http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 6 (3): 338-344, 2007
ISSN 1812-5638
© 2007 Asian Network for Scientific Information

Hardware Implementation of Hybrid Dynamic Voltage Scaling

[1]R. Seshasayanan and [2]S.K. Srivatsa
[1]Department of Electronics and Communication Engineering,
College of Engineering, Anna University, India
[2]Department of Electronics Engineering,
Madras Institute of Technology, Anna University, India

Abstract: The research presented in this study addresses reducing the power consumption of processor by adjusts the clock speed and voltage dynamically for hard real time tasks. The energy efficiency of a DVS algorithm largely depends on the performance of the slack estimation method used in it. The approach uses fixed priorities assigned tasks. It targets energy consumption reduction by using both on-line and off-line decisions, taken both at task level and at task-set level. We consider sets of independent tasks running on processors with dynamic voltage and frequency scaling techniques, where every deadline has to be met. Here we consider a few sets of independent tasks running on processor with dynamic frequency and voltage scaling. By using Hybrid DVS algorithm, the power savings is obtained. The efficient inter and intra algorithms are implemented in VHDL. The simulation and synthesize report are shown. By using this approach we achieve energy reductions.

Key words: Low-energy, hard real-time, FPGA, InterDVS, IntraDVS, scheduling etc

INTRODUCTION

Low energy consumption is today an increasingly important design requirement for digital systems, with impact on cost of the system, operating speed and importance on the environment. The energy dissipation and reducing power has been addressed by many research groups, at various abstraction levels. For CMOS circuits the energy consumption E has a quadratic dependency on the supply voltage V_{DD}, lowering the supply voltage V_{DD} is an effective way of reducing energy consumption of embedded systems such as digital cellular phones and personal digital assistants. Lowering the supply voltage also decreases the maximum achievable clock speed; in the CMOS circuit, the delay T_D is given by

$$T_d \alpha V_{DD} / (V_{DD}-V_t)^{\alpha} \tag{1}$$

where V_T is the threshold voltage and α is a velocity saturation index (Leeand sakurai, 2000). This energy delay trade-off makes possible various Dynamic Voltage Scaling (DVS) and frequency scaling techniques that adjust the clock speed and the supply voltage dynamically according to the performance requirements of given tasks.

With a recent explosive growth in the portable and mobile embedded device market, where low-power consumption is an important design requirement, several commercial variable-voltage microprocessors/embedded processors (Benini and DeMicheli, 2000; Ishihara and Yasuura, 1998) were developed. Targeting these processors, many DVS algorithms have been proposed or developed, especially for hard Dynamic Voltage Scaling (DVS) is an effective low-power design technique for embedded real-time systems. In recent years, many DVS algorithms have been proposed for reducing the energy consumption of embedded hard real-time systems (Burd *et al*., 2000; Hong *et al*., 1998; Pering *et al*., 1998). Since lowering the supply voltage also decreases the maximum achievable clock speed (Locke *et al*., 1989) various DVS algorithms for hard real-time systems have the goal of reducing supply voltage dynamically to the lowest possible level while satisfying the task's timing constraints.

Although each DVS algorithm is shown to be quite effective in reducing the energy/power consumption of a target system under its own experimental scenarios, these recent DVS algorithms have not been quantitatively evaluated under a unified framework, making it a difficult task for low power embedded system developers to select an appropriate DVS algorithm for a given application/system. A quantitative analysis of the energy-efficiency is particularly important because most of these

Corresponding Author: R. Seshasayanan , Department of Electronics and Communication Engineering College of Engineering, Anna University, India Tel: 9444123528

DVS algorithms are based on both static and dynamic slack analysis techniques whose performance is difficult to predict analytically. In addition, their energy efficiency fluctuate significantly depending on the workload variations, task set characterizations and execution paths taken, further requiring a quantitative comparison study.

For hard real-time systems where tasks have stringent timing constraints, the energy-delay trade-off makes the DVS problem more challenging. When the supply voltage and clock speed are lowered for reduced energy consumption, the execution times of the tasks may increase, resulting in deadline misses. Since deadline misses in real-time systems can cause catastrophic system failures, dynamic voltage scaling can utilize only slack times (or idle times) when adjusting voltage levels. For this reason, most existing on-line DVS algorithms for embedded systems use simple heuristics in estimating slack times. The comparison of DVS algorithms are shown in Fig. 1-3.

