Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 44 No. 8
August 2023

Synthesis and Analysis of Active Scheduling Procedures Targeting Reconfigurable 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— In the computing environment, multiple data packets arrive at the node for processing. Multiple tasks are engaged to handle these data packets which accepts these data packets process them and executes further required steps. For handling the multiple data packets simultaneously, multiple tasks are engaged which utilizes the limited resources on time sharing basis. To make the limited resources useful and handle the multiple tasks efficiently, strong, highly active task scheduling algorithm is required. This research paper demonstrates the multiple tasks arriving and getting executed simultaneously in addition to the routing tasks. 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 early generation processors are designed to perform basic operations of arithmetic and logical operations. These basic operations are then used to control different applications. From small to medium complex applications these processor environments perform better but with the increase in the complexity these processor fails to perform well. From this step modern processors are designed which is tightly designed architecture in which different required components like ADCs, signal conditioners, DAC and other significant components are pre-built. Such processors are well suitable for compact and medium to high applications requirements. But on the other side of the coin, to the applications where random pattern of data arrives and to be processed and produced randomly in size and pattern, these processor fails to perform. In such applications, programmable architectures are performing well.

Programmable architectures are the microelectronic components which can be reconfigured according to the need of the applications or according to the user requirements. Most popular programmable logic devices are the Complex Programmable Logic Device (CPLD) and the Field Programmable Gate Array (FPGA). These are the magical devices with which it is possible to design highly customized architecture and deploy them into the applications. The advantage that we get with deployment of these devices is that since the hardware is custom designed, it uses highly optimized power, time, area, and speed. We demonstrate use of this technology for the implementation of the scheduling algorithm in which multiple tasks are operated individually and in concurrent mode. Further, AMD-Xilinx programmable devices are widely used in distinct applications, we make use of the Xilinx High Level Synthesis (Xilinx-HLS) Tool and the proposed algorithm is targeted to the latest programmable device.

II. Previously Cited Technology

Several scheduling prototypes have been available for the actual time communication and actual-time applications. The most general types of scheduling prototypes contain allocation determined, priority determined, and time determined algorithms. In this paper, the authors [1] have proposed a normal scheduling technique that is employed to incorporate these prototypes in a single structure. Allocator and correspondent are employed as scheduler elements in the proposed technique. For every individual task, the structure recognized four scheduling characteristics as resources, priority, start instant and end instant. The authors show that the proposed structure can be utilized to effectively compute several scheduling algorithms.

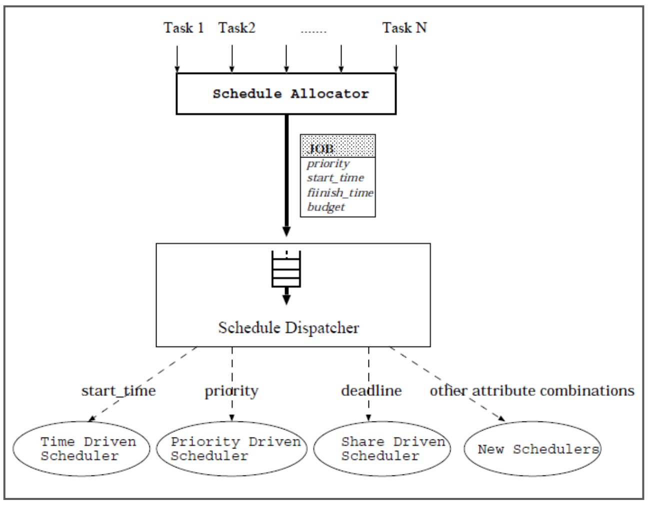
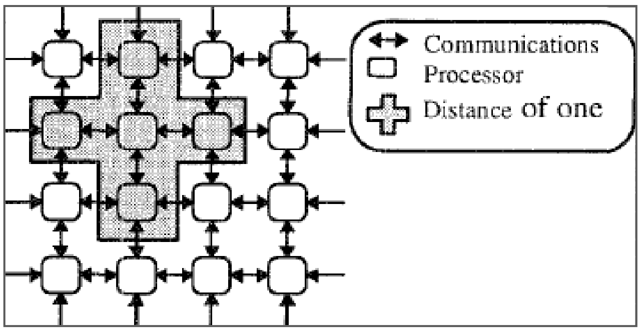


