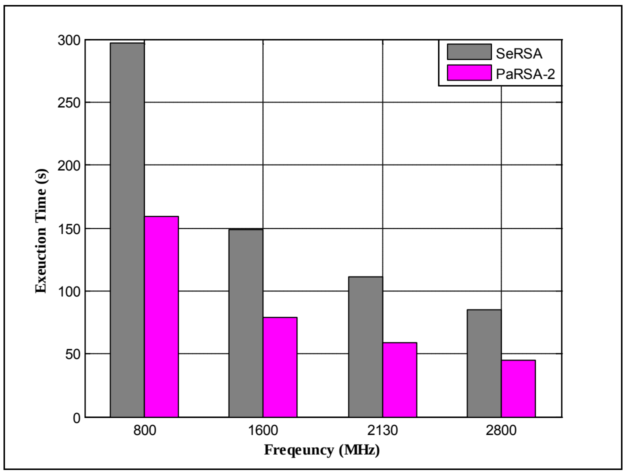

Fig.7: Performance of SeRSA and PaRSA-2 on Intel Core2Duo.

Third, PaRSA-2 and PaRSA-4 achieve a 47 % and 65 % performance improvement, as compared to SeRSA, respectively. Fourth, for SeRSA, PaRSA-2 and PaRSA-4, increasing the clock frequency from 1600 MHz to 3300 MHz has resulted in 52 % approximately. In other word, as the number of threads beyond 2 can lead to more performance improvement that increasing the clock frequency of the processor.

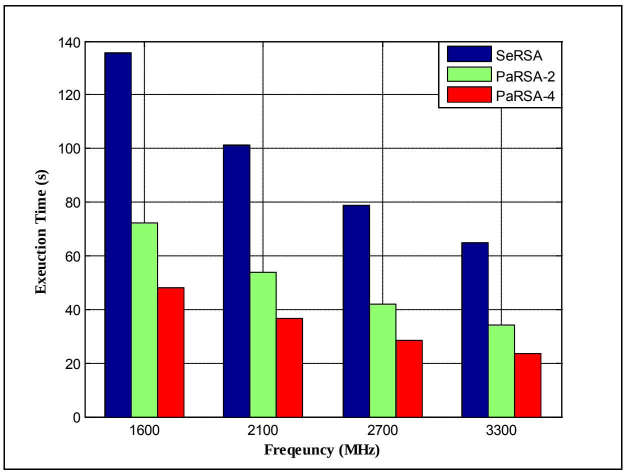

Fig.8: Performance of SeRSA and PaRSA on Intel Core i3.

Fig.9 shows the performance result of SeRSA and PaRSA on Intel Core i5 [35]. Intel Core i5 has four cores with each core supporting a single thread leading to a processor chip that can support four simultaneous software threads; only PaRSA-2 and PaRSA-4 have been studied on this processor. It can run at 1200, 1500, 1800, 2100 and 2400 MHz. The observations that can be drawn from Fig.9 are almost identical to that of Fig.8. However, by comparing the two processor at the 2100 clock rate (i.e. the common clock rate), it can be observed that Intel Core i5 slightly outperforms Intel Core i3 when running PaRSA-2 and PaRSA-4; this due to the fact that Intel Core i5 has four different cores with each core supporting a single running thread in contrast to Intel Core i3 where each pair of threads runs on the same core.

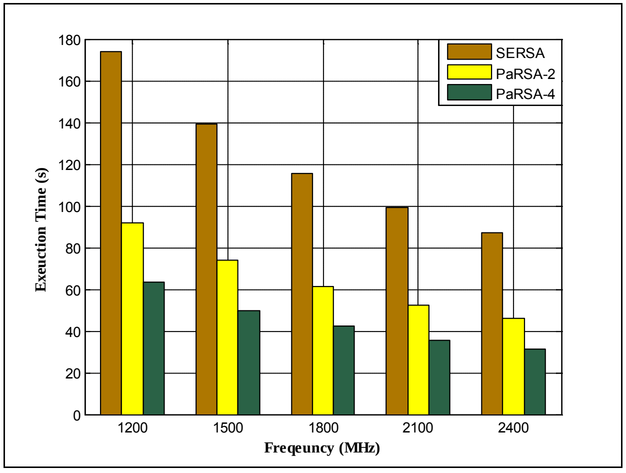

Fig.9: Performance of SeRSA and PaRSA on Intel Core i5.

In other words, when PaRSA runs on Intel Core i5, the running threads will encounter minimal inter-thread contention which results in a lower execution time as compared to Intel Core i3. Fig.10 summarizes the results of running SeRSA and PaRSA on Intel Core i7 [35]. Intel Core i7 consists of four cores with each core supporting two hardware threads which results in a processor chip that can support up to 8 simultaneous software threads. Consequently, PaRSA-8 has been evaluated on this processor in addition to PaRSA-2 and PaRSA-4. Similar to Intel Core i5, Intel Core i7 can run at 1200, 1500, 1800, 2100 and 2400 MHz. The results shown in Fig.10 confirm the results that have been depicted in Fig.8 and Fig.9 in all aspects. The most important observation that can be made is that increasing the number of threads can be more advantageous that increasing the clock frequency from performance perspective. Whereas increasing the clock frequency from 1200 to 2400 MHz leads to almost 50% performance improvement for SeRSA, PaRSA-2, PaRSA-4 and PaRSA-8, increasing the number of threads from 1 to 8 yields a 79% performance improvement at all clock rates.

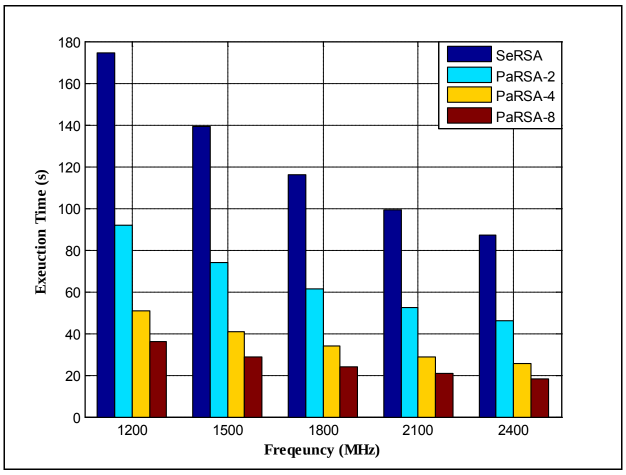

Fig.10: Performance of SeRSA and PaRSA on Intel Core i7.