The system to be designed is specified as an abstract set of tasks. Selecting the right architecture has been shown to have a great influence on the system energy consumption (Dave *et al.*, 1997). Recently, with the advent of Dynamic Voltage Supply (DVS) processors (Suzuki *et al.*, 1997), highly flexible systems can be designed. Selecting different supply voltage scaling to reduce the energy consumption is carried out. Since the supply voltage has a direct impact on processor speed, the supply voltage selection has to be taken together for analysis. Scheduling will give rise to another level of possibilities for achieving energy/power efficient systems, when the system architecture is fixed or the system exhibits a very dynamic behavior. For such dynamic systems, various power management techniques exist (Pedram, 2001). Several scheduling techniques for soft real-time tasks, running on DVS processors have already been available (Chandrakasan *et al.*, 1996; Pering *et al.*, 1998; Weiser *et al.*, 1994).

Energy reductions can be achieved in hard real time systems also, where no deadline can be missed, (Hong *et al.*, 1998; Lee and Krishna, *et al.*, 1999; Shin and Choi 1999; Yao *et al.*, 1995). In this study the focus is on hardware implementation of Hybrid DVS (that includes both inter DVS and intra DVS algorithms) for hard real-time system. In this approach the deadline of the task has to be met. Task level frequency scheduling decisions

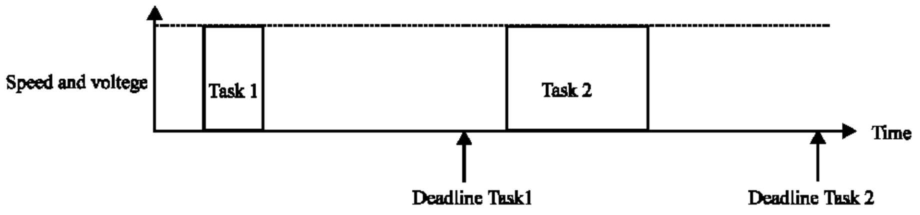

Fig. 1: Execution of tasks with out DVS scheduling

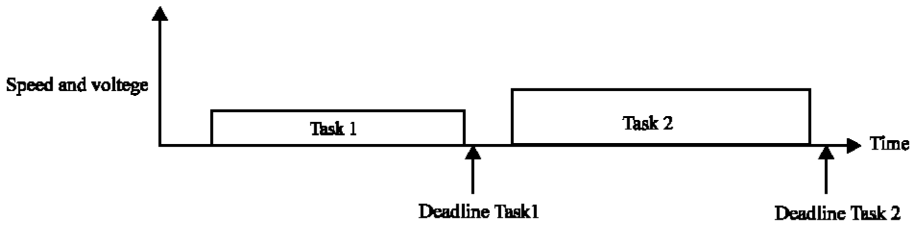

Fig. 2: Execution of tasks with inter DVS algorithms

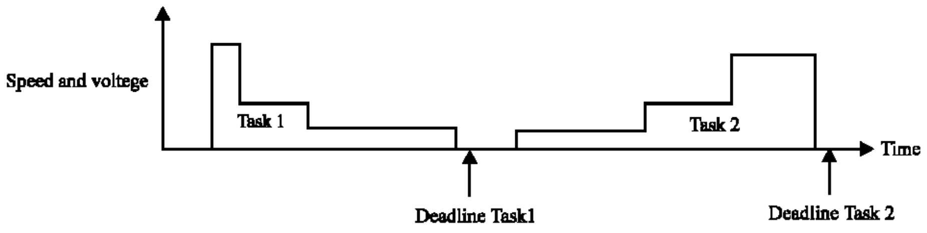

Fig. 3: Execution of tasks with intra DVS algorithms

can reduce even further the energy consumption. Some of these intra-task scheduling methods use several frequency levels inside a task and are implemented by using VHDL. Alternatively, fixing the schedule before the task starts executing as in (Gruian and Kuchcirski, 2001, Hong *et al.*, 1998) eliminate the internal scheduling overhead, but with possible affects on energy reduction. Statistics can be used to take full advantage of the dynamic behavior of the system, both at task level (Mossé *et al.*, 2000) and at task-set level (Yao *et al.*,1995). In this approach, the dynamic selection of the frequency is done here with out taking the stochastic data.