Fig. 1. Architecture of the scheduling algorithm

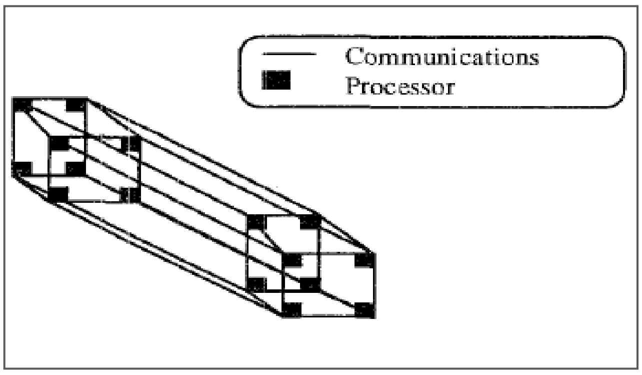
Error tolerance plays an essential role in real-time processing unit systems, as the timing constrictions should not be interrupted. For a real instant processor atmosphere, two conventional queues-based scheduling algorithms as linear time heuristics and possible shortest pathway are available. The linear time heuristics algorithm thoroughly approximates the best possible algorithm. The possible shortest pathway algorithm can give the best possible error tolerant schedules, but it is not practically applicable due to its time complication. The possible shortest pathway algorithms work on the assumption that there is at least a single error present within the instant interval of ’t’. The authors [2] proposed an enhanced shortest possible pathway algorithm on the supposition that there will be no further error through the lowest inter-error instant 't' following a single fault occurs. The proposed technique enhanced system presentation by employing additional main jobs in an error tolerant program and decreases the time complication in backup programs.

The central processing unit has been extensively employed for handling actual time procedures like manufacturing processing control systems. The utilization of central processing units for controlling actual instant instructions has speedily developed. The wide utilization of central processing systems to handle time complex instructions that have strict deadlines necessitates the utilization of simple and effective scheduling algorithms. In this paper, the authors [3] have taken into consideration the difficulty of periodic job scheduling in an actual instant atmosphere. The authors presented a preventative and non-preventative scheduling algorithm. A preventative scheduling algorithm is employed to evade needless preemption.

Nowadays, Parallel genetic algorithms are employed more than traditional genetic algorithms as they provide a faster solution to a wide range of problems. In the actual scenario, the hardware structure of the parallel genetic algorithm can deal with such actual instant problems. An appropriate structure for image processing parallel genetic algorithm was developed based on the traditional crossover method. Numerous problems are more matched to combinational handlers such as organize based crossover. In this paper, the authors [4] have proposed a novel hardware structure based parallel genetic algorithm employing organized based crossover which can minimize an innovative group of actual time combinational difficulties. The disk scheduling technique has been recognized as a general actual instant minimization problem to determine the advantages of the proposed hardware structure.



(a)



(b)

Fig. 2. Grid (a) and hypercube (b) topology

In real-time systems, a preprocessing period branch and bound implicit inventory algorithm endeavors to locate a possible allocation for a set of hard actual instant procedures. Procedures are supposed to be passively allocated to multiprocessors on a multimedia node. When it is contrasted with the handcrafted allocating techniques, the submission of this preprocessing scheduling algorithm to hard actual instant systems must decrease the resources needed for the processing period scheduling and context switching. The minimization standard is to optimize procedure tardiness defined as the dissimilarity between the procedure completion period and deadline. In this paper, the authors [5] show that the algorithm always doesn't be successful in obtaining the best possible output.

