LOW POWER DUAL PROCESSOR FAULT-TOLERANT SYSTEM

Borisav Jovanović, Milunka Damnjanović, *University of Niš, Faculty of Electronic Engineering*

Abstract *– The proposed low-power technique, which is based on based on Standby sparing technique, is used in fault tolerant multiprocessor systems. To validate technique, we have developed the multiprocessor system, consisting of two microcontroller cores. The system, maintains both the fault tolerance and low-power operation.*

1. INTRODUCTION

In modern System-on-Chip (SoC) design, the technology scaling has allowed more functionality per area, higher clock speeds, and lower switching energy, but has also increased the amount of leakage power. The dynamic voltage and frequency scaling (DVFS) and power gating became popular power management techniques and they are applied in many electronic circuits and systems [1].

The dynamic voltage and frequency scaling is the most utilized method for reducing dynamic component of power consumption. The DVFS reduces supply voltage and its use leads to a quadratic reduction in dynamic energy. However, in new technologies, the utilization of DVFS is limited by the significant increase of leakage power. Recent advances in circuit design have allowed researchers to operate chips with supply voltages far below the range normally used by integrated circuits [2].

Leakage power management is one of the main challenges in modern integrated circuit design. For example, the microprocessor manufacturers have adopted power-gating and programmable sleep operating modes to accommodate the growing popularity of low-power sensor, mobile and wireless applications. With these techniques, special transistors are used as gates that cut off the power from unused sub-blocks of a microprocessor, which substantially reduces the leakage power [3].

The hard real-time systems have to operate correctly even in the presence of faults. The faults can be classified into two categories - permanent and transient. Permanent faults may bring a system to stop and cannot be recovered without some form of hardware redundancy, while transient faults are often triggered by temporary causes and will disappear after some short time interval [4]. With reduced technology size, and operation at very low supply voltages, it is proven that modern computing devices are more susceptible to faults, and therefore, it is imperative to incorporate fault tolerant methods [5].

2. THE HARDWARE REDUNDANCY METHOD

The standby sparing technique [6] is used in hard real-time applications. The multi-microprocessor system that utilizes standby sparing method consists of two identical microprocessor cores. It is a hardware redundancy system, where redundancy lies in the utilization of one additional core. Since the power consumption is an important aspect of IC design, the system utilizes low power techniques such as DVFS and power gating. Besides, microcontroller cores incorporate active and standby operating modes.

The microprocessor core (called primary core) operates at the frequency maximum f_{max}, obtained at voltage level V_{max}. The primary core power consumption is reduced by decreasing the supply voltage. The microprocessor's voltage supply value is adjusted with scaling factor ρ which belongs to the range [ρ_{min},1].

$$\rho = \frac{V}{V_{\max}} \tag{1}$$

As DVFS method adjusts the supply voltage, the clock frequency f_{clk} is also adjusted because of the constraint:

$$f_{clk} \propto \frac{(V - V_T)^2}{V} \tag{2}$$

, where V_T is the threshold voltage, which is dependent on technology in which the circuit is implemented. Frequency tunings may spoil the performance of real-time systems, especially when there are hard deadlines to meet [6]. Because it can be assumed that operating frequency is approximately proportional to the supply voltage value, ρ will also be used to represent the microcontroller's speed. The scaling factor ρ is equal to the ratio of actual operating speed f_{clk} to the full speed f_{max}.

The spare processor core is used because the system has to be recovered from permanent faults. The spare core is identical to the primary core, executes the same sequence of tasks, but resides in standby operating mode most of the time. Besides, the spare core operates at maximum levels of operating frequency and supply voltage - f_{max} and V_{max}, because high supply voltage level enables the fault tolerant operation.

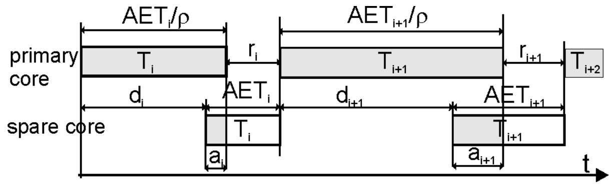

Fig.1. *The primary and spare cores execute tasks*

The operation of the system is explained on the example of group of tasks. Each task Ti is executed within actual execution time interval (AET), which is less or equal than worst case execution time interval (WCET). All the tasks should be completed before the specified deadline time D. For every task running on primary core that has been selected to be scaled down by DVFS, the separate recovery job is scheduled. The spare core executes the recovery task when primary core's task incurs faults. However, to meet the deadline time, the spare core is often turned on before the primary core finishes its operation and possible faults are checked. If primary core completes the task without any error, spare core cancels further operation, and returns to standby mode. If primary core fails executing the task, the backup task is completed by spare core.