NEED FOR THIS ARCHITECTURE

Most of Inter DVS and Intra DVS algorithms are available in software which is based on object oriented programming or Real time operating system. The time required to adjust voltage and frequency will be more and also the processor is involved for the selection of inter or intra DVS. If this is implemented in hardware the reduction of delay and better utilization of processor are achieved.

AGGRESSIVE SPEED REDUCTION INTERDVS ALGORITHM

The aggressive speed reduction algorithm is better one compared to various algorithms. We can adopt an 'aggressive' approach based on reducing the speed of the running task under certain conditions to a level which is even lower than the other algorithms. However, this speculative move might shift the task's worst-case completion time to a point later than the one in S_c under an actual high workload. We should be ready to increase the CPU speed beyond S' later to guarantee feasibility of future tasks.

More specifically, S_{optavg} is the optimal speed for the workload where each instance requires exactly its average computational demand (determined by a probability distribution function). At one end of the spectrum, S_{min}/S_{optavg} (which is usually much smaller than 1.0) corresponds to the extreme aggressiveness where we attempt to obtain the lowest speed level for the running task.

Whenever we can predict that the completion time of the currently ready task (T_1, T2, T_X) will not extend beyond the next event arrival/deadline. We can speculatively reduce the speed of T_X while guaranteeing that it will still complete before the next event (which is, by definition, earlier than or equal to the deadline of T_X). Care must still be taken in order to guarantee the timely completions of other ready tasks, which are waiting in the ready queue. The execution/completion of these tasks will be delayed until T_X completes. This algorithm is implemented in hardware and simulation result is shown in the Fig. 4 and device utilization reports are given in Table 1.

Table 1: Device utilization for Inter DVS algorithm

	Selected Device: 2v80cs144-6		
Resources	Used	Avail	Utilization (%)
Number of slices	437	512	85
Number of slice flip flops	104	1024	10
Number of 4 input LUTs	764	1024	74
Number of bonded IOBs	17	92	18
Number of GCLKs	1	16	6

