Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 44 No. 8
August 2023

Dynamic Scheduling Procedure in Reconfigurable Architecture for Multicore Environment

Ashish Subhashrao Bhopale
Dept. of Electronics and Telecommunication Engineering
Prof. Ram Meghe Institute of Technology and Research
Badnera-Amravati, India
profashishbhopale@gmail.com

Dr. Archana O. Vyas
Dept. of Electronics and Telecommuinication Engineering
G. H. Raisoni University, Amravati
Anjangaon Bari Road, Amravati, India
nyasaarchana@gmail.com

Abstract

Multiple tasks arrive at the node which are processed speedily to attend the data packets arriving from the multiple nodes and hence extend the required services. To handle multiple data packets, multiple tasks are executed at the same time to speed up the process. On the hardware side each node is having limited resources on which multiple tasks uses these hardware resources on time sharing basis. Through this research paper, a new technique for handling the multiple tasks concurrently is disclosed. The proposed technique is implemented using the reconfigurable architecture which is famous for concurrent hardware execution. The algorithm is described using high speed integrated circuit hardware description language. The description is targeted to the modern concurrent programmable hardware architecture. The hardware description is performed using the Xilinx Vivado High Level Synthesis (HLS) Tool.

Keywords— HDL, FPGA, Scheduling Algorithm, Xilinx Vivado, HLS, CPLD.

I. INTRODUCTION

The first-generation processors were compatible of executing formal arithmetic and logical functions. But with the development of the modern applications the requirement of the hardware resources has been increased dramatically. The modern applications include Wireless Sensor Network (WSN), Internet of Things (IoT), Graphics Processing, Signal Processing, and many other hardware intensive applications. To fulfill the need of the hardware resources, compact processor architectures are designed which helps to cope up with the listed modern applications. Another version of the processor architecture is the Application Specific Integrated Circuit (ASIC). These are the dedicated processor architectures which performs dedicated tasks. Some of the listed applications where ASICs are employed are integrated circuit used in the voice recorder, walky-talky and many other similar applications dedicated applications. Advantage of such dedicated processing ability is that they are very tightly optimized with respect to the time, power, area and speed.

Considering this versatility, a novel technique for task handling is proposed through this research paper. Distinct approach for task handling is proposed for multi-core processing environment. The hardware implementation is carried out using the high computing field programmable gate arrays. FPGA are the prototyping hardware tool before the ASIC fabrication. The implementation is initiated through function verification which is performed using the Xilinx Vivado High Level Synthesis (HLS) Tool. Soon after, the design is elaborated and finally synthesis is done. To get the analytical outcome, the design is implemented through place, map and route steps. For describing the hardware of the proposed procedure, hardware description language (HDL) is used.

II. LITERATURE REVIEW

Complex time disseminated systems are employed to handle the physical procedure in critical applications like microcontrollers in nuclear power plants and aviation systems. In such systems, the tasks must strictly meet the deadline. The authors proposed a new algorithm for real-time scheduling tasks in a disseminated atmosphere. When the job is transferred to the node point, the scheduling program attempts to arrange the job to meet the deadlines. If such a condition is not possible, it attempts to situate a different node point where this can be completed with a much higher possibility of achievement. By utilizing an amalgamation of gossiping and propagation, all the node points in the system notify one another about their status. The practical performance of the proposed algorithm is computed by simulation as well as analytical way. [1]

The stream representation of synchronized programming is extremely admired, and it is sustained by many of the processing systems. In this representation, an individual application has its separate area. Context toggle between applications is frequently a costly procedure. A function might be consisting of multiple synchronized tasks as dissimilar jobs belonging to the identical function share space location and additional resources. In this paper, the authors [2] developed a bandwidth distribution server that employed a scheduling approach for multi-job actual instant submissions that gives the dual characteristics of presentation assurance and inner submission separation. The main objective of the proposed algorithm is to enhance the actual instant presentation.

