Scientific Journal of Impact Factor (SJIF): 3.134

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2, Issue 12, December -2015

Implementation and Performance Evaluation of AES Algorithm on soft core processor with RTOS

Preeti Dixit[1], Jitendra Zalke[2]

[1]Department of Electronics Engineering, R.C.O.E.M
[2]Department Electronic Design and Technology, R.C.O.E.M

Abstract—*In the present scenario FPGA technology has capability to design high performance embedded systems which is based on its soft core and hard core processors, embedded memories and IP core. To achieve high performance scheduling is important as almost all programs have to wait for I/O of some kind, even a fetch from memory takes a longer time as compared to the speed of the CPU. In a simple system running a single process, the time in waiting for I/O is washed out, and therefore CPU cycles for it hence are lost forever. A scheduling system allows one process to use the CPU while further is waiting for I/O, thereby it makes a full use of lost CPU cycles. The challenge to build system "efficient" and "fair" achievable, issue to changeable environment, often subject to shifting priority policies. For this we need to have an efficient scheduling algorithm. Deciding a particular scheduling algorithm for process requires the familiarity of resource use for the definite scheduling plan. Hence, a proper study of the various scheduling algorithms with respect to the resource utilization is important. Thus the proposed work is to schedule AES cryptography algorithm which is taken as an application, on a soft core processor made using Microblaze with and without RTOS and comparison is done with relative speed obtained. XILKERNEL is chosen, the incorporation of XILKERNEL RTOS in FPGA enables implementation of various real-time scheduling algorithm like Round Robin and Priority Scheduling.*

Keywords: *XILKERNEL, MICROBLAZE, SCHEDULER,AES, RTOS*

I. INTRODUCTION

Real-time operating systems (RTOSs) is the need of the hour and hence used for development of applications for systems with complex time and where constraints of resource prevails. This situation often leads to automation, for example where one or more computers must synchronize the activities among one or more instruments involving time, process, or precedent constraints. Real time Operating system can be hard real time or soft real time. In the hard real time system its the necessity to meet the given timing constraint otherwise the system fails altogether, on the other hand in soft real time system even if the deadline is missed by few time unit, complete failure does not occurs. In OS the program can be divided into several task; which are to be scheduled efficiently to get maximum output. If the task are too many and not properly managed it may lead to contention[16]. Therefore the scheduler helps in managing the task and kernel helps in managing resources.

The scheduler which is an integral part of the RTOS is useful in determining the state of task i.e. which task will wait and which will execute. A scheduler aims at maximizing the throughput, dropping the waiting time. As a result the tasks are shuffled in these three stages. A task can be blocked when it has run out of things so that time can be managed properly. A task can be sent to ready state if any higher priority task arrives which needs immediate attention. As soon as the higher priority task is done the ready state task is shifted to running state and is executed. By this the proper management of time and resource is hence achieved.

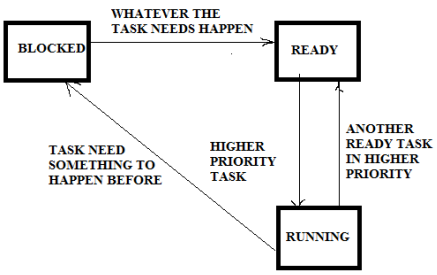

Figure 1.Task State

Kernel is nothing but a set of libraries. Kernel decides effective context switching hence is handled by the kernel smartly. Kernel are of two types microkernel and monolithic kernel. In RTOS microkernel are used most of the time. It serves as an abstraction level between the system resources and the hardware applications and hence provide communication

between the hardware. In a real time operating system the kernel is highly customized as it contains minimum number of services required to run a task. The kernel is responsible for various work such as task synchronization, task management, task scheduling, memory management, exception handling if at all it occurs, and very important time management and interrupt handling.

In the proposed work AES[10] is chosen as application to explore different capability of scheduling algorithm supported by the Xilkernel. Xilkernel module containing the scheduler supports both priority-driven, preemptive scheduling through time slicing or round-robin scheduling. Both of this are global scheduling policy and cannot be altered on a per-thread basis. This must be configured statically at kernel generation time. XPS provides us with settings to modify the kernel as per the requirement. It allows the user for example to initiate interrupt handlers, add thread support, specify clock resolution. All modifiable settings can be seen in Xilinx OS and libraries document[9].