In summary, the performance of RSA can be improved by either increasing the clock frequency for the same number of threads or increasing the number of threads for the same clock rate. However, the results shown in this section illustrates that increasing the number of threads can achieve more performance improvement than increasing the clock rate. In addition, our results indicate that, from performance perspective, the best processor configuration for cryptographic algorithms is a Multicore processor with a large number of hardware threads and a high clock frequency. However, this observation should be further investigated from power consumption and reliability perspectives as will be shown in the next subsections.

5.2 Power Consumption Analysis

This section shows the power consumption of PaRSA-*n* on the processor configurations shown in Table 1. For each configuration, *n* was set to the number of hardware threads supported by that processor. Both dynamic and leakage powers have been obtained for each processor configuration. The leakage power consists of gate leakage and sub-threshold leakage [37].The ultimate goal of this section is to determine the most power-efficient Multicore processor configuration for the RSA. Our power analysis was based on processor's clock frequency and its underlying process technology. Table 4 summarizes the power consumption of PaRSA-2 on Intel Core2Duo at all clock frequencies and process technologies. P_D is the dynamic power while P_L is the leakage power which consists of gate leakage (P_{LG}) and sub-threshold leakage (P_{LST}). P_{LST_PG} is the sub-threshold leakage when power gating is employed. It also shows the processor's chip area at each process technology. Several observations can be made based on the results shown in Table 4. First, as the process technology scales down, the total processor area decreases by around 50%. This can be explained by Dennard scaling [49]. According to Dennard scaling, transistor's dimensions decreases by around 30% as process technology scales down. Therefore, the total area occupied by a single transistor and in turn the processor chip decreases by 50% with respect to the previous process technology. Second, increasing the clock frequency leads to an equal-size increase in the dynamic power consumption of the processor under all process technologies. This can be explained by the following equation which shows how the power consumption of a CMOS can be calculated [50]:

$$P_D = \alpha f C V^2 \tag{8}$$

Where P_D is the dynamic power consumption, α is the activity factor, f is the processor's clock frequency, C is the load capacitance and V is the supply voltage. Eq. 8 indicates a direct proportionality between the clock frequency and the dynamic power consumption of the transistor and, in turn, the whole processor chip. Third, the total dynamic power of a processor decreases as the technology scales down. This can be explained by the scaling theory stated in [49]. According to [49], the load capacitance and the supply voltage decrease as the process technology scales leading to a power saving that approaches 50% as compared to the previous process technology. Fourth, the leakage power constitutes a significant portion of the total power consumption at lower process technologies especially when the processor runs at low clock rates. For example, when the Intel Core2Duo is implemented using the 22nm process technology and runs at 800 MHz, the leakage power constitutes approximately 74% of the total power consumption of the processor. This fact necessitates the use of power gating [37] in order to minimize the sub-threshold leakage which forms the majority of the leakage power of the processor as the technology scales down. In the previous example, the use of power gating reduces the leakage power by 47% and the total power by 35%.Based on Table 4, it can be observed that PaRSA-2 achieved its lowest power consumption when Intel Core2Duo is implemented using the 22 nm process technology and runs at 800 MHz. In other words, the most power-efficient dual-core processor for PaRSA-2 is a processor implemented at low process technology and runs at a low clock frequency. By considering the Intel Core2Duo (90nm, 800 MHz) as a reference case, the power-efficient configuration can achieve 75% power saving approximately. On the other hand, the power-efficient configuration can achieve a 91% power saving as compared to the Intel Core2Duo (90 nm, 2800 MHz).

On the other hand, Table 5 depicts the power consumption of PaRSA-4 on Intel Core i3 processor. It also shows the total processor area at each process generation. The observations that can be made from Table 5 are fourfold. First, the processor becomes more area-efficient as the process technology scales down. This can be explained by the scaling theory provided by [49]. Second, there is a significant increase in the total dynamic power as the clock frequency increases. The aforementioned observation can be explained by Eq. 8. Third, the dynamic power of the processor decreases as the process technology shrinks due to the same reasons provided for the Intel Core2Duo processor [49].Fourth, leakage power becomes the major source of power consumption as we move towards lower feature sizes. However, its effect can be minimized by using power gating techniques [37]. In summary, the most power-efficient Intel Core i3 configuration for PaRSA-4 is the configuration that is implemented using the 22 nm process technology, running at 1600 MHz and is using the power gating techniques to minimize leakage power consumption. Taking the Intel Core i3 (90 nm, 1600 MHz) as the base case, the power-efficient configuration i.e. Intel core i3 (22 nm, 1600 MHz) can achieve 80% power savings. On the other hand, the power-efficient configuration can achieve around 92.7% saving as compared to Intel core i3 (90 nm, 3300 MHz). Similarly, Table 6 summarizes the power consumption of PaRSA-4 on Intel core i5 processor. It also shows the total processor area under all process technologies. Several observations can be made based on the results shown in Table 6. First, Intel Core i5 becomes more area-efficient as the feature size thinks. This point can be explained by the scaling theory provided by [49]. Second, the dynamic power consumption of the processor is directly proportional to the clock frequency of the processor under all process technologies. This point can be directly noticed from Eq. 8. Third, the processor consumed less dynamic power as its feature size scales down. This point agrees with the scaling theory in [49]. Fourth, when the processor is implemented at small feature sizes, leakage power becomes the major source of power consumption which necessitates the use of leakage-aware design methodologies such power gating [37].

Table 4: Power Consumption of PaRSA-2 on Intel Core2Duo.

Process Technology: 90 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
800	338.68	27.6	11.78	0.86	10.91	5.62
1600	338.68	54.59	11.78	0.86	10.91	5.62
2130	338.68	72.47	11.78	0.86	10.91	5.62
2800	338.68	95.07	11.78	0.86	10.91	5.62
Process Technology: 65 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
800	186.87	15.63	14.66	1.34	13.32	6.89
1600	186.87	30.75	14.66	1.34	13.32	6.89
2130	186.87	40.76	14.66	1.34	13.32	6.89
2800	186.87	53.42	14.66	1.34	13.32	6.89
Process Technology: 45 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
800	90.96	8.23	12.24	0.67	11.57	6.03
1600	90.96	16.04	12.24	0.67	11.57	6.03
2130	90.96	20.21	12.24	0.67	11.57	6.03
2800	90.96	27.75	12.24	0.67	11.57	6.03
Process Technology: 32 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
800	59.36	4.65	15.8	0.78	15.02	7.81
1600	59.36	8.97	15.8	0.78	15.02	7.81
2130	59.36	11.84	15.8	0.78	15.02	7.81
2800	59.36	15.46	15.8	0.78	15.02	7.81
Process Technology: 22 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
800	29.45	2.56	7.28	0.01	7.27	3.82
1600	29.45	4.84	7.28	0.01	7.27	3.82
2130	29.45	6.36	7.28	0.01	7.27	3.82
2800	29.45	8.27	7.28	0.01	7.27	3.82

Table 5: Power Consumption of PaRSA-4 on Intel Core i3.