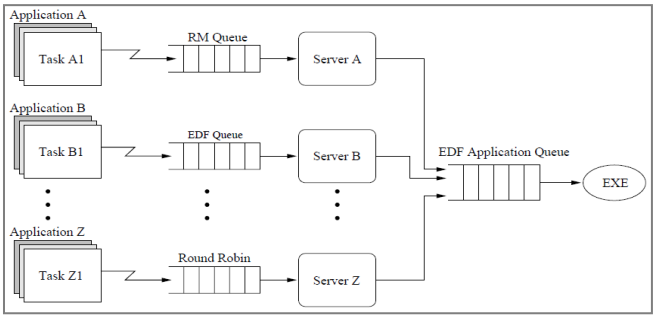

Fig. 1. Proposed System Structure

In any actual-time system, the probabilistic algorithm is the most important algorithm well known as a myopic algorithm. The characteristic function of the myopic algorithm is greatly depending on probabilistic functions. In this paper, the authors have proposed an enhanced probabilistic algorithm, which contains some new probabilistic functions. The proposed algorithm has three main functions the resource requirements, strict deadlines of the tasks and the computation time of the task. The most significant benchmark for actual instant scheduling algorithms is scheduling achievement proportion. The authors [3] have done widespread simulation analysis to calculate the efficiency of the enhanced algorithm. The outputs show that the scheduling achievement proportion of the enhanced probabilistic algorithm is better than that of a myopic algorithm.

Nowadays, the convenience and reliability of machine learning technologies have been evolved very rapidly. Specific information-based systems have been established by the ever-emergent numbers of business applications employing machine learning. In the field of actual instant control systems like robotics, process control, etc. machine learning techniques have been evolved as an interesting field, as they come out as a guaranteed methodology to deal with the growing complication of the systems to be handled. The collection of non-conventional techniques in actual instant systems is one of the critical moments. The authors [4] proposed a task algorithm that permits the demonstration of behavior with voluntary sections and numerous scheduling algorithms to integrate them into actual instant systems.

In past, the number of optimal scheduling algorithms for actual time models have been developed. On the other hand, some of the new research show that even if the best possible algorithms can program some practicable job set, sub best possible algorithms generally achieve better results when computed on actual evaluation systems. The authors have analyzed the overall best possible multiprocessor actual instant scheduling algorithms based on the conception of equality. In this paper, the authors have proposed a novel unfair scheduling algorithm based on the earliest deadline first which are having the characteristics of equality. The experimental outputs obtained gave confidence to the authors [5] as they show that the proposed scheduling algorithms generated only single migration and preemption per task during the schedule.

The actual time processing systems contain a dedicated processor for job scheduling that never be unsuccessful. As the dedicated processor gets unsuccessful, the entire processor system remains unsuccessful. To get rid of this trouble, the authors have proposed an error-tolerant allocation algorithm that is based on the shifting double token technique. The function of the priority processor is to hold priority tokens and distribute them to the jobs. The backup processor holds the backup token, if the priority processor becomes unsuccessful, it does priority processor formation. Furthermore, the deadline allocation approach is employed for backup job scheduling, contrasted with heuristic allocation which permits simple execution and enhanced allocation probability. [6]

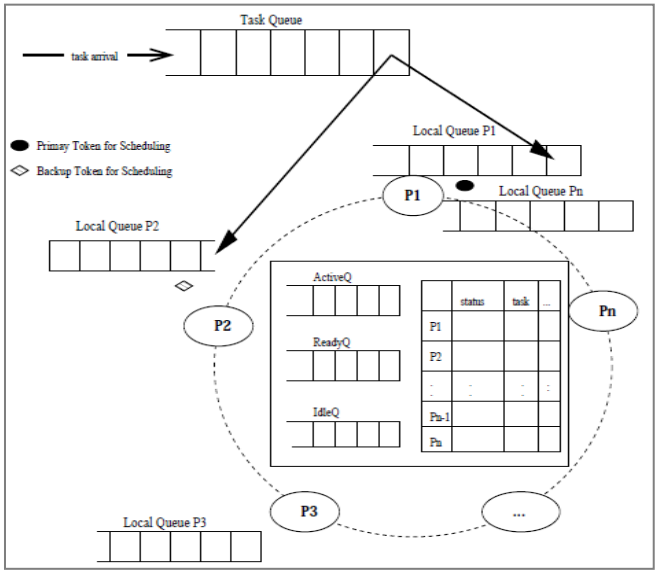

Fig. 2. Proposed System Architecture

In demanded primary networks, this paper focused on the problem of allocating periodic packets. The actual instant characteristic of the demanded primary networks may affect the periodic packets to miss their deadlines and output in the low allocation of periodic packets. In this paper, the authors proposed a novel packet allocation algorithm to implement a primary-based preventative packet broadcast on the framework base. The proposed scheduling algorithm transmits the primary packet for whole nodes before a node broadcasts a periodic packet. A node can be able to broadcast its packet only when its packet is at the top of the message queue. The authors [7] have derived required and adequate circumstances for dynamic and static primary allocation to find out the schedule of periodic packets.