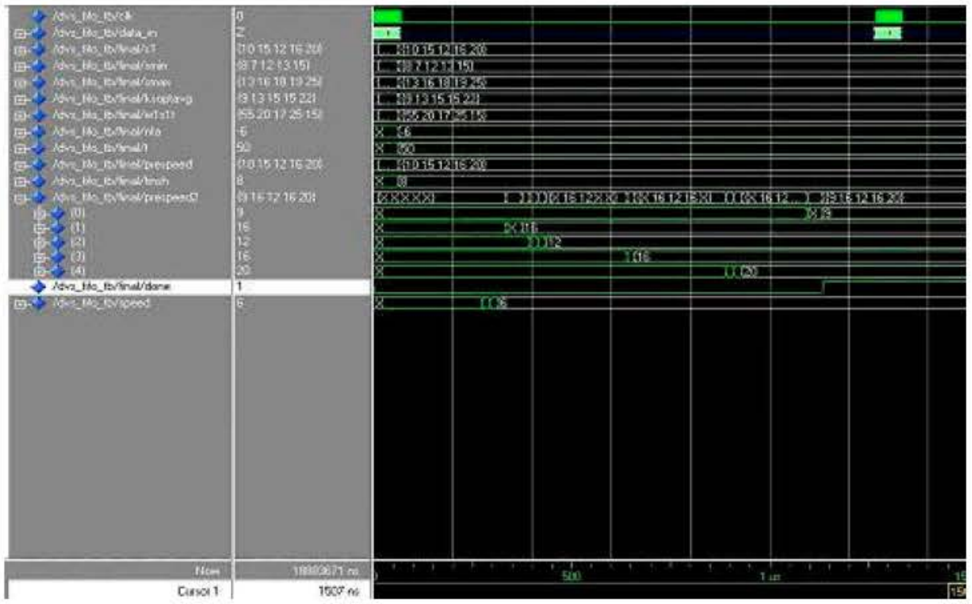

Fig. 4: Simulation result of inter DVS algorithm

INTRADVS ALGORITHM

This method is based on an observation that, from the energy consumption point of view, it is better to execute with a lower speed in the beginning and increase the execution speed later when necessary. In this study the IntraDVS algorithm uses the slack time which is gained from the previous task for the current task and the execution time history of the current task. Using this data, the selection of the frequency for the current task is done. There are four frequency used namely ¼, ½, ¾ and 1 (block and circuit diagrams for clock generation are shown in Fig. 5 and Fig. 6, respectively). If there is only one task is present then the slack time from the previous history is taken and 12% time of the task is run at a clock frequency of ¼, 12 % time of the task is run at clock frequency of ½, 12 % time of the task is run at clock frequency of ¾ and the remaining time of the task is run at clock speed. In this method the clock speed are statically determined by computing the differences between the execution cycles of the predicted execution path (e.g., Worst Case Execution Path (WCEP) and the execution cycles of the execution path actually taken). When the actual execution deviates from the predicted execution path, then the change in workload is detected and clock speed is adjusted.

Table 2: Device utilization for Intra DVS algorithm

	Selected Device: 2V500fg256		
Resources	Used	Avail	Utilization (%)
Ios	21	172	12.21
Function Generators	3469	6144	56.46
CLB Slices	1735	3072	56.48
Dffs or Latches	778	6660	11.68

The above algorithm is implemented using state diagram and it takes twelve states. This is implemented using VHDL and the simulation result is shown in Fig. 8, RTL view shown in Fig. 7 and synthesis reports is given in Table 2

The five task sets and the slack time are considered for the simulation. The result shows that the clock frequency is being selected based on slack time and the execution time of the present task.

HYBRID DVS ALGORITHM

The two algorithms are implemented in FPGA. Depending upon the task and slack time a particular algorithm is selected and it will adjust either voltage or clock speed. The block diagram is represented in the Fig. 9. The Table 3 gives the selection of algorithm depending upon the task set values.

The Hybrid DVS simulation result is given in Fig. 10. In simulation result shows only one DVS algorithm is active. The DVS algorithm is selected depending upon the number of task sets available. In this case the clock frequency is adjusted as per the algorithm. By using this approach the power reduction is achieved.

The synthesis report is generated using the VHDL code. In this Hybrid DVS algorithm the device utilization is the combination of InterDVS, IntraDVS and extra

Table 3: Selection of DVS algorithm

Heuristic	Description
S_0	Uses only intra mode since only one task is available
S_1	Uses the intra mode when the unused slack time is more than a predefined amount of slack time
S_2	Uses inter mode since the number of task is more than 1