In modern disseminated actual instant applications require vibrant ad adaptable scheduling algorithms to give enduring assurance to application things. In this paper, the authors have proposed a novel scheduling algorithm that makes the use of task negligence and the things important to take efficient scheduling judgments. The proposed scheduling algorithm utilizes actual instances and resources to identify the practicability of the jobs and to allocate the things to the microprocessors. Jobs instances characteristic and tolerance assessment report is taken from processor to processor; acquiescent a system extensive scheduling approach that necessitates restricted calculations. The main objective of the proposed algorithm is to guarantee that a low-priority task doesn't interrupt the implementation of a high-priority task. [6]

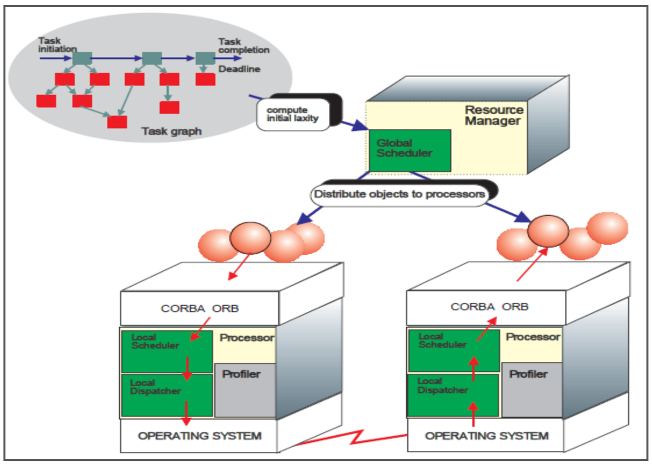


Fig. 3. The scheduling algorithm

Several times complicated applications need conventional functions, and the jobs of these applications must have to meet their deadlines. A deadline failure to attend can be disastrous for jobs with strict deadlines. For quality of service degradable or soft actual time jobs, instantly estimated outputs of degraded quality and intermittent deadlines are adequate. The authors proposed a new dynamic scheduling algorithm for incorporated scheduling for actual instant jobs in multiprocessor systems. The most important goal of the proposed algorithm is to enhance the arrangement of jobs by making the utilization of the characteristics of these models in quality-of-service degradation. The proposed algorithm shows how it can be accepted incorporated scheduling for multiprocessor systems and strict actual instant jobs and performance of their efficiency in service of quality degradation. [7]

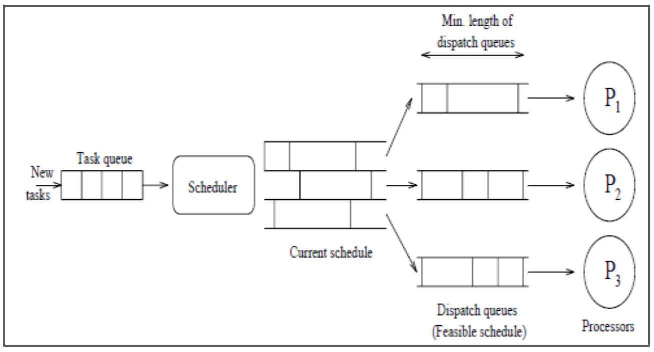

Min. length of dispatch queues
New tasks
Task queue
Scheduler
Current schedule
Dispatch queues (Feasible schedule)
P1
P2
P3
Processors

Fig. 4. The scheduling model

The authors [8] presented an incorporated structure for the identification of non-predictable workloads, triggered by the identification of faults in an actual instant system. The structure is specifically constructed for the execution of predetermined priority preventative systems. A detailed investigation for recuperation workloads is done by constructing a standard for receptiveness of error recuperation executions. This encourages the authors to validate the timing exactness of actual time workloads under momentary recuperation workloads and give elegant deprivation to the actual time workload at the time of recuperation. By provisionally removing the low priority jobs, the receptive algorithm constructed by the authors is predictable to service the jobs with the highest priorities without damaging low priority jobs.

An efficient time redundancy technique for accomplishing error forbearance in actual time system when gap redundancy can't be employed for mass restraints. In this paper, the authors [9] have proposed an error-tolerant scheduling algorithm for actual instant systems consisting of strict and rigid periodic jobs. According to the priority, rigid jobs can be normally missed out on only one occasion according to a predefined quality of service characteristic. While strict jobs have the highest priority. The proposed algorithm guarantees that every job occurrence is fulfilled within its instant restraints by the priority task. The algorithm increases the utilization of processor inactive time by executing the highest priority tasks and mechanically retrieves the auxiliary time conserved by de-allocating support tasks.