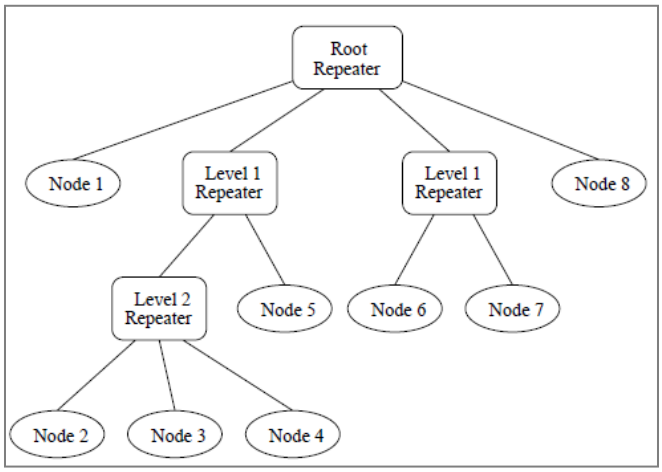

Fig. 3. Network with Multiple Repeaters

Embedded systems need a huge amount of time to construct as it is an integration of hardware with software. Integration of proper hardware and software should attain smooth coordination and this procedure needs less time. Embedded systems are manufactured to execute actual instant allocation for a variety of applications. Utilization of actual instant allocation for fabrication industrialized gives several important compensations like high throughput, precision and generally fast system reaction. In this paper, the authors have developed a chair manufacturing algorithm that provides a

solution for chair manufacturing by several employees. The main objective is to make the legs, backrest, and seat of the chair and together these parts within a minimum period. The proposed algorithm uses the processors to attain minimum delay, no escaping deadline and high manufacturing standards with the low market period. [8]

In dispersed actual instant systems, the point-to-point delay is the most essential timing restraint especially in the field of the internet of things and wireless sensor networks. Systems may require gathering information from sensors and responding instantaneously. Therefore, jobs should be computed in a space restrained approach. In dispersed actual instant systems, communication is decayed into a collection of instructions, as the space amongst two successive computations of job can be twice of its period. In dispersed actual instant systems, an additional point-to-point delay may occur as a computation would not be forever ready in a period. [9]

Conventional actual time systems have mostly ignored the utilization of disks due to their impulsiveness and comparative sluggish speed. Numerous actual-time applications advantage considerably from the utilization of disks to access and accumulate actual-time information. The authors [10] analyzed the difficulty of achieving assured timely admittance to files on a disk in an actual instant system. The authors employed two disk scheduling algorithms as just in time scheduling and the earliest deadline scheduling algorithm. One more algorithm is proposed to enhance the disk performance that could be upset when an actual instant allocation algorithm such as the earliest deadline first is applied immediately. Admission handled approaches with basically satisfactory characteristics of presentation and usability are presented.