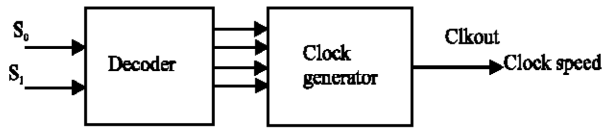

Fig. 5: Block diagram of Clock generation unit

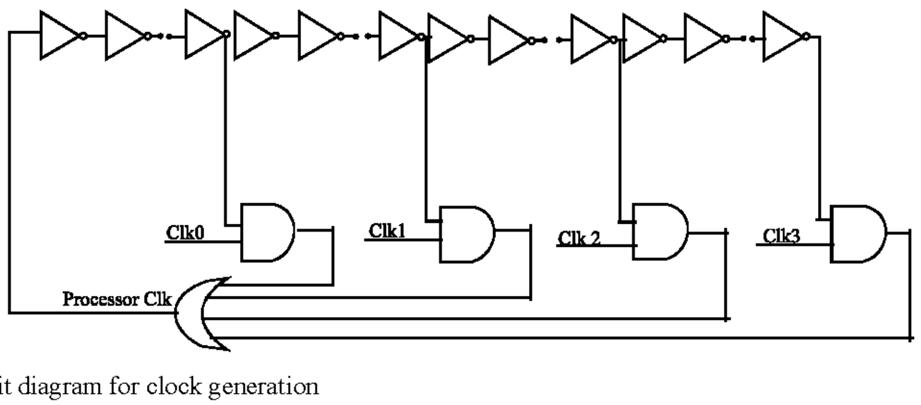

Fig. 6: Circuit diagram for clock generation

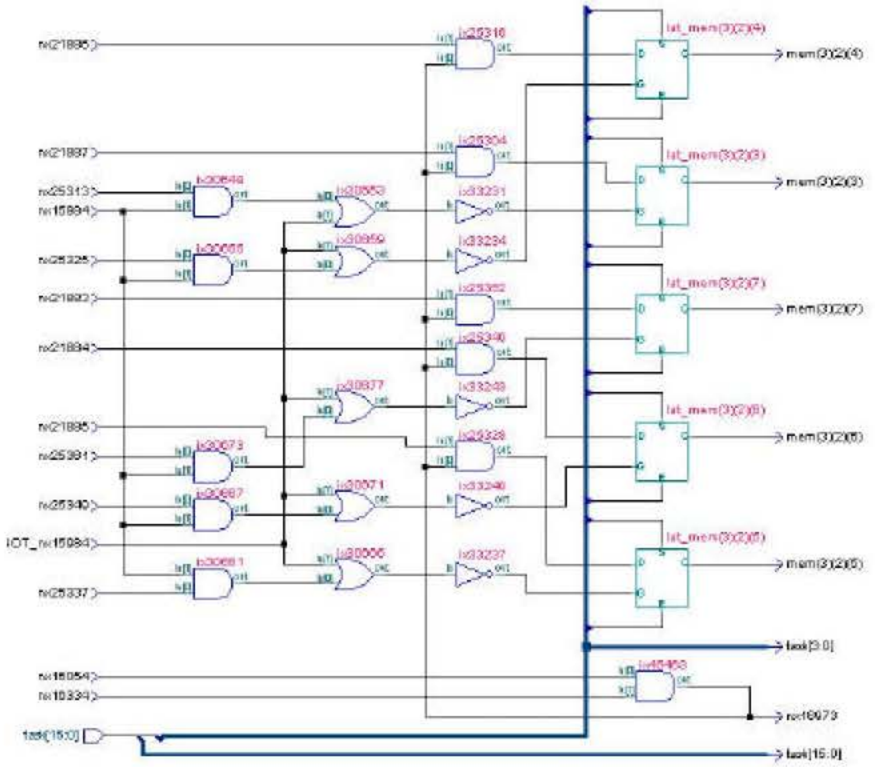

Fig. 7: RTLView

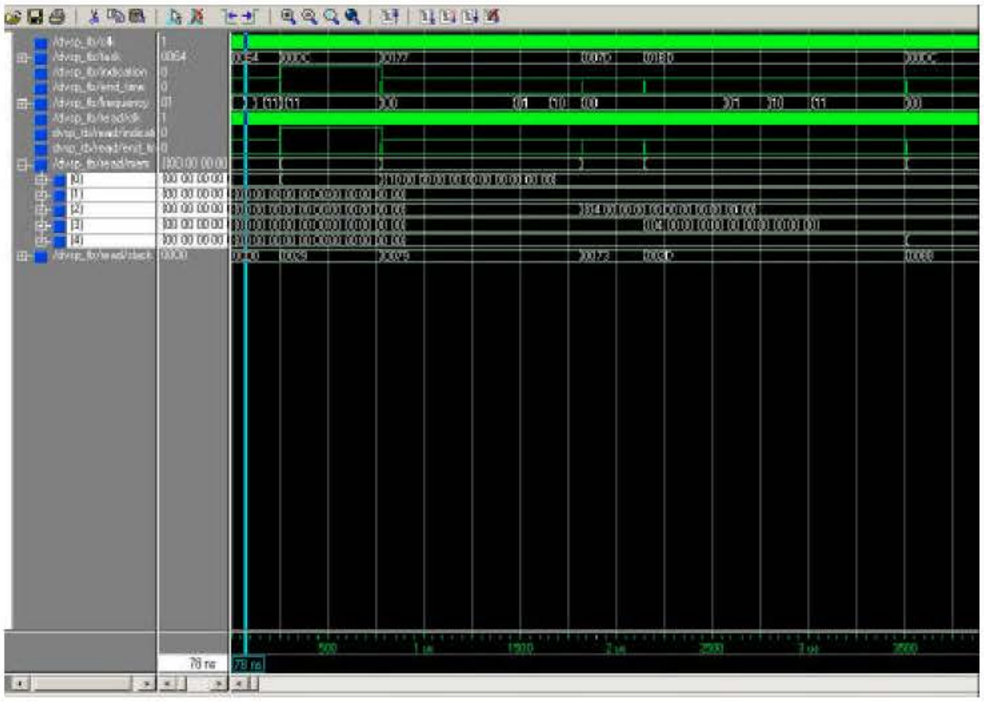

Fig. 8: Intra DVS Simulation result

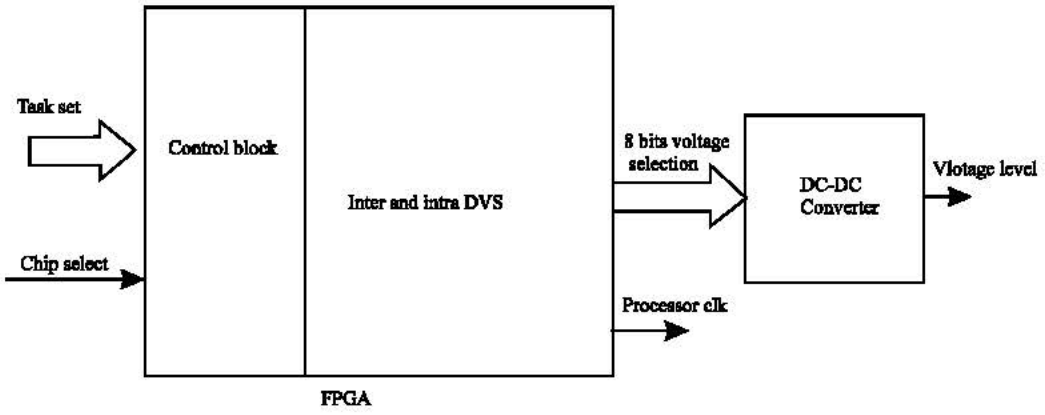

Fig. 9: Block diagram of hybrid DVS