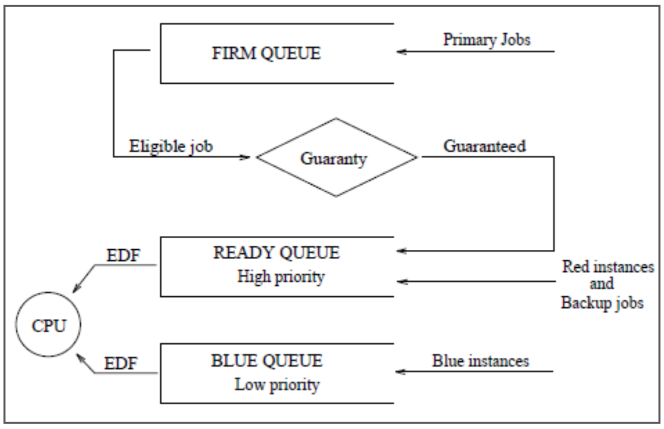


Fig. 5. Error tolerant algorithm

Enhance processing units are more proficient to evaluate parallel applications, several applications might be organized by some parallel jobs. Several applications might need timing restraints to make use of the preferred presentation. These restraints are available in control systems as hard tasks and multiprocessing systems as soft tasks. In this paper, the authors analyzed the trouble of giving separation and actual instant computation in preparation of multi-string applications on a particular processor. The proposed algorithm can be employed in a wide range of applications such as in a network association, wherever dissimilar streams of messages might need a convinced stage of function, or in multiprocessing actual instant systems, in which applications might need an assurance of the quality of service. The proposed scheduling algorithm is especially appropriate for soft actual instant and multiprocessing atmospheres as it doesn't need the accurate information of the tasks times and inter appearance times. [10]

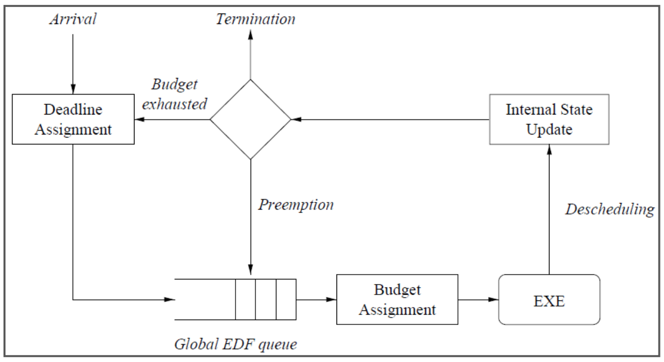


Fig. 6. Bandwidth sharing server

III. Proposed Scheduling Algorithm

Effectiveness of the proposed technique is demonstrated by assuming the traffic light signal. In this case, traffic may arrive from four possible directions like East, West, North and South. Apart from having the routine traffic, there is possibility of traffic arriving for the last journey, traffic arriving with the VIP person and possibility of the sudden ambulance appearance. Considering this assumption, highest priority must be given to the ambulance, the second priority must be given to the last journey and third priority must be given to the VIP vehicle and when no other priority vehicles or traffic is appearing, in that case, routine traffic must be handled. Since the module is generic in nature open loop nature of the module is discussed by assuming that the routine traffic needs 5 clock cycles for handling, that means to handle the routine task we need 5 clock cycles. The time required to handle the priority task that is the ambulance is 5 clock cycles that means, module takes 5 clock cycles to complete the first priority task. The second priority task that is the time required to clear the traffic arising from the last journey is 10 clock cycles. That means to execute the second priority task module takes 10 clock cycles duration. On the other hand, to handle the traffic arising through the VIP vehicles takes 15 clock cycles of time. That means to execute the third priority task module takes 15 clock cycles of time. In these assumptions, the routine task of the module is indicated as the *"rt"*, the first priority task is indicated as the *"fpt"*, the second priority task is indicated as the *"spt"*, and the third priority task is indicated as the *"tpt"*. The subsequent figure Fig.7 depicts the systematic flow of the proposed procedure.

As indicated, 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.8.