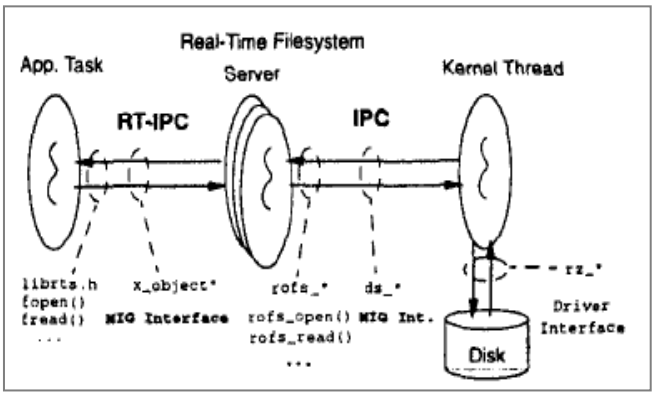

Fig. 4. File System Service Layers

III. Proposed Scheduling Algorithm

To demonstrate the performance of the proposed system, eight tasks are assumed to be arriving at the processor node for task execution. These right tasks are divided into two groups, the task one to task 4 are configured into top tasks and task 5 to task 8 are configured into bottom tasks. Each of the tasks is having their own priority and are executed accordingly. When no task from top or bottom group is under execution then routine task will be executed which indicates processor routine executions. It is assumed that the routine task takes 5 clock cycles of time duration for completion of one phase. After completion of the first phase second phase of the routine task is executed and this continues until the higher priority tasks invoke interrupt service signal. Once the high priority signals are inserted, with immediate from the next clock cycle the routine task execution is suspended and corresponding higher priority tasks are serviced. The subsequent figure discloses the systematic flow of the proposed system.

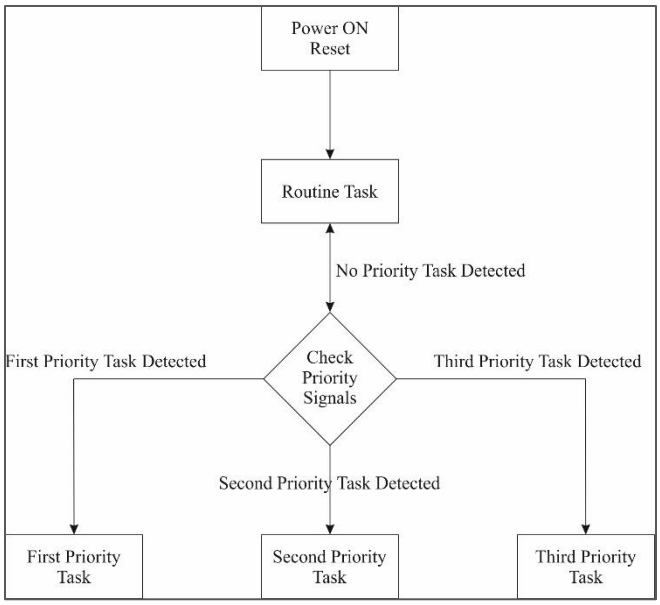

Fig. 5. Systematic Flow Chart

Once the higher priority tasks complete the execution, the routine task is re-initiated from the moment where it was suspended. For effective demonstration of the said system, it is assumed that the routine task takes 5 clock cycle of duration for execution while the higher priority tasks need 10 clock cycles for execution. As depicted, the power on reset condition indicates the default situation of the procedure in which the procedure is set to operate. This has been indicated through the figure Fig.6.

The simulation period from 0 ns to 1000 ns indicates the uninitialized state of the system. The simulation period from 1000ns to 1200ns indicates power on reset conditions. At the 1200^{th} ns the power on reset is deactivated. The clock signal which is synchronizing all the components in the system is initialized from the 1000^{th} ns moment. The subsequent figure fig.6 depicts the above details.

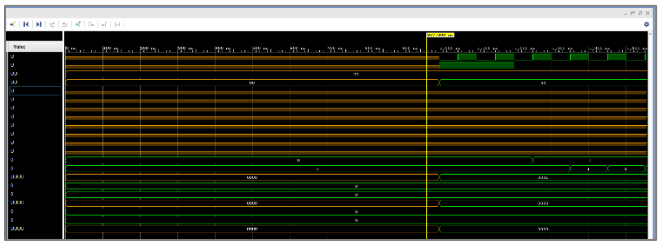

Fig. 6. Uninitialized and Power On Reset Activation and Deactivation

After deactivation of the power on reset condition at simulation period 1200ns, the system check for the higher priority tasks seeking services, since the no request signal is identified, the system initiates routine task execution from 1300ns simulation period. Since the routine task need 5 clock cycles for completion of the task, it completes the first phase

of the routine task at 1800ns simulation period and initiates second phase of the routine task execution and completes at 2300ns simulation period. In this execution period, the system continuously observes the request signal from the higher priority tasks. This is depicted through the figure Fig.7.