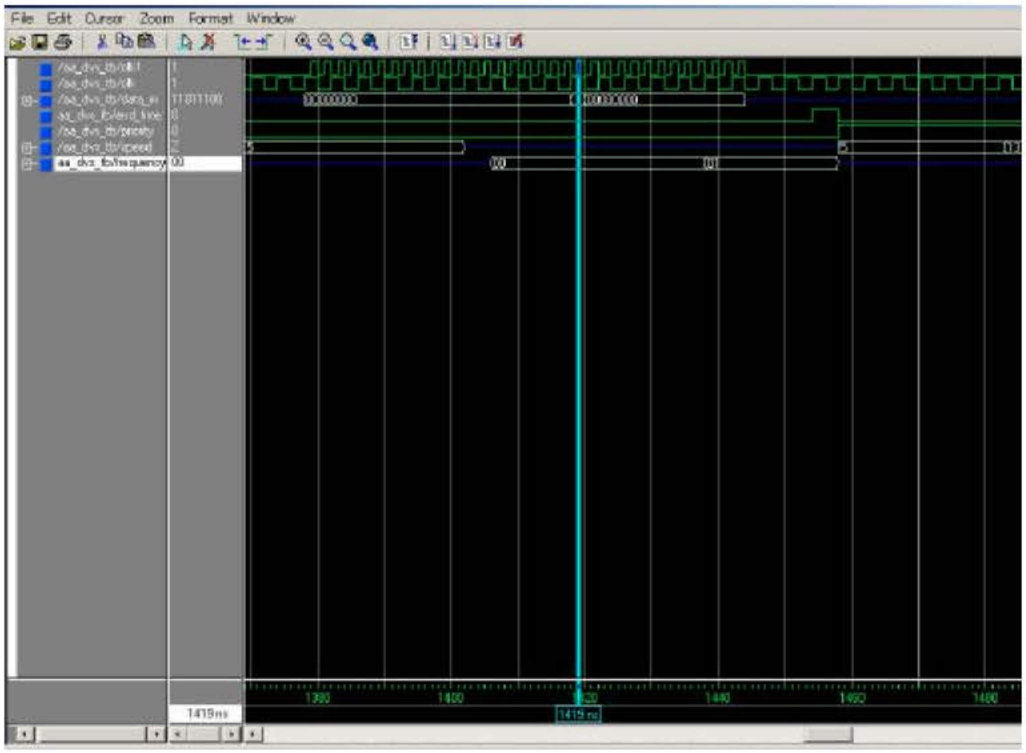

Fig. 10: Hybrid DVS simulation result

Table 4: Device utilization for hybrid DVS algorithm

	Selected Device: 2V1000bg575		
Resources	Used	Avail	Utilization (%)
Ios	22	328	6.71
Function generators	4840	10240	47.27
CLB slices	2420	5120	47.27
Dffs or latches	1185	11224	10.56

hardware is requited to select which DVS algorithm to be run. The FPGA usage for both algorithms is shown in Table 4.