The 1000ns to 1100 ns simulation period indicates the power on reset condition in the fig.8. Initially the routine task *"rt"* is operated in which the routine traffic at the traffic signal is routed efficiently. At the same moment, the priority traffic is also checked concurrently. If no priority traffic is detected, then the routine traffic is set to operate with 5 clock cycles of time in each place. This condition is depicted through the figure Fig.9. After deactivating the power on reset conditions. At 1100 ns, first the higher priority traffic signals are identified at 1200ns. Since no higher priority traffic interrupts are identified, the routine task is activated at 1300 ns. The routine task continues to operate in repeated mode and at the same moment also check the generation of higher priority interrupt signals.

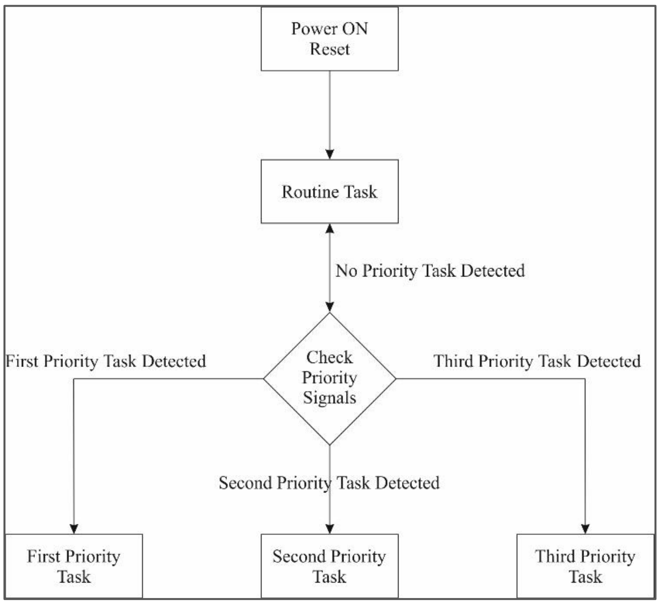

Power ON Reset
Routine Task
No Priority Task Detected
First Priority Task Detected
Check Priority Signals
Third Priority Task Detected
Second Priority Task Detected
First Priority Task
Second Priority Task
Third Priority Task

Fig. 7. Systematic Flow Chart

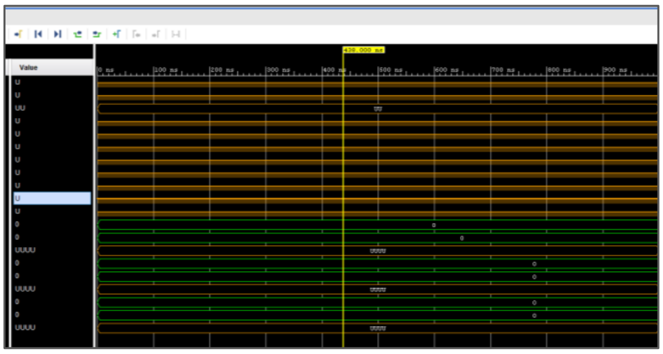


Fig. 8. Power On Reset Condition

As soon as the higher priority traffic is identified, the routine traffic moment that is routine task is suspended and the procedure is switched to operate the respective higher priority traffic that means the higher priority task. This moment is indicated through the figure Fig.9 below.

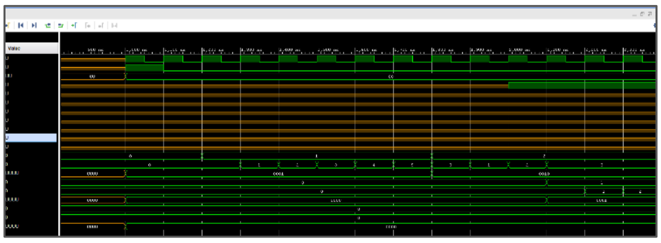


Fig. 9. Execution of Routine Traffic Task and Higher Priority Traffic Insertions

As shown in the figure above, since no higher priority task has been detected till the moment 1200ns, routine task is deployed for execution from 1300ns. At 1800ns of simulation period the first phase of the routine task is completed and since no higher priority task is active, the second phase of the routine task is initiated. While the second phase of the routine task is under process from 1800ns, the first priority task is inserted at 2000 ns of simulation period and accordingly, the routine task is suspended at 2100 ns and first priority task is serviced from 2200 ns of simulation period. Once the first priority task completes at 3200 ns of simulation period, the routine task which was suspended due to insertion of the first priority task *"fpt"*, is reinitiated. This is depicted through the following figure Fig. 10.