Organization of the paper is as follows, II sections describes the background and the related work done in this field with the description of XILKERNEL and related work done. Section III describes the methodology that is adopted. The propose work aims to schedule AES on RTOS environment and to do the operation in a given time span and to use resource very efficiently.

II. BACKGROUND AND RELATED WORK

A. Embedded System

Embedded system are mainly dedicated to single functioned. It has constraints related to the resources, cost, size, power and performance. A tradeoff is always present amongst the above mentioned factors. An embedded system should be reactive and use very less resources. To obtain high speed and handle the complexity an OS/RTOS is used. The core of OS is kernel that contains scheduler which schedules threads, leading to multithreading and hence leading to speed enhancement, but there are relating issues that is how to make the threads work properly together.

B. Xilkernel

Xilkernel a very small and robust kernel for Xilinx embedded processors. It is integrated into the XPS, so therefore it makes it to be able to configure and also a platform can be built using Xilkernel. Xilkernel key features are a POSIX API targeting embedded systems. Here threads can be statistically declared to startup with the kernel which may be utilized in multitasking. The kernel is highly scalable. It has a POSIX support application development and porting are made to be easy as it can be made to run almost at power on other kernels that use the POSIX as a standard[9].

In figure2 shows Xilkernel Module. Xilkernel is a set of library of kernel functions. It supports Round Robin algorithm

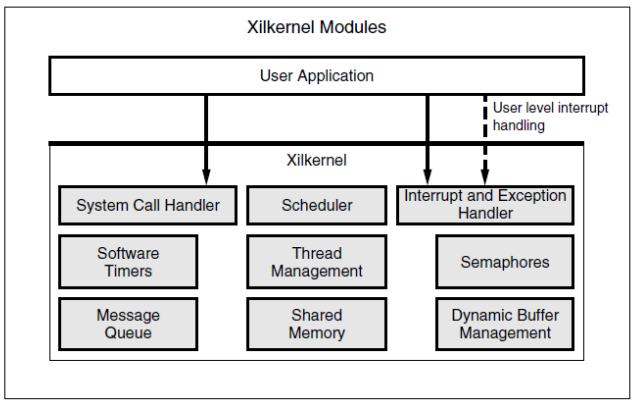

Figure 2. Xilkernel Module

and Priority based preemptive scheduling and it cannot be altered on the per thread basis. A process in Xilkernel can have the following states as shown in figure3 [9]

- PROC NEW - When a process is created it enters this state.
- PROC READY - A process that is ready to execute.
- PROC RUN -A running process.
- PROC WAIT - A process blocked on a shared resource.

• PROC DELAY - A process waiting on a timeout.
•PROC TIMED WAIT - A process blocked on a shared resource with a timeout.
• PROC DEAD - A terminated process.

The kernel decides according to the scheduling policy which task should be in which state as per the state shown in the figure3.

C. Scheduling policies

Round Robin Scheduling Policy is a very intuitive policy here there are task that is to be scheduled after a regular period of time, Round robin scheduling is a preemptive of first-come first-served scheduling. Processes are served in a first-in-first-out fashion but here each and every process is allowed to run for a limited amount of time. Once the process completes, that particular process is placed on the end of the run queue and then has to wait for its turn. Benefit of round robin scheduling significantly improves average response times if the processes are fewer in number.

In Round Robin scheduling all the processes are taken to be equally important, generally is not true every time. As in case CPU may get CPU intensive task or less intensive task so setting a priority to the task helps in efficient execution leading to speed enhancement[4]. A priority scheduler, the scheduler explicitly takes up the uppermost priority process to run. The system can be preemptive or non preemptive, a preemptive one is chosen for a real time purpose as an interrupt has to be acknowledged and served as soon as possible as the interrupt increases the overall reactivity of the system. Preemptive scheduling, a process branches at whatever time a higher priority becomes offered in the run stand in line.

Priority scheduling provides with a better mechanism here the importance is relative of all process may be correctly defined. In the Xilkernel the highest priority task is assigned 0.