The Fig.1 shows the execution of successive tasks T_i and T_{i+1}. The primary core, which operates at speed ρ, finishes the task within the time interval AET_i/ρ. The spare unit, working at f_{max}, starts the execution after delay d_i and finishes after the time interval AET_i. During the delay time the spare unit is in standby mode. At the moment when primary core finishes the task T_i, the operation of this task is checked. The error detection is usually assumed to be part of the software and error detection overhead is considered as a part of the execution time [7]. If error is not found, the execution of spare task is cancelled. If primary core fails, the next task T_{i+1} can be started after the spare unit completes the backup task T_i.

To reduce the total power dissipation, the time interval during which the spare stays in Active mode should be minimized. Therefore, the delay time d_i should be increased. but it is limited by the deadline time D.

The inter-task time r_i defined as the delay between two consecutive tasks T_i and T_{i+1} can be determined by Eq.3:

$$r_i = d_i - AET_i(1/\rho - 1) \tag{3}$$

The energy consumption of spare depends on the duration of spare active time interval a_i, given by Eq.4:

$$a_i = AET_i - r_i = \frac{AET_i}{\rho} - d_i \tag{4}$$

The equation can be modified as follows:

$$a_i + r_i = AET_i \tag{5}$$

The maximum value of sum of all inter-task times r_i is given by the slack time SLT, and it is equal to the difference of deadline time D and the sum of all actual execution times (Eq.6).

$$SLT = \left(\sum_{i=1}^{N} r_i\right)_{max} = D - \frac{1}{\rho}\sum_{i=1}^{N} AET_i \tag{6}$$

The active time A of spare unit contributes the most of total power consumption. The active time A is:

$$A = \sum_{i=1}^{N} a_i = \sum_{i=1}^{N} AET_i - \sum_{i=1}^{N} r_i \tag{7}$$

The spare active time minimum is:

$$(A)_{min} = \sum_{i=1}^{N} AET_i - D, \quad \sum_{i=1}^{N} AET_i > D \Big/ \left(1 + \frac{1}{\rho}\right)$$
$$(A)_{min} = 0, \quad \sum_{i=1}^{N} AET_i \le D \Big/ \left(1 + \frac{1}{\rho}\right) \tag{8}$$

3. THE MODIFICATION OF HARDWARE REDUNDANCY METHOD

The power consumption of spare unit can be larger than the consumption of primary unit. The modification of standby sparing method, which is based on concatenation of successive tasks, reduces the total power consumption.

Consider the execution of two consecutive tasks T_i and T_{i+1}. The primary core operates at speed ρ and completes the task T_i and T_{i+1} for time interval AET_i/ρ and AET_{i+1}/ρ. Immediately after a primary core finishes the T_i, it starts the execution of the next task T_{i+1}. The delay period between T_i and T_{i+1} is lost. The spare starts with execution after delay d_i. If the task T_i is completed without error, the recovery task T_i is stopped. The task T_{i+1} is started on the primary unit. At the moment when primary core finishes the T_{i+1}, the operation of T_{i+1} is checked. If task T_{i+1} is finished without errors, the recovery task T_{i+1} is cancelled.

The execution of recovery tasks T_i and T_{i+1} is completed if faults occur on the primary core. The execution of T_{i+1} on primary core is cancelled. In this situation, the total energy consumption is increased because spare unit completes both tasks T_i and T_{i+1}. The fault tolerance of tasks T_i is preserved, but for T_{i+1} it is slightly degraded because this task is executed only by spare core. The probability of an error to occur is small because of high supply voltage level of spare core.