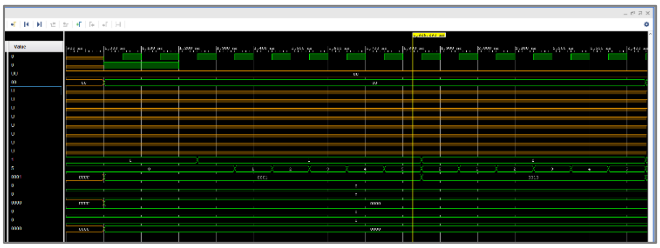

Fig. 7. Execution of First Phase and Second Phase of Routine Task

The multiple phases of the routine tasks are executed repeatedly. At the simulation moment of 3900ns the higher priority task from the top group is identified. Upon detection of the request signal, the routine task is suspended at 4000ns simulation period and the higher priority tasks are executed from the 4100ns simulation period. This situation is depicted through the figure Fig.8. The higher priority task completes the execution and at 5000ns simulation period, upon which the suspended routine task is re-initiated from the 5100ns simulation period.

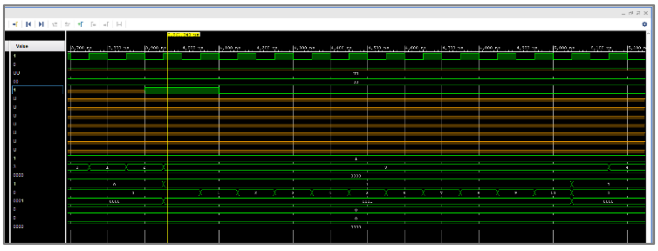

Fig. 8. Suspension of Routine Task and Activation of Higher Priority Tasks

The current phase of the routine task is completed at the simulation period 5300ns and next phase of the routine task execution is initiated at the 5400ns duration. But, at 5500ns moment two of the higher priority task form the top group itself, inserted service request signal in response to which system assigns the priority and executes the higher priority task from 5600ns to 6600ns and immediately then second priority task is executed from 6600ns to 7600ns duration. This is depicted through the figure Fig. 9 and Fig.10 below, respectively.

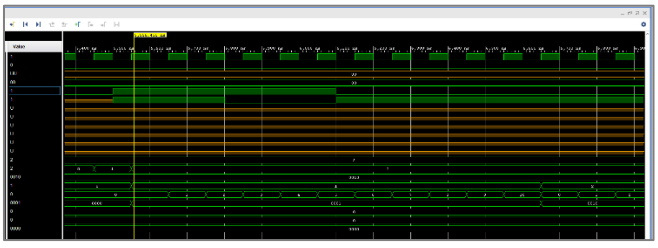

Fig. 9. Insertion and Completion of First Higher Priority Task

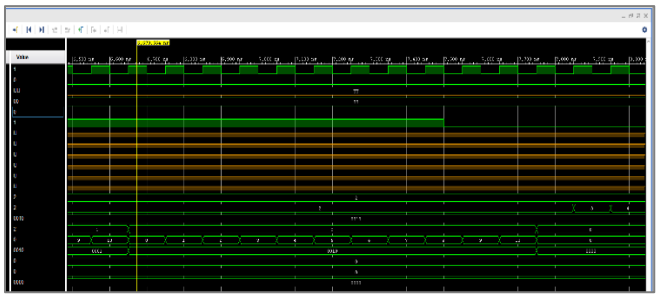

Fig. 10. Insertion and Completion of Second Higher Priority Task

On completion of all of the higher priority tasks, the routine tasks are executed from the moment where it was suspended from 7800ns duration onwards.

While completing another phase of the routine task, two higher priority tasks from top and bottom group of the tasks insert service request signal at 8200ns simulation period, in response to which the routine task is suspended at 8300ns duration and both higher priority tasks are executed concurrently from the 8400ns duration. This is depicted through the following figure Fig. 11.

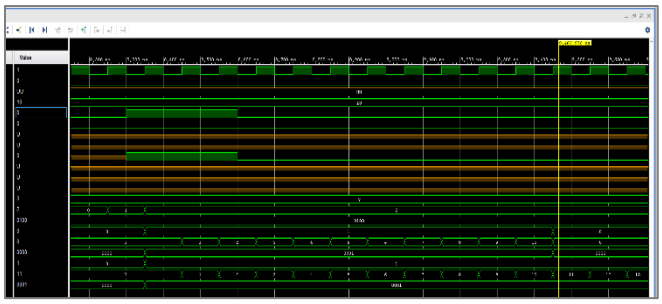

Fig. 11. Handling of Both Priority Tasks Concurrently

Both higher priority tasks complete the execution at 9400ns moment and again the routine task execution is proceeded from the 9500ns simulation period.