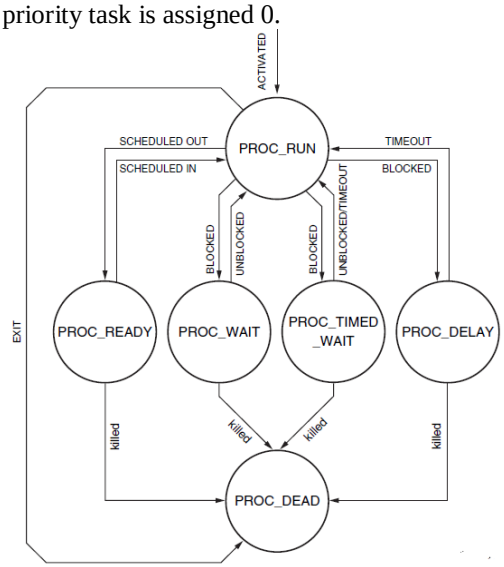

Figure 3. Process Context Switching

The various parameters like the turnaround time, waiting time, response time are the parameter to decide which is good for which application. According to the applications need, for increasing the performance, the scheduling policy are decided. Scheduling Policy Earliest Deadline First, here the scheduling is done according to the task having the least deadline, here no priority is assigned but the task gets virtual priority in accordance with their deadline. It has better performance on a processor but not suited for multiprocessor. Rate Monotonic Algorithm is good for scheduling on a multiprocessor here according to the rate of occurrence of the task the priority is decided, it is static.

D. *Thread Synchronization and Communication*

Scheduling increases speed of operation, it involves many process, threads that helps in speedy operation and hence their synchronization is also important. Single sequential flow in process is known as thread. Multiple threads are present in one process which happens to share heap storage, static storage, and code. Every thread possess own registers and stack. Threads share the same single address space and synchronization is needed when threads access same memory locations. So in an application there are many threads that try to access resources at a time which creates chaos and so comes the need of synchronization. It is ensured by the use of semaphore or mutex offered by the Xilkernel. Semaphore is a signaling device which helps prevent problem like shared resource i.e. more than a thread trying to make change at same memory location. Mutex is a simpler version of semaphore used where only a resource is involved.

The intercommunication amongst the thread is very important. It is done by Shared Memory or Message Queue offered by Xilkernel[9]. Message Queue is buffer used in non sharing environment, message to be communicated is put in

the buffer common to all. Shared Memory is used in a way where a thread stores data in a specific location from where the other threads reads from. This communication helps in speedy processing.

E. Microblaze

MicroBlaze is a RISC core. It is developed and maintained by Xilinx and is therefore optimized for Xilinx FPGA boards. An overview of MicroBlaze architecture can be seen in figure4.

The MicroBlaze is a soft core processor that is to a great extent configurable, that allows us to select a specific features required as per the design[8]. The features are thirty-two 32-bit general purpose registers, 32-bit instruction word with three operands and two addressing modes. It has 32-bit address bus, single issue pipeline. The MicroBlaze processor allows enabling of additional functionality as per the demand of the design to be realized[6].

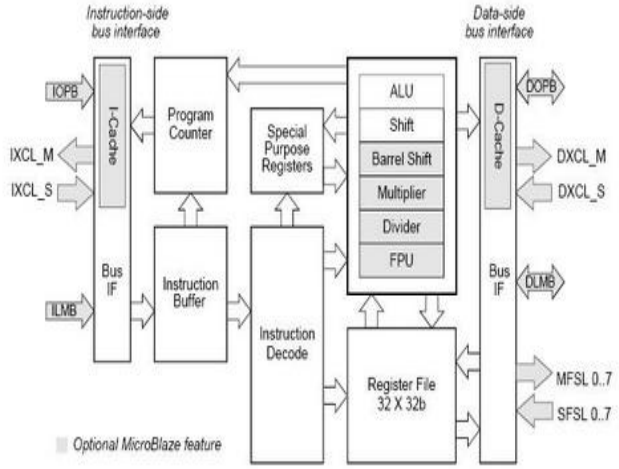

Figure 4. Overview of Microblaze

F. AES Algorithm

The AES algorithm [10] [20]which is very popular form of cryptology due to its wide acceptance in the applications like cloud computing security, banking. AES the initialization the key size utilized for an AES indicates the quantity of reiterations of change adjusts that change over the data, called the plaintext, into the last yield, called the cipher text. The quantity of cycles of redundancy are as per the following: 10 cycles of reiteration for 128 keys, 12 cycles of reiteration for 192 and 14 cycles of reiteration for 256 bit keys. In figure 5 the first step is Key Expansions that is round keys, are got from the figure key utilizing Rijndael's key calendar. AES requires a different 128 bit round key square for each round in addition to one more. Here in the initial Round is AddRoundKey, every byte of the state is joined with a square of the round key utilizing bitwise xor. Rounds are namely:

•SubBytes—a non-direct substitution step where every byte is supplanted with another as indicated by a lookup table.
•ShiftRows—a transposition step where the last three columns of the state are moved consistently a certain number of steps.
•MixColumns—a blending operation that works of the segments of the state, joining the four bytes in every section.
•AddRoundKey.

In the last Round (no MixColumns) but SubBytes, ShiftRows, add round key, this scrambles the data generating the chipper text to be sent as the encrypted data. After this decrypting the data where AES unscrambling is not indistinguishable to encryption here steps done backward it utilizes inverses of every stride with an alternate key calendar since result is unaltered when swap byte substitution and movement columns swap blend sections and include (changed) round key.

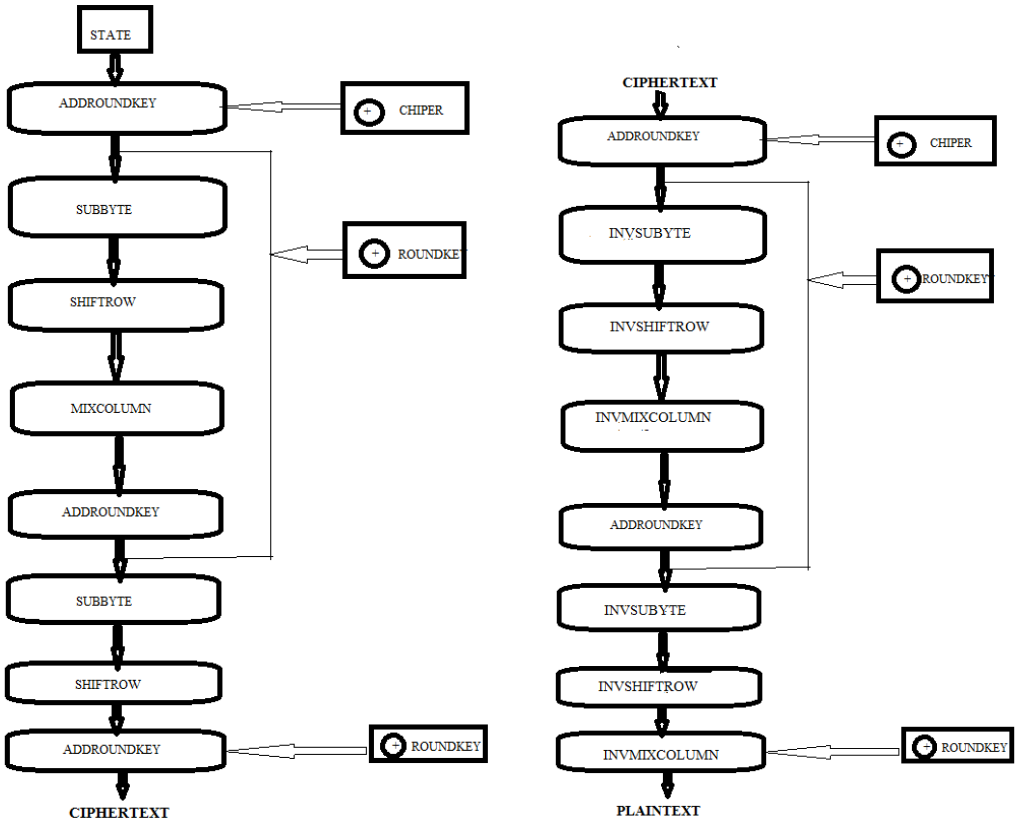

Figure 5. AES Encryption and Decryption

III. DESIGN METHODOLOGY

A. Implementation of AES on Microblaze soft core processor without RTOS:

AES is implemented in C language on the soft core processor Microblaze to measure the various performance parameter like the throughput, response time, the resource utilization and the time factor. The timing is measured with the Software Profiling. The resource utilized can be seen from the .mrp and .par file[2]. The parameters like throughput, waiting time, turnaround time can be calculated by measuring the clock ticks as per the definitions shown in table-1, Throughput is the total amount of work done in a unit of time. Waiting Time is the amount of time that is taken by a process in ready queue. Turn Around Time is the sum total of waiting time & execution time.