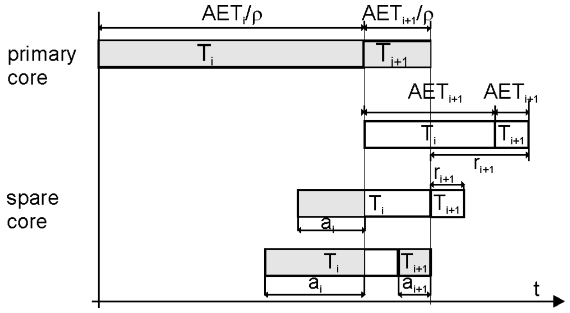

Fig.2. *The concatenation of tasks, case 1: $AET_i > AET_{i+1}/\rho$*

First, the relation given by Eq.9 is considered. The relation is illustrated in Fig.2.

$$AET_i > \frac{AET_{i+1}}{\rho} \tag{9}$$

In the case when spare core executes only the part of task T_i, the sum of inter-task times and spare active intervals of tasks Ti and Ti+1 is equal to:

$$\sum_{j=i,i+1} (r_j + a_j) = AET_i + AET_{i+1}\left(1 - \frac{1}{\rho}\right)$$
$$r_{i+1} \in \left[AET_{i+1}, AET_i + AET_{i+1}\left(1 - \frac{1}{\rho}\right)\right] \tag{10}$$

When the spare unit executes the parts of tasks, the sum of inter-task times and spare active times of tasks Ti and Ti+1 is:

$$\sum_{j=i,i+1} (r_j + a_j) = AET_i + AET_{i+1}\left(2 - \frac{1}{\rho}\right) - r_{i+1},$$
$$r_{i+1} \in [0, AET_{i+1}] \tag{11}$$

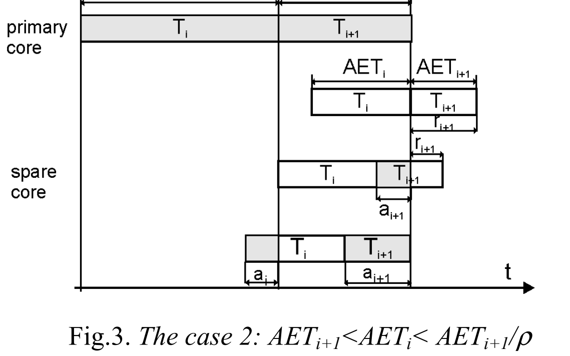

Fig.3. *The case 2: AET_{i+1}<AET_i< AET_{i+1}/ρ*

Then, the relation given by Eq.12 is considered. The situation is shown in Fig.3.

$$AET_{i+1} < AET_i < \frac{AET_{i+1}}{\rho} \tag{12}$$

If spare unit executes only task T_{i+1}, the sum of inter-task times and spare active times of tasks T_i and T_{i+1}, is equal to:

$$\sum_{j=i,i+1} (r_j + a_j) = AET_{i+1},$$
$$r_{i+1} \in \left[AET_i + AET_{i+1}\left(1-\frac{1}{\rho}\right), AET_{i+1} \right] \tag{13}$$

When the spare unit executes the parts of both tasks, the sum is given by Eq.14:

$$\sum_{j=i,i+1} (r_j + a_j) = AET_i + AET_{i+1}\left(2-\frac{1}{\rho}\right) - r_{i+1},$$
$$r_{i+1} \in \left[0, AET_i + AET_{i+1}\left(1-\frac{1}{\rho}\right) \right] \tag{14}$$

The third situation, given in Fig.4, is expressed by condition Eq.15. The sum of inter-task times and spare active times of tasks T_i and T_{i+1}, is provided by Eq.16.

$$AET_i < AET_{i+1} \tag{15}$$

$$\sum_{j=i,i+1} (r_j + a_j) = AET_{i+1},$$
$$r_{i+1} \in [0, AET_{i+1}] \tag{16}$$

The power consumption of spare microprocessor core is reduced if spare active time interval a_i is minimized. The concatenation of tasks decreases the total sum of inter-task times and spare active times.

In the first case, which is shown in Fig.2, the sum of inter-task times and spare active times of tasks is reduced by value AET_{i+1}/ρ. It other two cases, given in Fig.3 and Fig.4, the reduction is equal to the value of AET_i.

The described operation is repeated until the sum of all inter-task times and spare active times becomes equal or less than slack time SLT. Further reduction of the sum would not yield the energy efficiency.