Process Technology: 90 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**$P_{LST\ PG}$**
1600	302.22	48.6	10.99	0.9	10.09	5.01
2100	302.22	63.73	10.99	0.9	10.09	5.01
2700	302.22	81.76	10.99	0.9	10.09	5.01
3300	302.22	99.79	10.99	0.9	10.09	5.01
Process Technology: 65 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**$P_{LST\ PG}$**
1600	163.03	26.67	14.84	1.4	13.44	6.71
2100	163.03	34.85	14.84	1.4	13.44	6.71
2700	163.03	44.66	14.84	1.4	13.44	6.71
3300	163.03	54.47	14.84	1.4	13.44	6.71
Process Technology: 45 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**$P_{LST\ PG}$**
1600	81.24	14.29	12.38	0.7	11.68	5.86
2100	81.24	18.63	12.38	0.7	11.68	5.86
2700	81.24	23.83	12.38	0.7	11.68	5.86
3300	81.24	29.03	12.38	0.7	11.68	5.86
Process Technology: 32 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**$P_{LST\ PG}$**
1600	44.49	7.89	15.5	0.81	14.69	7.41
2100	44.49	10.25	15.5	0.81	14.69	7.41
2700	44.49	13.08	15.5	0.81	14.69	7.41
3300	44.49	15.92	15.5	0.81	14.69	7.41
Process Technology: 22 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**$P_{LST\ PG}$**
1600	23.11	4.23	7.08	0.01	7.07	3.59
2100	23.11	5.47	7.08	0.01	7.07	3.59
2700	23.11	6.96	7.08	0.01	7.07	3.59
3300	23.11	8.44	7.08	0.01	7.07	3.59

Based on Table 6, it can be concluded that the most power-efficient Intel Core i5 configuration for PaRSA-4 is the configuration that is implemented at 22 nm feature size and runs at 1200 MHz i.e. Intel Core i5 (22 nm, 1200 MHz).Moreover, power gating techniques should be used to minimize leakage power contributions. In other words, the most power-efficient quad-core processor for the RSA should be implemented using small feature size and run at low clock frequency.

Table 6: Power consumption of PaRSA-4 on Intel Core i5.

Process Technology: 90 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	460.754	35.57	18.3	1.69	16.61	8.08
1500	460.754	44.31	18.3	1.69	16.61	8.08
1800	460.754	53.05	18.3	1.69	16.61	8.08
2100	460.754	61.79	18.3	1.69	16.61	8.08
2400	460.754	70.53	18.3	1.69	16.61	8.08
Process Technology: 65 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	253.17	20.04	26.1	2.59	23.51	11.51
1500	253.17	24.93	26.1	2.59	23.51	11.51
1800	253.17	29.81	26.1	2.59	23.51	11.51
2100	253.17	34.7	26.1	2.59	23.51	11.51
2400	253.17	39.58	26.1	2.59	23.51	11.51
Process Technology: 45 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	126.143	10.68	21.57	1.3	20.27	9.96
1500	126.143	13.25	21.57	1.3	20.27	9.96
1800	126.143	15.81	21.57	1.3	20.27	9.96
2100	126.143	18.38	21.57	1.3	20.27	9.96
2400	126.143	20.94	21.57	1.3	20.27	9.96
Process Technology: 32 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	70.38	5.87	26.77	1.48	25.29	12.5
1500	70.38	7.26	26.77	1.48	25.29	12.5
1800	70.38	8.64	26.77	1.48	25.29	12.5
2100	70.38	10.03	26.77	1.48	25.29	12.5
2400	70.38	11.41	26.77	1.48	25.29	12.5
Process Technology: 22 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	37.14	3.24	11.9	0.02	11.88	5.93
1500	37.14	3.98	11.9	0.02	11.88	5.93
1800	37.14	4.27	11.9	0.02	11.88	5.93
2100	37.14	5.46	11.9	0.02	11.88	5.93
2400	37.14	6.2	11.9	0.02	11.88	5.93

It should also be equipped with leakage-mitigation techniques as the processor becomes more leakage-consuming at low process technologies. By taking the Intel Core i5 (90 nm, 1200 MHz) as a base case, the power-efficient configuration can achieve around 82.6 % power saving. On the other hand,

considering the Intel Core i5 (90 nm, 2400 MHz) as a reference case, the power-efficient configuration can achieve approximately 89.6% power saving. On the other hand, Table 7 illustrates the power consumption of PaRSA-8 on Intel Core i7. It also gives the total processor area under all feature sizes. By analyzing the power consumption values provided in Table 7, it becomes apparent that the same observations that have been made based on Table 4, 5 and 6 applies also to Intel Core i7.

Table 7: Power consumption of PaRSA-8 on Intel Core i7.

Process Technology: 90 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	506.012	38.15	19.03	1.78	17.25	8.38
1500	506.012	47.53	19.03	1.78	17.25	8.38
1800	506.012	56.91	19.03	1.78	17.25	8.38
2100	506.012	66.3	19.03	1.78	17.25	8.38
2400	506.012	75.68	19.03	1.78	17.25	8.38
Process Technology: 65 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	278.038	21.16	27.31	2.74	24.57	12.01
1500	278.038	26.32	27.31	2.74	24.57	12.01
1800	278.038	31.48	27.31	2.74	24.57	12.01
2100	278.038	36.65	27.31	2.74	24.57	12.01
2400	278.038	41.81	27.31	2.74	24.57	12.01
Process Technology: 45 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	138.492	11.24	22.61	1.38	21.23	10.42
1500	138.492	13.49	22.61	1.38	21.23	10.42
1800	138.492	16.65	22.61	1.38	21.23	10.42
2100	138.492	19.35	22.61	1.38	21.23	10.42
2400	138.492	22.65	22.61	1.38	21.23	10.42
Process Technology: 32 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	76.93	6.17	28.11	1.59	26.54	13.1
1500	76.93	7.63	28.11	1.59	26.54	13.1
1800	76.93	9.09	28.11	1.59	26.54	13.1
2100	76.93	10.55	28.11	1.59	26.54	13.1
2400	76.93	12.01	28.11	1.59	26.54	13.1
Process Technology: 22 nm						
Frequency (MHz)	**Area (mm^2)**	**P_D(W)**	**P_L (W)**			
			Total	**P_{LG}**	**P_{LST}**	**P_{LST_PG}**
1200	40.48	3.4	12.51	0.02	12.49	6.22
1500	40.48	4.18	12.51	0.02	12.49	6.22
1800	40.48	4.96	12.51	0.02	12.49	6.22
2100	40.48	5.75	12.51	0.02	12.49	6.22
2400	40.48	6.63	12.51	0.02	12.49	6.22

It can be observed that PaRSA-8 consumes its lowest power on Intel Core i7 (22 nm, 1200 MHz) with power gating capability. In other words, the most power-efficient processor for PaRSA-8 is an octa-core processor implemented using small feature size and is operated at low clock rate. By taking Intel Core i7 (90 nm, 1200 MHz) as a reference case, Intel Core i7 (22 nm, 1200 MHz) achieves around 83.1% power saving. On the other hand, by taking Intel Core i7 (90 nm, 2400 MHz) as our reference case, the power-efficient configuration can achieve around 89.8% power saving.