CONCLUSIONS AND FUTURE WORK

The above two best algorithms are implemented in FPGA. Using this algorithm the processor frequency and voltage can be varied in order to reduce the power. This hardware implementation will minimize the power consumption about 15 to 20% depending upon the number of tasks. In addition to reduction of power the processor can be better utilized (since DVS algorithm is moved from software to Hardware). This hardware can be a peripheral device to the processor (or coprocessor).

The future work will be implementing this in reconfigurable architecture where only one algorithm is active.

REFERENCES

Benini, L. and G. DeMicheli, 2000. System-level power optimization: Techniques and tools ACM Trans. Design Automation of Electronic Syst., 5: 115-192.

Burd, T., T. Pering, A. Stratakos and W. Brodersen, 2000. A dynamic voltage scaled microprocessor system IEEE J. Solid-State Circuits, 35: 1571-1580.

Chandrakasan, A., V. Gutnik and T. Xanthopoulos, 1996. Data driven signal processing an approach for energy efficient computing: In Proceedings of ISLPED, 347-352.

Dave, B.P., G. Lakshminarayana. and N.K. Jha, 1997. COSYN: Hardware-software co-synthesis of embedded systems In Proceedings of the 34th DAC, pp: 703-708.

Gruian, F. and K.L. Kuchcinski, 2001. Task scheduling for low-energy systems using variable voltage processors. In: Proceedings of ASP-DAC, pp: 449-455.

Hong, I., M. Potkonjak and M.B. Srivastava, 1998. On-line scheduling of hard real-time tasks on variable voltage processor. In; Digest of Technical Papers of ICCAD, pp: 653- 656.

Ishihara, T. and H. Yasuura, 1998.Voltage scheduling problem for dynamically variable voltage processors, in: Proceedings of ISLPED pp: 197-202.

Lee, S. and T. Sakurai, 2000. Run-time voltage hopping for low power real-time systems In: Proceedings of the 37th DAC, pp: 806-809.

Lee, Y.H. and C.M. Krishnal, 1999. Voltage-clock scaling for low energy consumption in real-time embedded systems; In: Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications, pp: 272-279.

Locke, C.D., D.R. Vogel and T.J. Mesler, 1989. "Building a predictable avionics platform in Ada: A case study: In: Proceedings of RTSS, pp: 181-189.

Mosse, D., H. Aydin, B. Childers and R. Melhem, 2000. Compiler-assisted dynamic power-aware scheduling for realtime applications. Workshop on Compilers and Operating Systems for Low-Power.

Pedram, M., 2001. Power optimization and management in embedded systems., Proceedings of ASP-DAC, pp: 239-244.

Pering, T., T. Burd and R. Brodersen, 1998. The simulation and evaluation of dynamic voltage scaling algorithms: In: Proc Intl. Symposium on Low Power Electronics and Design, pp: 76-81.

Shin, Y. and K. Choi, 1999. Power conscious fixed priority scheduling for hard real-time systems. Proceedings of the 36th DAC, pp: 134-139.

Suzuki, K., S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda and T. Kuroda, 1997. A 300MIPS/W RISC core processor with variable supply voltage scheme in variable threshold-voltage CMOS: Proceedings of the ICC, pp: 587-590.

Weiser, M., B. Welch, A. Demers and S. Shenker, 1994. Scheduling for reduced CPU energy In: Proc. UNSNIX Symposium on Operating Systems, Design and Implementation, pp: 13-23.

Yao, F., A. Demers and S. Shenkar, 1995. A scheduling model for reduced CPU energy. In: Proc IEEE Annual foundations of Computer Science, pp: 374-382.