Table-1-To calculate Parameters

Task	Clock ticks used
1	a
2	b
3	c
4	d

Waiting time for task2 = (a+b)/2

Throughput for task2 = 2/a+b+context switch time(cs)

Response time for task2 = (0+a+CS+a+b+2CS)/2

Figure 6 shows that the AES algorithm is mapped on the Microblaze. Here we will firstly run the AES encryption and decryption algorithm on a soft core processor, the processor will be made using Xilinx Platform Studio then further its

timing analysis is done. In AES the initialization the key size utilized for an AES indicates the quantity of reiterations of change adjusts that change over the plaintext, into the last yield, called the cipher text. After this decryption, AES unscrambling here steps done backward it utilizes inverses of every step with an key. Successful implementation of AES on microblaze was done. Figure-7 shows result of the encrypted text, cipher text, decrypted text.

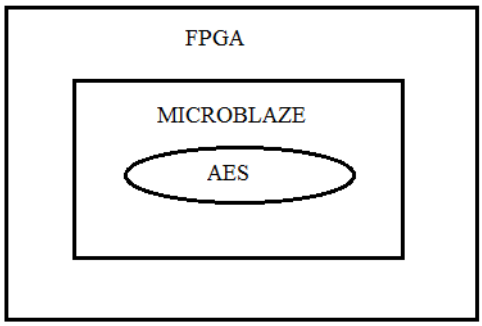

Figure 6. Implementation of AES algorithm on soft core processor

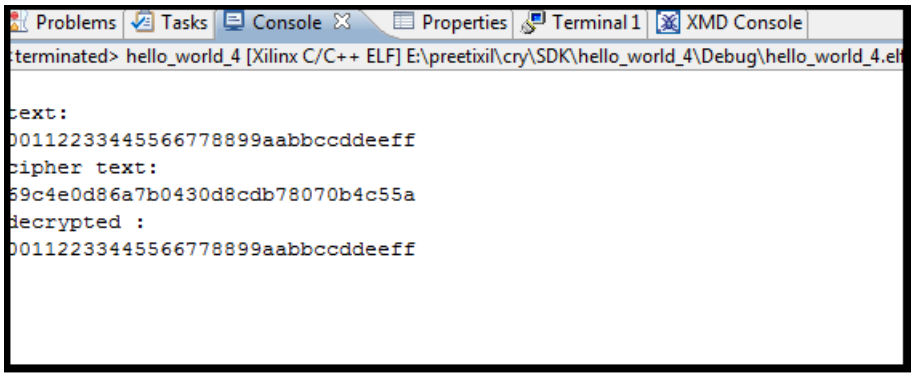

Figure 7. Result of AES Encryption and decryption

Figure 8 shows the profiling results of the implementation. Profiling gives the timing information by creating the gmon file. The results obtained as per the software profiling tells time consumed during encryption and decryption without RTOS. Table-2 shows the encryption and decryption time.