5.3 Energy Dissipation Analysis

This section shows the energy that PaRSA-*n* consumes on the processor configurations shown in Table 1. For each configuration, *n* is set to the number of hardware threads supported by that configuration; this is the situation where PaRSA-*n* achieves its optimal performance. The total energy that PaRSA-*n* dissipates on a particular processor depends on its execution time and the total power consumed by the processor. In other words:

$$Energy = Power * Execution\,Time \tag{9}$$

Fig.11 depicts the total energy consumed by PaRSA-2 on Intel Core2Duo processor. It shows the total energy under all possible clock frequencies and feature sizes. The total energy is the sum of dynamic and leakage energies. In Fig.11, the x-axis of each subplot indicates the process technology while the y-axis shows the total energy. Each subplot corresponds to a particular clock frequency. It can be observed that, at all clock frequencies, PaRSA-2 has consumed its lowest energy when the processor was implemented using the 22 nm process technology. In addition, the most energy-efficient configuration is the Intel Core2Duo (22nm, 2800 MHz). Although the Intel Core2Duo (22 nm, 800MHz) was the Most power-efficient configuration (section 5.2), it consumes more energy that Intel Core2Duo (22 nm, 2800 MHz). This is due to the fact that processor energy depends not only on its power consumption but also on its execution time. The execution time of PaRSA-2 on Intel Core2Duo (22nm, 2800MHz) is less than its execution time on Intel Core2Duo (22nm, 800MHz); the magnitude of execution time reduction is higher than power consumption increases as PaRSA-2 moves from Intel Core2Duo (22 nm, 800MHz) to Intel Core2Duo (22 nm, 2800 MHz). Whereas Intel Core2Duo (22 nm, 2800 MHz) has a 71.70% reduction in execution time as compared to Intel Core2Duo (22 nm, 800 MHz), it has a 58.03% increases in its total power consumption. Therefore, its execution time reduction offsets its power consumption increase which in turn leads to less energy dissipation as compared to Intel Core2Duo (22 nm, 800 MHz).

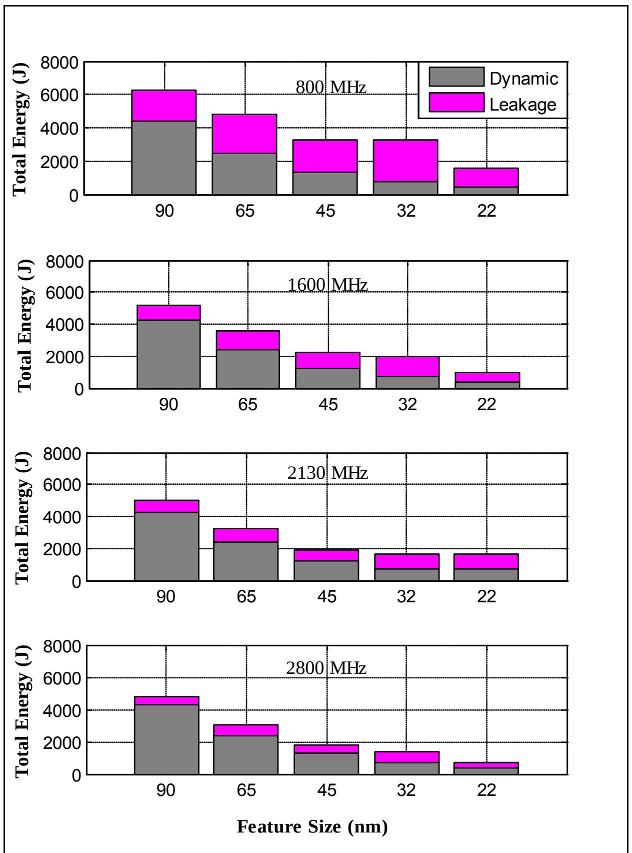

Fig.11: Energy dissipation of PaRSA-2 on Intel Core2Duo.

On the other hand, Fig.12 illustrates the energy dissipation of PaRSA-4 on Intel Core i3 processor. it shows the total energy dissipated at all possible clock rates and process technologies. Based on Fig.12, it can be observed that the most energy-efficient Intel core i3 configuration for PaRSA-4 is the Intel Core i3 (22 nm, 3300 MHz). compared with the results shown in Table 5, the most energy-efficient configuration for PaRSA-4 is not necessarily equivalent to the most power efficient configuration since the total processor energy is a function both power consumption and execution time.

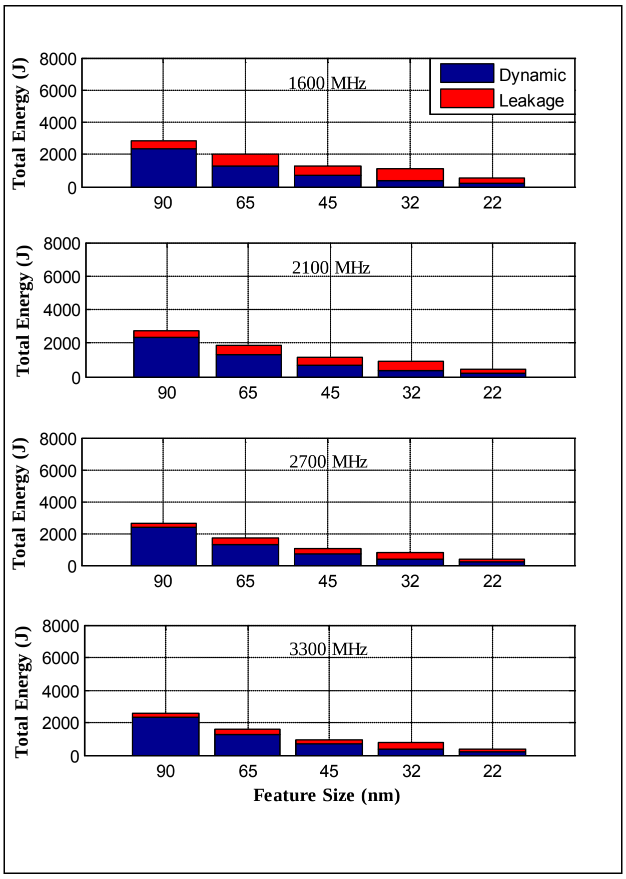

Fig.12: Energy Dissipation of PaRSA-4 on Intel Core i3.