Above two examples of the task handling demonstrated the task handling on priority basis and concurrency basis. The second case study, since the tasks are executed concurrently, a multicore environment is created which is then targeted to the virtex-7 series field programmable gate array device. The Virtex-7 series xc7v585tffg1157-1 device is specifically targeted.

The hardware description language which is used for describing the hardware architecture, was initially designed for hardware simulation purpose, but later it is also used for synthesis purpose. Synthesis means hardware description using the software coding. Since the HDL was initially designed for simulation purposes only, all the statements that means sentences are not synthesizable. Which means they don’t have direct hardware meaning. RTL schematic view confirms the perfect hardware conversion of the HDL constructs. While performing the schematic RTL analysis using the Xilinx Vivado HLS tool following RTL schematics are recorded.

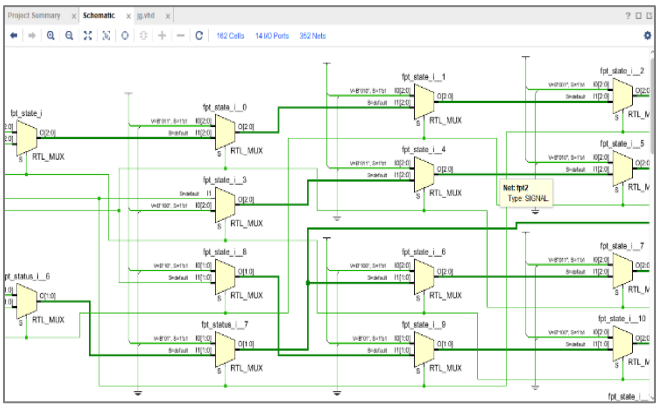

Fig. 12. RTL Schematic View (a)

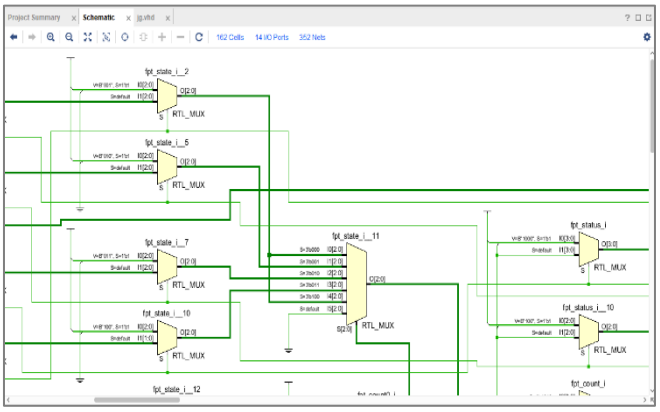

Fig. 13. RTL Schematic View (b)

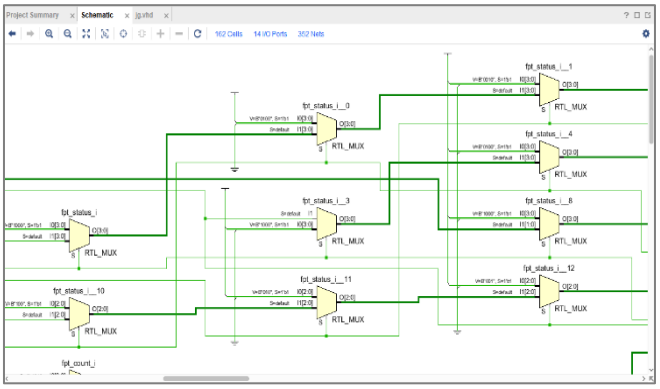

Fig. 14. RTL Schematic View (c)

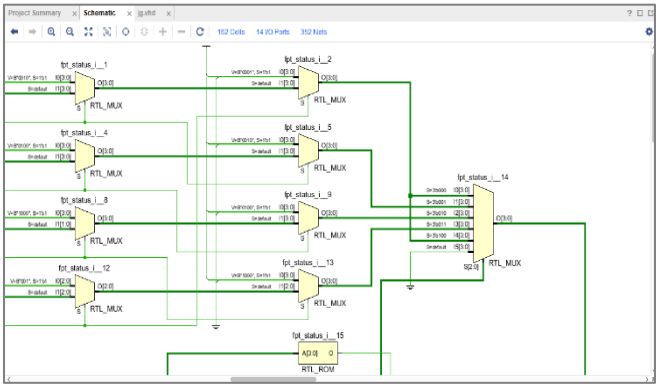

Fig. 15. RTL Schematic View (d)