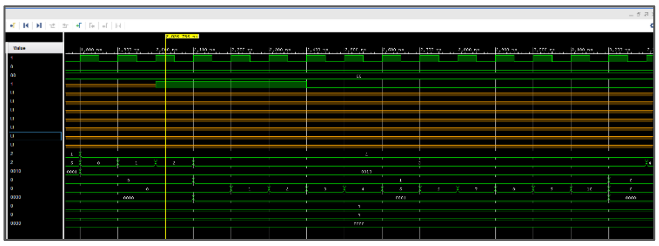


Fig. 10. Completion of First Priority Task and Re-Initiation of Suspended Task

Accordingly, the second priority or third priority tasks are operated by suspending the routine task and after completion of the same again the routine task is initiated. If multiple priority tasks are activated at the same moment, then on priority first priority task is executed then second priority task is initiated and after completion of the second priority task, third priority task is initiated and finally after completion of all higher priority tasks, routine task is reinserted for execution.

While performing the synthesis process, the proposed architecture of the scheduling procedure is targeted to the virtex-7 series field programmable gate array device. The Virtex-7 series xc7v585tffg1157-1 device is specifically targeted. While performing the schematic RTL analysis using the Xilinx Vivado HLS tool following RTL schematics are recorded.

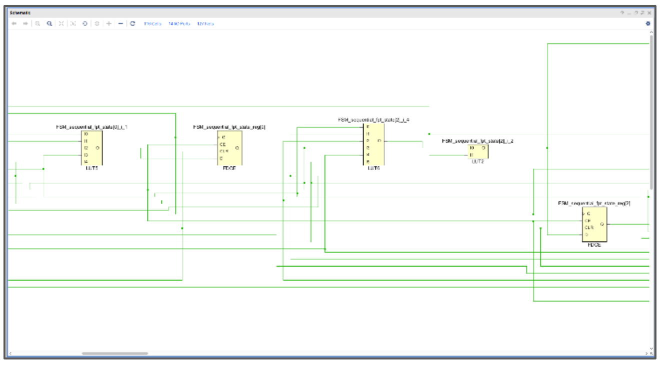


Fig. 11. RTL Schematic View (a)

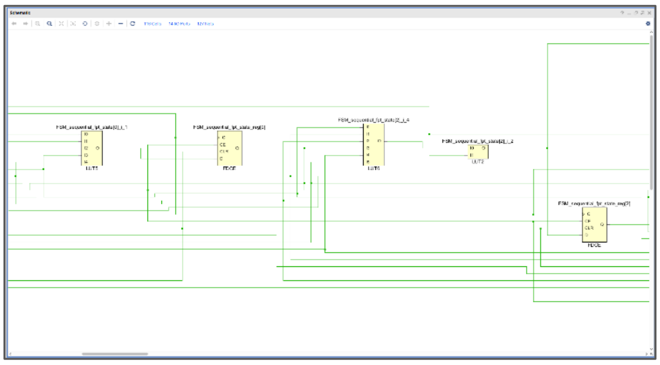


Fig. 12. RTL Schematic View (b)

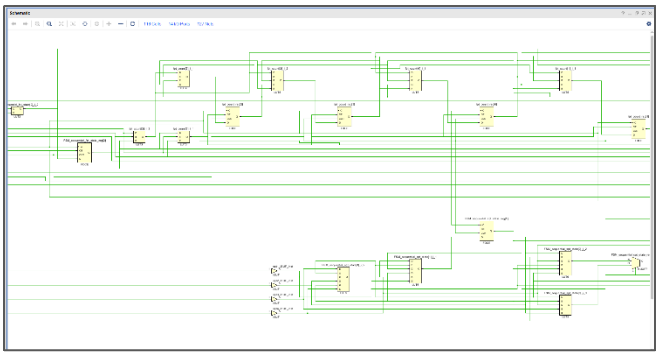


Fig. 13. RTL Schematic View (c)