Intel Core i3 (22 nm, 3300) has a 37.2% increase in its power consumption as compared to the Intel Core i3 (22 nm, 1600 MHz) i.e. the most power-efficient configuration. Moreover, it has achieved a 51.3% reduction in the execution time of PaRSA-4 as compared to the Intel Core i3 (22 nm, 1600 MHz). Therefore, the percent reduction in execution time outweighs the percent increase in the total power consumption which has led to an overall reduction in the total energy dissipation of the Intel Core i3 (22 nm, 3300 MHz) as it executes PaRSA-4. Fig.13 shows the total energy dissipation of PaRSA-4 on Intel Core i5 processor. The energy values are shown for each possible pair of process technology and clock rate supported by this processor. The results given by Fig.13 indicate that the most energy-efficient Intel Core i5 configuration for PaRSA-4 is that implemented at 22 nm and run at 2400 MHz i.e. Intel Core i5 (22 nm, 2400 MHz).Similar to the previous processor configurations, the energy-efficient configuration of Intel Core i5 is different from the power-efficient configuration which was found to be Intel Core i5 (22 nm, 1200 MHz). Compared to the Intel Core i5 (22 nm, 1200 MHz), Intel Core i5 (22 nm, 2400 MHz) has a 19.6% increase in power consumption and 50.8% reduction in execution time.

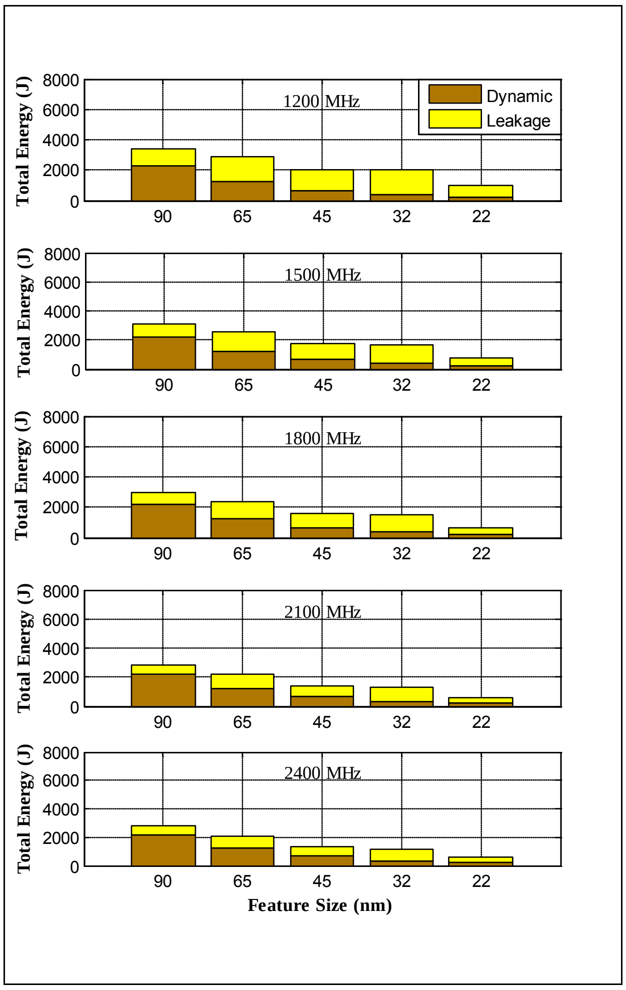

Fig.13: Energy Dissipation of PaRSA-5 on Intel Core i5.

Therefore, an overall energy saving has been achieved. So far, there are two energy-efficient processor configurations for PaRSA-4: Intel Core i3 (22nm, 3300 MHz) and Intel Core i5 (22 nm, 2400 MHz). our results indicate that PaRSA-4 has a 593.091 Joules of energy dissipation and a 31.11 seconds of execution time on Intel Core i5 (22 nm, 2400 MHz). on the other hand, it has a 363.478 Joules of energy and a 23.42 seconds of execution time on Intel Core i3 (22 nm, 3300 MHz). In other words, Intel Core i3(22 nm , 3300 MHz) has achieved a 35.5% reduction in energy and a 24.7% reduction in execution time as compared to the Intel Core i5 (22 nm, 2400 MHz). Although the two processors support the same number of hardware thread, Intel Core i3 has outperformed Intel Core i5 due to its high clock frequency; it has a lower execution time and lower energy dissipation. In summary, it can be concluded that the best Multicore processor configuration, in terms of performance and energy, for PaRSA-4 is a processor that supports four simultaneous hardware threads and runs at a high clock rate. However, this observation should be further investigated based on the lifetime reliability of the processor as will be shown in the next subsection.

Fig.14 illustrates the energy dissipation of PaRSA-8 on Intel Core i7. The total energy dissipation has been shown for all possible combinations of process technology and clock rate.

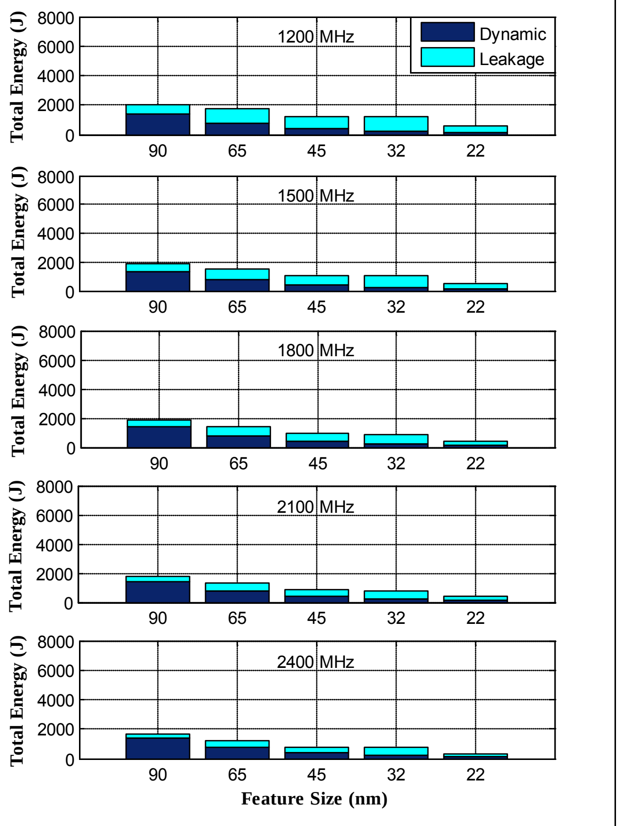

Fig.14: Energy Dissipation of PaRSA-8 on Intel Core i7.