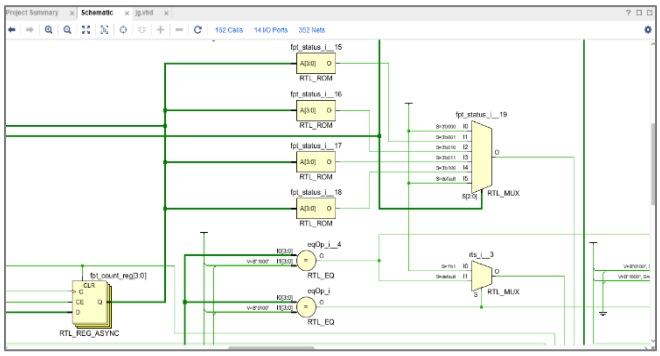

Fig. 16. RTL Schematic View (e)

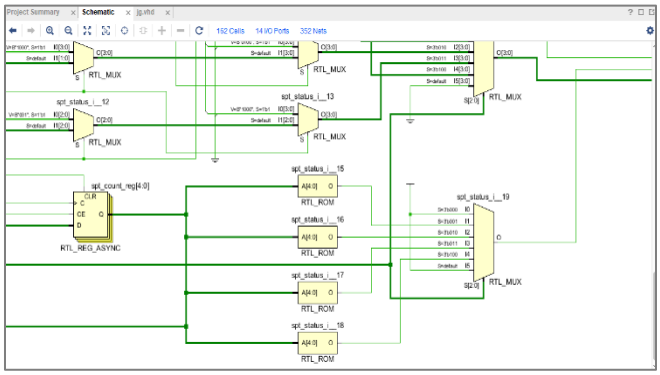

Fig. 17. RTL Schematic View (f)

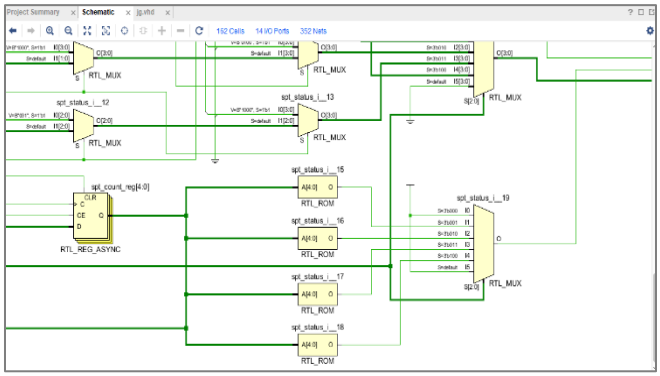

Fig. 18. RTL Schematic View (g)

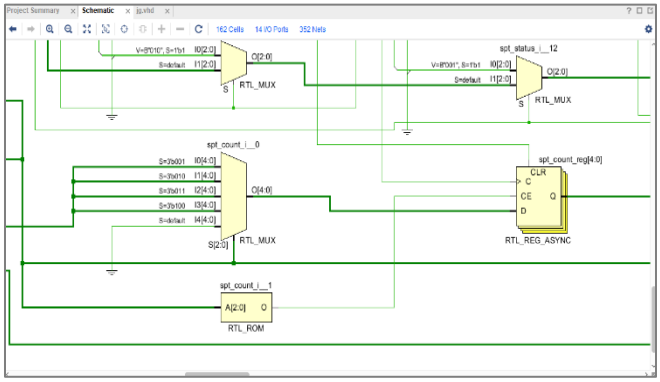

Fig. 19. RTL Schematic View (h)

Finally, the synthesis process is carried out, in which the implementation of the proposed architecture into the targeted FPGA device is carried out in three steps of Place, Map and

Route. In these three steps, the software to hardware converted components are virtually placed on the device with different possibilities to meet the highest possible level of optimizations. Once the higher level of optimization is found then the hardware components are mapped and interconnected together which is also called as routing. Upon final implementation of the design, different parameters like power utilization, area utilization in terms of hardware complexity, time utilization which indicates minimum and maximum path delays and speed of operations are recorded. These parameters are disclosed through the subsequent table.