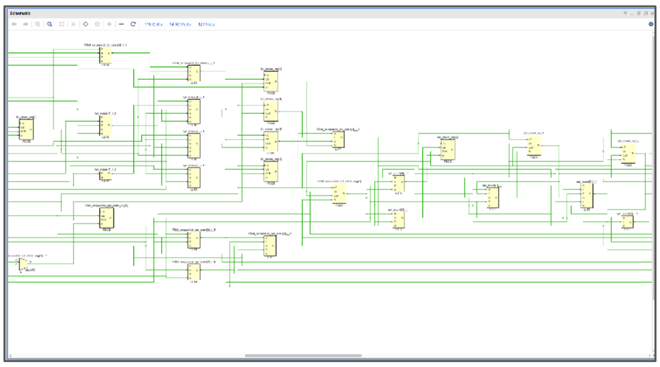


Fig. 14. RTL Schematic View (d)

The hardware description language was initially designed for hardware architecture simulations, 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 of 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.

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	305.997552 MHz
2.		Estimated Period	3.268 ns
3.		Total Complexity	169
4.		Power	0.002 W

IV. Conclusion

Scheduling procedure for active task using the reconfigurable environment is proposed through the paper. In this assumptions, three priority tasks along with the routine task is considered for implementation. For effective description of the model, traffic light signals and other different conditions are assumed. 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 third priority task will be executed. Once all the priority tasks are completed, then routine task execution is engaged. The described architecture is first time simulated then elaborated to confirm the hardware conversion and finally, the description is synthesized. The outcomes are indicated through the table 1. As indicated, estimated frequency of 305.997552 MHz is recorded which assures 3.268 ns of maximum delay at the cost of 0.002W of energy consumption.

References

[1] Y. -. Wang and K. -. Lin, "Implementing a general real-time scheduling framework in the RED-Linux real-time kernel," Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054), 1999, pp. 246-255, doi: 10.1109/REAL.1999.818850.

[2] Hyungil Kim, Sungyoug Lee and Byeong-Soo Jeong, "An improved feasible shortest path real-time fault-tolerant scheduling algorithm," Proceedings Seventh International Conference on Real-Time Computing Systems and Applications, 2000, pp. 363-367, doi: 10.1109/RTCSA.2000.896412.

[3] H. Singh, "Scheduling techniques for real-time applications consisting of periodic task sets," Proceedings of 2nd IEEE Workshop on Real-Time Applications, 1994, pp. 12-15, doi: 10.1109/RTA.1994.316133.

[4] B. C. H. Turton and T. Arslan, "A parallel genetic VLSI architecture for combinatorial real-time applications-disc scheduling," First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995, pp. 493-498, doi: 10.1049/cp:19951097.

[5] T. F. Abdelzaher and K. G. Shin, "Comment on "A pre-run-time scheduling algorithm for hard real-time systems"," in IEEE Transactions on Software Engineering, vol. 23, no. 9, pp. 599-600, Sept. 1997, doi: 10.1109/32.629495.

[6] V. Kalogeraki, P. M. Melliar-Smith and L. E. Moser, "Dynamic scheduling for soft real-time distributed object systems," Proceedings Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2000) (Cat. No. PR00607), 2000, pp. 114-121, doi: 10.1109/ISORC.2000.839518.

[7] A. Mittal, G. Manimaran and C. S. R. Murthy, "Integrated dynamic scheduling of hard and QoS degradable real-time tasks in multiprocessor systems," Proceedings Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No.98EX236), 1998, pp. 127-136, doi: 10.1109/RTCSA.1998.726408.

[8] P. Mejia-Alvarez and D. Mosse, "A responsiveness approach for scheduling fault recovery in real-time systems," Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium, 1999, pp. 4-13, doi: 10.1109/RTTAS.1999.777656.

[9] M. Caccamo and G. Buttazzo, "Optimal scheduling for fault-tolerant and firm real-time systems," Proceedings Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No.98EX236), 1998, pp. 223-231, doi: 10.1109/RTCSA.1998.726422.

[10] G. Lipari and G. Buttazzo, "Scheduling real-time multi-task applications in an open system," Proceedings of 11th Euromicro Conference on Real-Time Systems. Euromicro RTS'99, 1999, pp. 234-241, doi: 10.1109/EMRTS.1999.777470.