Fig.14 implies the fact that the most energy-efficient process technology and clock rate, of Intel core i7, are 22 nm and 2400 MHz respectively. This observation confirms the aforementioned fact that the power-efficient configuration and the energy-efficient configuration are not necessarily equivalent. Whereas the power-efficient configuration for PaRSA-8 was the Intel Core i7 (22 nm, 1200 MHz), the energy-efficient configuration is the Intel Core i7 (22 nm, 2400 MHz). The energy-efficient configuration has a 19.7% increase in power consumption and 49.9% reduction in execution time as compared to the power-efficient configuration. Thus, an overall energy saving has been achieved since the amount of execution time reduction is greater than the amount of power increase. Therefore, it can be observed that the best Multicore processor configuration for PaRSA-8 from energy perspective is a processor that supports 8 hardware threads and runs at a high clock rate. Based on the results shown in this section, the most energy-efficient configurations for PaRSA-2, PaRSA-4 and PaRSA-8 are Intel Core2Duo (22 nm, 2800 MHz), Intel Core i3 (22 nm, 3300 MHz) and Intel Core i7 (22 nm, 2400 MHz) respectively.Table 8 summarizes the execution time and the total energy of the optimal configurations for PaRSA-*n* where *n* is the number of hardware threads supported by the associated processor. The term optimal refers to the processor configuration that has achieved the lowest execution time and energy dissipation among all configurations that support the same number of hardware threads.

Table 8: PaRSA-n Optimal Configuration Parameters.

Algorithm	**Configuration**	**Energy (Joules)**	**Time (Seconds)**
PaRSA-2	Intel Core2Duo (22 nm, 2800 MHz)	700.994	45.08
PaRSA-4	Intel Core i3 (22 nm, 3300 MHz)	363.4784	23.42
PaRSA-8	Intel Core i7 (22 nm, 2400 MHz)	341.5776	17.94

The observation that can be made based on Table 8 is that PaRSA-*n* can achieve a substantial performance improvement and energy savings by increasing the number of hardware threads and using a processor configuration whose number of hardware threads is at least equal to *n*. PaRSA-8 has achieved a 60.2 % performance improvement as compared to PaRSA-2 and a 23.4 % performance improvement as compared to PaRSA-4. On the other hand, it has achieved a 51.3 energy saving as compared to PaRSA-2 and a 6.03 % energy saving as compared to PaRSA-4. Therefore, from performance and energy perspectives, PaRSA-8 is the best implementation of the RSA algorithm and Intel Core i7 (22 nm, 2400 MHz) is the optimal processor configuration for this algorithm. However, this result should be further investigated in terms of lifetime reliability as will be shown in the next subsection.

5.4 Lifetime Reliability Analysis

This section shows the lifetime reliability of the different processor configurations under the PaRSA-*n* workload. In section 5.3, it has been shown that all the energy-efficient configurations were obtained at the same process technology. Consequently, only clock rate has been considered for the sake of reliability analysis. As shown in section 4.3, the reliability analysis framework relies mainly on processor temperature in order to quantify its lifetime reliability [40-42]. Hence, it is necessary to obtain the temperature of the processor at different clock rates. In order to achieve this goal, the psensor [48] utility has been used to read the temperature of the processor while a particular PaRSA-*n* workload is running. In order to estimate the lifetime reliability at clock rates other than those supported by the real hardware, an empirical model has been developed. This model captures the relationship between processor's temperature and its clock rate and can be used to predict processor's temperature at each possible clock rate. Fig.15 illustrates the relationship between processor temperature and its clock rate. The temperatures obtained by psensor were first plotted and a curve-fitting operation has been performed to obtain a mathematical formula that expresses processor temperature as a function of its clock rate. It can be observed that there is a quadratic relationship between processor temperature and its clock rate. The temperature values obtained by the psensor utility or the developed model have then be input to the RAMP model [40-42] in order to get an estimate of the processor's lifetime reliability as it runs a particular PaRSA-*n* workload. Fig.16 depicts the mean time to failure (MTTF) as a function of clock frequency. It shows the MTTF based on each of the physical failure mechanisms described in section 4.3. The x-axis indicates the clock rate, while the y-axis shows the MTTF as a function of the clock rate. Each subplot is labeled with the corresponding failure mechanism.

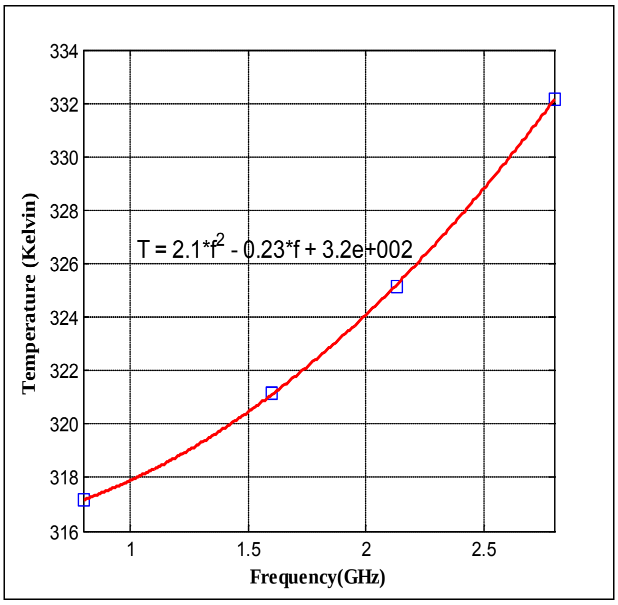

Fig.15: The Relationship between processor temperature and its clock rate.

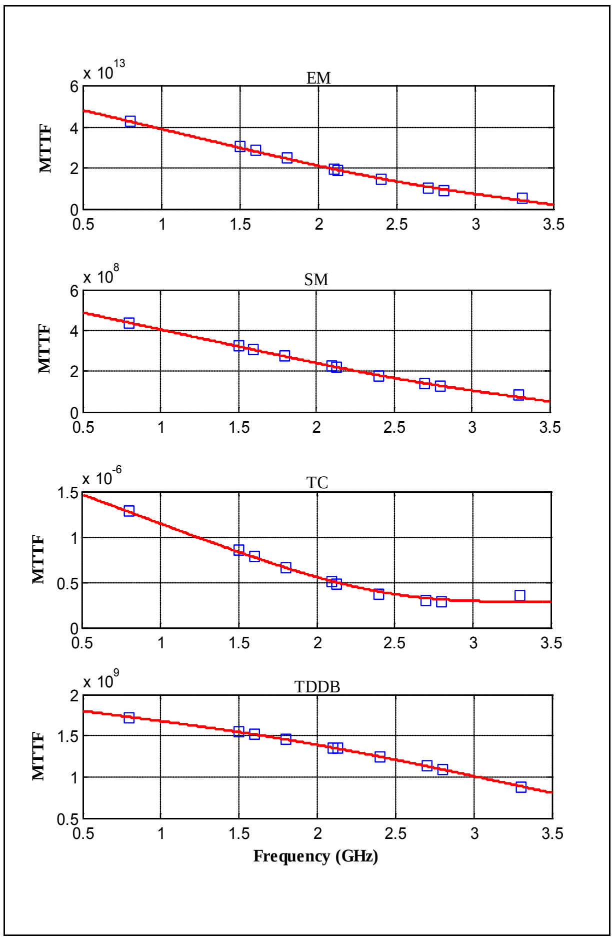

Fig.16: Processor's MTTF as a function of clock rate.