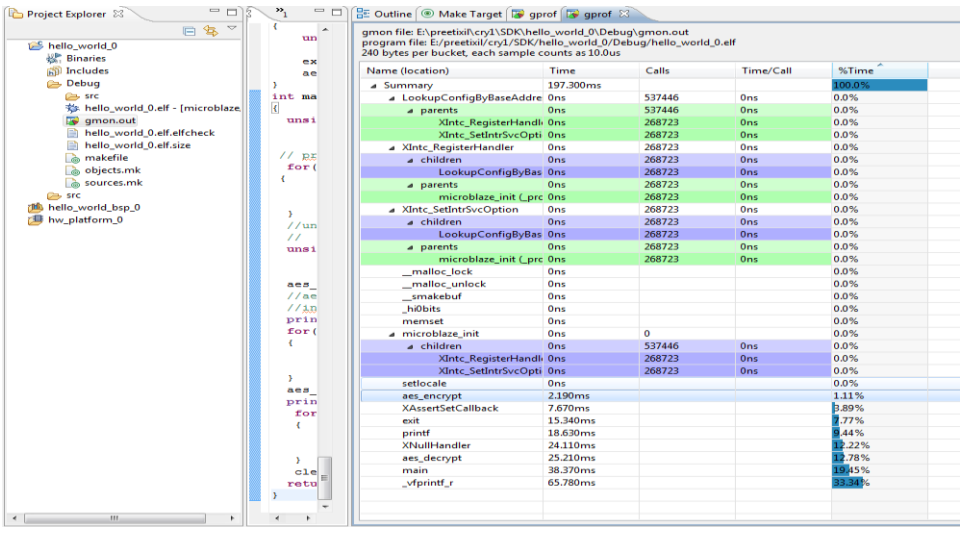

Figure 8. Software profiling results

Table-2-Timing Calculation

Sr. No.	Process	Time taken to Execute
1	Encryption	21.92ms
2	Decryption	13.16ms

The various parameters calculated from the software profiling results are as follows:

1) The waiting time for encryption process is: 2.190ms
2) The waiting time for decryption process is: 25.210ms
3) Throughput for encryption process: 0.033ms
4) Throughput for decryption process: 0.072ms
5) Response time for encryption process: 0.03s
6) Response time for decryption process: 0.013s

B. Implementation of AES using Xilkernel

Figure 9 shows the flow diagram of the proposed design of implementing AES using the XILKERNAL, in which the Microblaze will be configured in EDK then it will be exported and launched on SDK which facilitates creating of application. Firstly without using RTOS the data will be encrypted to ensure safe and secure transmission.

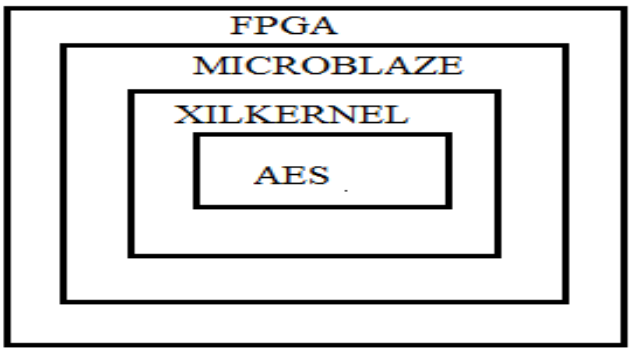

Figure 9. AES algorithm implementation with RTOS (Xilkernel)

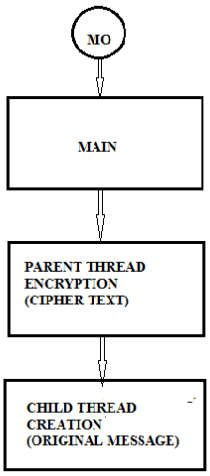

Figure 10. Thread Creation

The AES algorithm is divided into the two threads in order to increase efficiency of the system by reducing the idle time spent in memory bound operations. Figure 10 shows, here the process is initiated by loading of kernel then the control passes to the main program where there are two threads. The parent thread calling the child thread. The parent thread is of encryption by adding the round key and then forming the cipher text which is being communicated. This leads to child thread initialisation where the decryption algorithm is carried. The encryption and decryption here is done on a single processor.

C. Proposed system for implementation

The proposed system is shown in figure 11, here the audio clip in the mp3 format is taken as an input data[19]. This input data needs to be communicated in real time scenario with security. Then the audio will be provided to the parent thread of encryption. The encrypted audio signal turns the audio that is the input into some encrypted audio that is like a noise, a scrambled data. Then the decryption will be done in real time scenario by the child thread, getting back the original audio. It is received successfully after decrypting the audio which is coded. Further the task of encryption and decryption will be divided into more threads to efficiently make the execution speedy and to utilize the resources efficiently.