Table 1: Statistical Analysis of Synthesis Outcomes

Sr. No.	Device	Parameter	Units
1.	Xilinx Virtex – 7	Estimated Frequency	416.84035 MHz
2.		Estimated Period	2.399 ns
3.		Total Complexity	117
4.		Power	0.002 W

IV. CONCLUSION

Scheduling procedure for active task using the reconfigurable environment is proposed through the paper. In this assumptions, eight priority tasks are bifurcated into two groups each are having four priority tasks, along with the routine task is considered for implementation. In the proposed system, routine task is the default task which reconfigurable device is executing and at the same moment, the higher priority tasks are also checked for their activeness. If multiple priority tasks are engaged then routine task is suspended immediately and first priority task will be executed then after completion of the first priority task, second priority task will be executed and after completion of the second priority task routine task will be executed. Once all the priority tasks are completed, then routine task execution is engaged. In the second phase of the execution, concurrency of task execution is demonstrated which also demonstrate the multi core environment where multiple tasks are executed simultaneously. The outcomes are indicated through the table 1. As indicated, estimated frequency of 416.84035 MHz is recorded which assures 2.399 ns of maximum delay at the cost of 0.002W of energy consumption.

REFERENCES

[1] A. Bestavros and D. Spartiotis, "Probabilistic job scheduling for distributed real-time applications," [1993] Proceedings of the IEEE Workshop on Real-Time Applications, 1993, pp. 97-101, doi: 10.1109/RTA.1993.263108.

[2] G. Lipari and S. K. Baruah, "Efficient scheduling of real-time multi-task applications in dynamic systems," Proceedings Sixth IEEE Real-Time Technology and Applications Symposium. RTAS 2000, 2000, pp. 166-175, doi: 10.1109/RTTAS.2000.852461.

[3] Zhu Xiangbin and Tu Shiliang, "An improved dynamic scheduling algorithm for multiprocessor real-time systems," Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2003, pp. 710-714, doi: 10.1109/PDCAT.2003.1236397.

[4] H. Hassan, A. Crespo and A. Garcia, "Scheduling algorithms for improving the response in intelligent real-time environments," Proceedings Seventh Euromicro Workshop on Real-Time Systems, 1995, pp. 93-98, doi: 10.1109/EMWRTS.1995.514298.

[5] G. Nelissen, V. Berten, J. Goossens and D. Milojevic, "Reducing Preemptions and Migrations in Real-Time Multiprocessor Scheduling Algorithms by Releasing the Fairness," 2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications, 2011, pp. 15-24, doi: 10.1109/RTCSA.2011.57.

[6] Sungyoung Lee, S. K. Oh and Chul Hee Woo, "Dual-token-based fault-tolerant scheduling for hard real-time multiprocessor systems," Proceedings Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No.98EX236), 1998, pp. 232-238, doi: 10.1109/RTCSA.1998.726423.

[7] Taewoong Kim, Heonshik Shin and Naehyuck Chang, "Scheduling algorithm for hard real-time communication in demand priority network," Proceeding. 10th EUROMICRO Workshop on Real-Time Systems (Cat. No.98EX168), 1998, pp. 45-52, doi: 10.1109/EMWRTS.1998.685067.

[8] V. S. Vora and A. K. Somkuwar, "An advanced approach for implementation of real time scheduling algorithm for efficient mass production," 2013 International Conference on Intelligent Systems and Signal Processing (ISSP), 2013, pp. 224-227, doi: 10.1109/ISSP.2013.6526907.

[9] C. Chuang, Y. Chen and C. Hsueh, "Scheduling Low-Utilized Real-Time Systems with End-to-End Timing Constraints," 2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2016, pp. 98-98, doi: 10.1109/RTCSA.2016.53.

[10] A. Molano, K. Juvva and R. Rajkumar, "Real-time filesystems. Guaranteeing timing constraints for disk accesses in RT-Mach," Proceedings Real-Time Systems Symposium, 1997, pp. 155-165, doi: 10.1109/REAL.1997.641278.