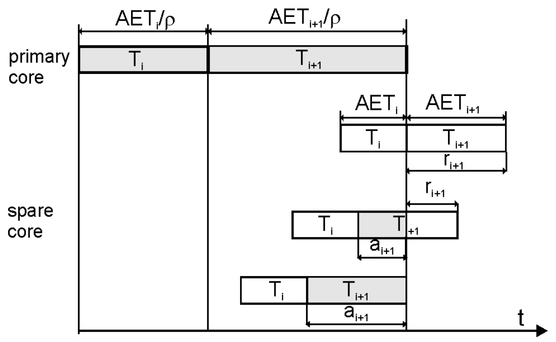

Fig.4. *The concatenation of tasks, case 3: $AET_i < AET_{i+1}$*

4. THE IMPLEMENTATION OF THE SYSTEM

The low power, fault tolerant system is created. The 8052 microcontroller [8] was used which global architecture of consists of core, memory blocks, and peripheral units. The peripherals are comprised of three timer/counter circuits, three digital input/output parallel ports, one asynchronous universal receiver/transmitter and one I2C serial communication block. Three main memory blocks are present: program memory (on-chip 8kB SRAM), external data memory XRAM (physically consisting of on-chip 2kB SRAM), and internal data memory IRAM (comprising of 256 Internal Dual-Port RAM and Special Function Register set).

The microcontroller was implemented using Synopsys 90nm digital standard cell library and Synopsys tool suite [9]. The microcontroller's layout was partitioned into three power domains. The MCU core, the peripherals and memories were put into distinct domains which enable shutting down of inactive blocks. The header MTCMOS [2] transistors were utilized for implementation of power domains. During standby operating mode the core is switched off from the power supply, and the peripheral units and memories are kept powered.

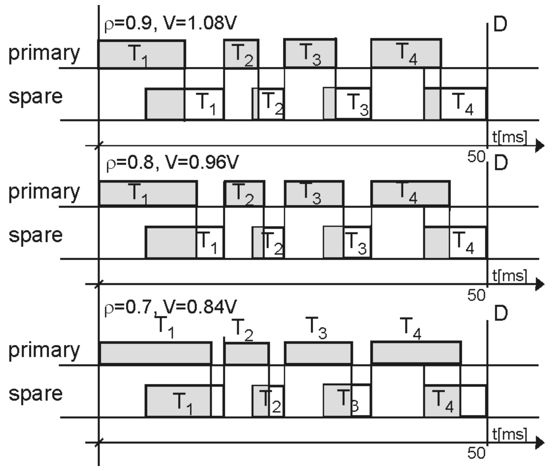

Fig.5. *A schedule of tasks, when standby sparing is applied*

The fault tolerant system was created consisting of two identical 8052 cores, called the primary and spare cores. The primary and spare cores were implemented in different power domains. Since the spare core executes the same program code as primary unit, it shares with primary core RAM memory blocks. The RAM memories represent the largest blocks. The RAM area is three times larger than the area of single core [8]. Therefore, the additional microcontroller core does not increase significantly the total chip occupied area.

The spare core operates at nominal supply voltage V_{max}=1.2V and executes the instructions at the maximum clock frequency f_{max}=120MHz. The supply voltage of primary core can be reduced by DVFS from V_{max} downto $V_{reduced}$=0.84V.

The system is evaluated on a schedule of tasks which energy consumption values were estimated. The schedule consists of four tasks T1, T2, T3 and T4 with actual execution intervals equal to 10ms, 4ms, 6ms and 8ms respectively (Fig. 5). Standby sparing method was applied. Three power analyses were conducted with different values of parameter ρ. The normalized speed of primary core was changed in the range from 0.9 to 0.7. The corresponding voltage supply levels of primary core are 1.08V, 0.96V and 0.84V.