Fig.16 illustrates the fact that the MTTF of a processor is a decreasing function of clock rate. In other words, a processor running at a low clock rate would sustain a longer period of time before encountering temperature-induced failures as compared to a processor that runs on a higher clock rate. Therefore, it is important to re-evaluate the optimal configurations specified in section 5.3 based on the lifetime reliability of the processor. Table 9 shows a comparison between the power-efficient and the energy-efficient configuration for each PaRSA-*n*. It shows the percent increase in MTTF that the power- efficient configuration can achieve as compared to the energy-efficient configuration for each PaRSA-*n* under all possible physical failure mechanisms. The comparison results have been reported in terms of the percent increase in MTTF values since processor reliability is directly proportional to its MTTF.

Table 9: MTTF comparison.

Algorithm	MTTF (EM)	MTTF (SM)	MTTF (TDDB)	MTTF (TC)
PaRSA-2	368.5%	251.2%	59.26%	333.3%
PaRSA-4	458.82%	280.25%	72.73%	100.1
PaRSA-8	146.58%	112.43%	30.40%	150%

As shown in sections 5.2 and 5.3, power-efficient configurations have lower clock rates than the energy-efficient ones. Therefore, they exhibit low heat dissipation and spans a longer lifetime as compared to energy-efficient processors that run at higher clock rates. This fact can be directly observed from Table 9 which shows that moving from energy-efficient configuration (i.e. higher clock rate) to power-efficient configuration (i.e. lower clock rate) results into a significant increase in processor reliability under various physical failure mechanisms. Taking lifetime reliability into consideration, the optimal configurations for PaRSA-2, PaRSA-4 and PaRSA-n will be Intel Core2Duo (22 nm, 800 MHz), Intel Core i3 (22 nm, 1600 MHz) and Intel Core i7 (22 nm, 2400 MHz) respectively. Based on the results shown in this section, it can be observed that the optimal processor configuration for RSA is a Multicore processor with a large number of hardware threads, low clock rate and a small feature size.

6. Conclusion

In this paper, an extensive design space exploration (DSE) has been performed in order to figure out the optimal Multicore processor configuration for cryptographic algorithms. All experiments were based on a parallel version of the RSA algorithm tuned for optimal performance settings. Our results indicate that a careful balance between processor specifications i.e. Clock rate, number of hardware threads and process technology should be achieved in order to obtain the optimal processor configuration that maintains a reasonable tradeoff between performance, power consumption, energy dissipation and lifetime reliability of the processor. However, the appropriate setting of processor specifications depends on the design constraints and system requirements.

References:

[1] www.us-cert.gov.

[2] J. Pieprzyk and David Pointcheval, *Parallel Authentication and Public key encryption. Information Security and Privacy,* Lecture notes in computer science, Vol. 2727, 2003,pp. 387-401.

[3] R. L. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems,* Communications of the ACM, Vol. 21, Issue 2, 1978, pp. 120-126.

[4] P Barrett, *Implementating the Rivest, Shamir and Aldham Public-key Encryption Algorithm on Standard Digital Signal Processor,* Proceedings of CRYPTO'86, Lecture Notes in Computer Science, 1986, pp. 311–323.

[5] B. Chapman, G. Jost, and R. V. D. Pas, *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation),* The MIT Press, 2007.

[6] A. J. Menezes, S. A. Vanstone and P. C. V. Oorschot, *Handbook of Applied Cryptography,* CRC Press, Inc., Boca Raton, FL, USA, 1986.

[7] C. Fu and Z.-L Zhu, An Efficient *Implementation of rsa Digital Signature*

Algorith,. In Proc. of the 4th International Conference on Wireless Communications, Networking and Mobile Com-puting, 2008, pp. 1–4.

[8] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. 22, No. 6, 1976, pp. 644–654.

[9] P. Hamalainen, N. Liu, M. Hannikainen and T. D. Hamalainen, *Acceleration of Modular Exponentiation on System-on-a-Programmable-Chip*, In. Proc. of the International Symposium of the system-on-Chip, 2005, pp. 14-17.

[10] K. Skadron, P. S. Ahuja, M. Martonosi and D. W. Clark, *Branch Prediction, Instruction-Window Size, and Cache Size: Performance Trade-Offs and Simulation Techniques*, IEEE Transactions on Computers, Vol. 48, Issue 11, 1999, pp. 1260-1281.

[11] A. Gellert, G. Palermo, V. Zaccaria, A. Florea, L. Vintan and C. Silvanto, *Energy-Performance Design Space Exploration in SMT Architectures Exploiting Load Value Predictions*, In Proc. of the Design Automation and Test in Europe Conference and Exhibition (Date), 2010, pp. 271-274.

[12] S. K. Dash, T. Srikanthan, *Instruction Cache Tuning for Embedded Multitasking Applications*, In Proc. of the IEEE/IFIP International Symposium on Rapid System Prototyping, 2009, pp. 152-158.

[13] T. S. R. Kumar, C. P. Ravikumar and R. Govindarajan, *Memory Architecture Exploration Framework for Cache Based Embedded SoC,* In Proc. of the 21st International Conference on VLSI Design, 2008, pp. 553-559.

[14] M. Y. Qadri and K. D. M. Maier, *Data Cache Energy and Throughput Models: Design Exploration for Embedded Processor,* EURASIP Journal on Embedded Systems, 2009 , Article 13 (Jan. 2009).

[15] A. G. Silva-Filho, F. R. Cordeiro, C. C. Araujo, A. Sarmento, M. Gomes, E. Barros and M. E. Lima, *An ESL Approach for Energy Consumption Analysis of Cache Memories in SoC Platforms,* International Journal of Reconfigurable Computing, Vol. 2011, pp. 1-12,2011.

[16] S. Przybylski, M. Horowitz and J. Hennessy, *Performance Tradeoffs in Cache Design,* In Proc. of the 15th Annual International Symposium on Computer Architecture, 1988, pp. 290-298.