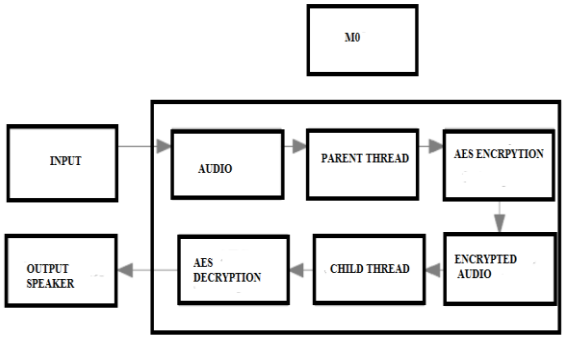

Figure 11. Proposed system

IV. CONCLUSION

This paper aims at exploring the time and resource management which is provided by the RTOS (Xilkernel). The AES algorithm is taken as application to encrypt and decrypt the bulk of data for the secure transmission purpose. The AES algorithm works out here without using RTOS and its various timing parameters are analyzed. Further to improve, use of embedded recourses and to get better timing administration, the implementation of AES algorithm with RTOS (Xilkernel) is proposed. RTOS is consistent with many number of task, by undertaking synchronization, and deterministic practices. RTOS uses shared memory, queue to communicate between threads unlike the normal OS to save time. Further the paper discuss the different functionalities of XILKERNEL, which is an RTOS. The POSIX string abilities in the single processor are illustrated.

REFERENCES

[1] M.Reddy1,Y.Amar,"Evaluation Of Microblaze and Implementation Of AES Algorithm using Spartan-3E",Vol. 2, Issue 7, July 2013 International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

[2] S.Saha,A.Chkrabarti,R.Ghosh"Exploration of Multi-thread Processing on XILKERNEL for FPGA Based Embedded Systems",2013 19th International Conference on Control Systems and Computer Science.

[3] A.Allkoci,E.Dhima,I.Tafa,"Comparing Priority and Round Robin Scheduling Algorithms",IJCSI International Journal of Computer Science Issues, Vol. 11, Issue 3, No 1, May 2014.

[4] P.Singh,V.Singh,A.Pandey,"Analysis and Comparison of CPU Scheduling Algorithms",ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 1, January 2014,International Journal of Emerging Technology and Advanced Engineering

[5] A.Kamboh,A.Krishnamurthy,J.Vallabhaneni,"Demonstration of Multitasking using ThreadX RTOS on Microblaze and PowerPC".

[6] R.Klenke"Experiences Using the Xilinx Microblaze Softcore Processor and uCLinux in Computer Engineering Capstone Senior Design Projects"

[7] R.Paul,S.Saha;S.Sau;A.Chakrabarti4,"Real Time Communication between Multiple FPGA Systems in Multitasking Environment Using RTOS"

[8] "Microblaze processor reference guide." [Online]. Available: http://www.xilinx.com

[9] "Xilkernel (v5.00.a), ug708." [Online].

[10] A.Deshpande, M.Deshpande, D.Kayatanavar,"FPGA Implementation of AES Encryption and Decryption",International Conference On "Control, Automation, Communication And Energy Conservation - 2009, 4th-6th June 2009

[11] B Weerasinghe ,"Improving Throughput of RC4 Algorithm using Multithreading Techniques in Multicore Processors ",International Journal of Computer Applications (0975 – 8887) Volume 60– No.16, December 2012.

[12] A.Bastoni, B. Brandenburg,H. AndersonDepartment"An Empirical Comparison of Global, Partitioned, and Clustered Multiprocessor EDF Schedulers",2010 31st IEEE Real-Time Systems Symposium.

[13] A.Thekkilakattil, R.Dobrin, S.Punnekkat "Towards Preemption Control Using CPU Frequency Scaling in Sporadic Task Systems", 2011 IEEE.

[14] D.Alcaide and L.Pablo',"On Scheduling Models: An Overview",IEEE 2009

[15] J. Grobler, A.Engelbrecht ,"A scheduling-specific modeling approach for real world scheduling",IEEE 2007

[16] R.Shanmugapriya, S. Padmavathi,Dr.S.Shalinie ,"Contention Awareness In Task Scheduling Using Tabu Search",2009 IEEE International Advance Computing Conference (IACC 2009).

[17] Kaing Hawg "Advanced Computer Archintecture".

[18] David. E. Simon "An Emedded Software Primer".

[19] Huang, Yen, Chiang, Chang, Chang "The Five Modes AES Applications in Sounds and Images", 2010 Sixth International Conference on Informational Conference On Information Assurance and Security.

[20] Wang, Su, Horng, C.Wu, and Chih-Tsun Huang,"Single- and Multi-core Configurable AES Architectures for Flexible Security", Vol. 18, No. 4, April 2010 Ieee Transactions On Very Large Scale Integration (Vlsi) Systems

[21] http://www.xilinx.com