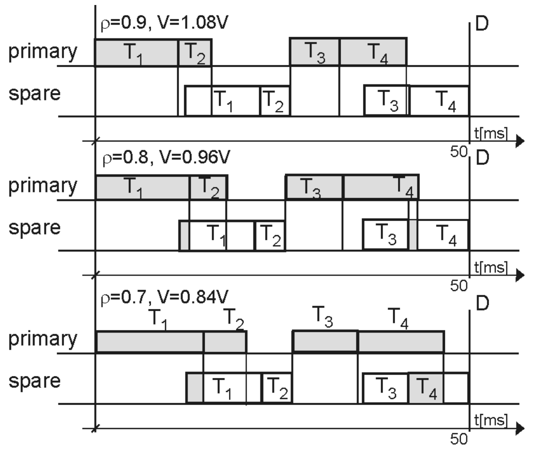

Fig.6. *A schedule of tasks after the concatenation of tasks is performed*

The power consumption was obtained after the layout was implemented and layout netlist was verified. To estimate the dynamic power, the switching activity of the nets was recorded during logical verification. The results are given in the Table 1. The total energy consumption values are 313.25μJ, 306.02μJ and 315.04μJ. After, the described method was applied. The task T1 is joined with T2, while T3 is concatenated with T4. The parameter ρ of primary core was changed in the range from 0.9 to 0.7. The Fig. 6 gives the schedules where tasks are concatenated. The energy consumption of modified microcontroller system was estimated. The obtained values are: 223.9μJ, 198.1μJ and 200.1μJ; in other words, approximately 33% energy reduction is obtained compared to the non-optimized implementation.

Table 1. *The power optimization results*

Normalized speed ρ	Primar. core voltage [V]	Energy of primary core [μJ]	Energy of spare core [μJ]	Total energy [μJ]
Before optimization				
0.9	1.08V	223.905	89.351	313.256
0.8	0.96V	178.535	127.491	306.026
0.7	0.84V	138.52	176.526	315.046
After optimization				
0.9	1.08V	223.905	0	223.905
0.8	0.96V	178.535	19.614	198.149
0.7	0.84V	138.52	61.646	200.166

5. CONCLUSION

In this paper, the modification of standby sparing technique is proposed, in which the tradeoff between the fault tolerance and energy consumption is made. The technique relies on hardware redundancy and utilizes low power techniques: dynamic voltage scaling and power gating.

To validate the technique, the hard real time system was created, which incorporates two identical microcontroller cores. The first microprocessor core is called the primary unit and operates at lower voltage, reducing the power consumption. The other microcontroller core, called spare unit, exploits power gating to conserve the power and ensures the fault tolerant operation. The spare unit operates at higher voltage, but is most of the time inactive, reducing the total energy consumption.

6. REFERENCES

[1] M. Keating, D. Flynn, R. Aitken, A. Gibbons, K. Shi, Low Power Methodology Manual, Springer, 2007.

[2] P. Bipul, A. Agarwal, K. Roy "Low-Power Design Techniques for Scaled Technologies," *Integration, The VLSI Journal*, Vol. 39, Issue 2 (2006), pp. 64–89

[3] H. Kopetz, Real-time systems: Design principles for distributed applications, Kluwer Academic Publishers, 2002

[4] S. Poledna, Fault tolerant real-time systems: The problem of replica determinism, Kluwer Academic Publishers, 1996

[5] D.K. Pradhan, Fault tolerant computer system design, Prentice Hall, 1996

[6] A. Ejlali, B. M. Al-Hashimi M. T. Schmitz P. Rosinger, S. G. Miremadi, "Combined Time and Information Redundancy for SEU -Tolerance in Energy Efficient Real-Time Systems," *IEEE Trans. VLSI Systems*, Vol.14 no.4 pp.323-335, April 2006.

[7] P. Eles, V. Izosimov, P. Pop, Z. Peng "Synthesis of Fault Tolerant Real-Time Systems" in *Proc. Design, Automation and Test in Europe*, 2008

[8] B. Jovanović, M. Zwolinski, M. Damnjanović, "Low power digital design in Integrated Power Meter IC," *in Proc. of the Small Systems Simulation Symposium* 2010, Niš, Serbia

[9] Synopsys 90nm Generic Library for Teaching IC Design, http://www.synopsys.com, accessed April 2010

[10] B. Jovanović, M. Damnjanović, "Energy efficiency of hard real-time systems based on standby sparing," *Zbornik LVI konferencije ETRAN* 2012

Sadržaj – Predložena tehnika zasnovana je na Standby sparing metodi. Koristi se u real-time sistemima za uštedu snage dispacije mikroprocesora koji rade na malim naponima napajanja. Za proveru tehnike razvijen je multiprocesorski sistem koji se sastoji od dva identična jezgra mikrokontrolera. Sistem obezbeđuje toleranciju na greške i malu potrošnju.

DVOPROCESORSKI SISTEM MALE POTROŠNJE TOLERANTAN NA GREŠKE

Borisav Jovanović, Milunka Damnjanović