[17] M. Alipour and M. E. Salehi, *Design Space Exploration to Find the Optimum Cache and Register File Size for Embedded Applications,* In Proc. of the 9th International Conference on Embedded Systems and Applications, 2011, pp. 214-219.

[18] M. Alipour, H. Taghdisi and S. H. Sadeghzadeh, *Multi objective design space exploration of cache for embedded applications,* In. Proc. of the 25th IEEE Canadian Conference on Electrical and Computer Engineering, 2012, pp.1-4.

[19] Y. Cai, M. T. Schmitz, A. Ejlali, B. Al-Hashimi and S. R. Reddy, *Cache Size Selection for Performance, Energy and Reliability of Time-Constrained Systems*, In Proc. of Asia and South Pacific Conference on Design Automation, 2006, pp. 923-928.

[20] M. Alipour and H. Taghdisi, *Effect of Thread Level Parallesim on the Performance of Optimum Architecture for Embedded Applications,* International Journal of Embedded systems and Applications. Vol. 2, No. 1, 2012, pp. 15-24.

[21] S. Eyerman, L. Eeckhout and K. D. Bosschere, *Efficient Design Space Exploration of High Performance Embedded Out-of-Order Processors,* In. Proc. of Design Automation and Test in Europe, 2006, pp. 1-6.

[22] G. Palermo, C Silvano and V. Zaccaria, *Multi-objective Design Space Exploration of Embedded Systems, Journal of Embedded Computing,* Vol. 1, Issue 3, 2005, pp. 305-316.

[23] A. Assaduzzamn, F. Sibai and M. Rani, *Impact of level-2 Cache Sharing on the Performance and Power Requirements of Homogenous Multicore Embedded Systems,* Microprocessors

and Microsystems, Vo. 33,Issue 5-6, 2009, pp. 388-397.

[24] A. Assaduzzamn and M. Rani, *Level-2 Shared Cache versus Level-2 Dedicated Cache for Homogenous Multicore Embedded Systems*, In Proc. of the 7th International Conference on Computing, Communications and Control Technologies, 2009.

[25] A. Assaduzzaman, M. Rani and F. Sibai, *On the Design of Low-Power Cache Memories for Homogenous Multicore Processor*, In Proc. of the 22nd International Conference on Microelectronics, 2010, pp. 387-390.

[26] A. Assaduzzaman, *A Power-Aware Multi-Level Cache Organization Effective for Multicore Embedded Systems*, Journal of Computers, Vol. 8, No. 1, 2013, pp. 49-60.

[27] F. Sibai, *On the Performance Benefits of Sharing and Privatizing Second and Third Level Cache Memories in Homogenous Multi-core Architectures*, Microprocessors and Microsystems, Vol. 32, Issue 7 , 2008, pp. 405-412.

[28] V. Saravanan, S. K. Chandran, S. Punnekkat and D. P. Kothari, *A Study on the Factors Influencing Power Consumption in Multithreaded and Multicore CPUs*, WSEAS Transactions on Computers, Vol. 10, Issue 3, 2011, pp. 93-103.

[29] W. Bielecki and D. Burak, *Parallelization of the AES Algorithm*, In. Proc. of the 4th WSEAS International Conference on Information Security, Communications and Computers,2005, pp.224-228.

[30] S. Saxena, N. Kishore, D. Handa and B. Kapoor, *Comparative Analysis of Sequential and Parallel Implementations of RSA*, International Journal of Scientific and Engineering Research, Vol. 4, Issue 8, 2013,pp. 2100-2103.

[31] V. Garg and V. Arunachalam, *Architectural Analysis of RSA Cryptosystem on FPGA*, International Journal of Computer Applications, Vol. 26, No. 8, 2011, pp. 30-34.

[32] M. Hill and M. Marty, *Amdahl's law in the Multicore era*, IEEE Computer, vol. 41, no. 7, 2008, pp.33 -38.

[33] S. Akther and J. Roberts, *MultiCore Programming: Increasing Performance through Software Multi-threading*, The Intel Press, 2006.

[34] www.fedoraproject.org.

[35] www.intel.com.

[36] wiki.archlinux.org.

[37] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen and N. P. Jouppi, *The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area and Timing*, ACM Transactions on Architecture and Code Optimization.,Vol. 10, No. 1 , 2013, Article 5.

[38] www.itrs.net.

[39] R. Ubal, B. Jang, P. Mistry, D. schaa and D. Kaeli, *Multi2Sim: a simulation framework for CPU-GPU computing*, In Proc. of the 21st International Conference on Parallel Architectures and Compilation Techniques, 2012, pp. 335-344.

[40]J. Srinivasan, S. V. Adve, P. Bose and J. Rivers, *The Case for Lifetime Reliability-Aware Microprocessors*, In Proc. of the 31st International Symposium on Computer architecture, 2004, pp. 276-287.

[41] J. Srinivasan, S. V. Adve, P. Bose and J. Rivers, *Lifetime Reliability: Toward an Architectural Solution*, IEEE Micro, Vol. 25, Issue 3, 2005, pp. 70-80.

[42] J. Srinivasan, S. V. Adve, P. Bose and J. Rivers, *The Impact of Technology Scaling on Lifetime Reliability*, In Proc. of the International Conference on Dependable Systems and Networks, 2004, pp. 177-186.

[43] D. A. Patterson and J. L. Hennessy, *Computer Architecture: a Quantitative Approach*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA,2006.

[44] J. J. Clement, *Electromigration Modeling for Integrated Circuit Interconnect Reliability Analysis*, IEEE Transactions on Device and Materials Reliability. Vol. 1, Issue 1, 2001, pp. 33-42.

[45] Yi- L. Cheng, B-J. Wei and Yi-L. Wang, *Scaling Effect on Electromigration in Copper Interconnects*, In Proc. of the 16th IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits, 2009, pp. 698-701.

[46]J. H. Stathis, *Reliability limits for the gate Insulator in CMOS technology,* IBM Journal of Research and Development, Vol. 46, 2002, pp. 265-286.

[47] E. Wu, J.Sune, W. Lai, E. Nowak, J. McKenna, A. Vayshenker and D. Harmon, *Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides. Solid-state Electronics Journal*, Vol. 46, 2002, pp. 1787-1798.

[48] https://aur.archlinux.org/packages/psensor.

[49] S. Borkar, *Design Challenges of Technology Scaling*, IEEE Micro, Vol. 19, Issue 4, 1999, pp. 23-29.

[50] A. P. Chandrakasan, S. Sheng and R. W. Brodersen, *Low-power CMOS Digital Desig, IEEE Journal of Solid-state Circuits*.Vol. 27, Issue 4, 1992, pp.473-484.