Concerns about High Functionality IP Cores in a System with Functional Redundancy

Ferreira, Victor M. Goulart
Department of Computer Science and Communication Engineering, Kyushu University

Yasuura, Hiroto
Department of Computer Science and Communication Engineering, Kyushu University

出版情報：第2回組込みシステム技術に関するサマーワークショップ(SWEST2)報告書, pp.73-78, 2000-08-01
バージョン：
権利関係：

Concerns about High Functionality IP Cores in a System with Functional Redundancy

Victor M. Goulart Ferreira and Hiroto Yasuura
Department of Computer Science and Communication Engineering
Kyushu University
Kasuga-koen 6-1, Kasuga, Fukuoka, Japan 816
email: goulart@c.csce.kyushu-u.ac.jp

Abstract

This paper discusses the impact of the use of high functionality IP cores on software development for a system designed with Functional Redundancy. Functional Redundancy is a novel approach for system design which aims at shortening TAT (Turn Around Time) for SoC designs and allowing dynamic exchange of system's characteristics of power consumption and performance. It is based on having different implementation instances of circuits with same functionality, but with different performance-energy consumption characteristics, in the same chip. An RTOS aware of the functional redundancy present in the circuit will make the dispatch of instructions to the highest performance or to the lowest power consuming IP core, according to the system load, thus, obtaining minimum power consumption while meeting performance requirements. Problems related with IP cores' level of functionality and its applicability to Functional Redundancy design approach are also discussed.

Keywords: *High Functionality IP Cores, Functional Redundancy, System Synthesis, RTOS, SoCs.*

1. Introduction

Advances on transistor integration technologies have enabled to effectively realize SoC (System-On-a-Chip) solutions, which implement the whole functionality of a system on a single chip. SoCs are present at portable devices such as notebooks, PADs and communication devices, control parts of an automobile and so on. Such real-time processing systems demand high performance computing, and low power consumption, as most of these systems are battery operated.

Designers usually face a strict compromise between performance and power consumption of a design, while taking into account die size of the IC. Nevertheless, as transistor gate sizes continue to shrink, chip area is no longer a big burden on design constraints. Furthermore, ways to make good use of the now extraordinary available number of transistors is pressing. The use of IP cores and libraries, with different implementation instances of circuits with same functionality, gives the designer more freedom to exploit the design solution reducing TAT and, consequently, time to market.

It is extremely difficult to make a good composition of performance and low power consumption, specially for dynamic systems where the system workload changes over time. The decisions made at the very beginning of system design will permanently define the characteristics of the system, giving no more room for changes. This can lead to overestimations of a design with greater power consumptions.

In order to overcome this problem we have proposed the Functional Redundancy design approach[3], where even after chip fabrication it is possible to make use of completely different performance and power consumption characteristics combinations. The exchange of a certain composition of performance and power consumption is orchestrated by a Real-time Operating System (RTOS). In a real-time control system, the code of each task is known a priori and hence can be analyzed to determine its characteristics in terms of computation time, resources, and precedence relations with other tasks. The task's parameters can be used by the operating system to verify its schedulability within the specified timing requirements. We will be back to this topic later.

A preemptive schedule is a schedule in which the running task can be arbitrarily suspended at any time, to assign the CPU to another task according to a predefined scheduling policy. We are interested in reducing the power consumption of such dynamic systems, preserving high performance. Die size area is not our concern since silicon technology advances has been lowering the price of SoCs. The

set of tasks (system load) may change over time, although off-line scheduling can also benefit from Functional Redundancy.

The rest of this paper is organized as follows: Section 2 presents an overview of some techniques used in order to meet low power consumption not sacrificing performance, i.e. attending task deadlines. The Functional Redundancy design approach is presented in Section 3. Section 4 discusses some problems related with high functionality cores. Section 5 presents discussions over the methodology and its feasibility. Section 6 concludes the paper.

2. Related Works

Some works have been proposed to satisfy both high performance and low power consumption using dynamically variable voltage hardware. Another approach is cutting power supply to those parts of the circuit which are not going to be used. Although it does not have any impact in performance, power consumption can be greatly reduced.

2.1. Dynamically Variable Voltage Hardware

Variable voltage processors[7, 11] which can vary its supply voltage dynamically have been proposed. In these processors a combination of hardware and software solutions allow it to change supply voltage. Reducing the supply voltage however increases circuit delay and decreases clock speed.

Using the variable voltage processor, tasks with strict real-time constraints can be executed with a high supply voltage (therefore, high clock frequencies), and tasks with loose time constraints are done with a low supply voltage. The effectiveness of the approach is guaranteed as reducing the supply voltage leads to drastic power reduction because power consumption in CMOS circuits typically increases in proportion to the square of its supply voltage,

$$P_{dynamic} = \sum_{k=1}^{M} CL_k \cdot switch_k \cdot V_{DD}^2 \qquad (1)$$

where M is the number of gates in the circuit, CL_k the load capacitance of gate g_k, $switch_k$ the switching count of g_k per clock cycle, and V_{DD} the supply voltage. Taking into account the number of total cycles executed, $cycle_{task}$, the total energy consumption is expressed as,

$$\begin{aligned} E_{task} &= cycle_{task} \cdot \sum_{k=1}^{M} CL_k \cdot switch_k \cdot V_{DD}^2 \\ E_{task} &= cycle_{task} \cdot P_{dynamic} \qquad (2) \end{aligned}$$

Real-time task scheduling algorithms for the variable voltage processor for hard and soft real-time constraints[8, 5] also have been proposed.

2.2. Selectively Cutting Power Supply Off

In this approach low power consumption is obtained by cutting off the power supply of unused parts of the circuit. The unused parts can be whole or part of cores or regions in the circuit, like part of a bus or the upper digits of an adder, and so on (Figure 1).

The extreme of such technique for processor cores is the system shutdown[10] and other techniques for system (semi-)hibernation, like the new Drowsy Mode of the next generation Intel StrongARM Technology. [4].

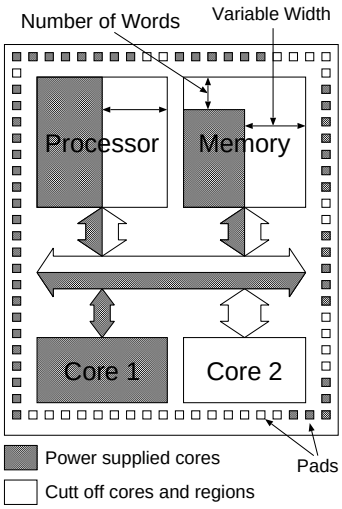

Figure 1. Low power solution by cutting power supply of cores and parts of the circuit off.

FlexSys (Flexible System LSI)[6] is a chip architecture with several cores connected by internal buses. The cores include microprocessor, memories and peripheral circuits suited for a particular application domain. The designer when implementing a customized chip, specifies which cores are going to have their power lines cut off at the mask level in the fabrication phase. The main aim of FlexSys is to minimize the number of layout masks which have to be designed (those used to customize the power supplying area). Actually if the design changes a new whole set of masks must be remade, leading to unnecessary expenditures. Nevertheless, the problem of deciding which cores will be integrated in this overestimated architecture seems to be as hard as, if not harder than deciding which cores to connect in a application specific circuit. Also, the advantages of FlexSys

may only be visible for mass-production circuits, where the unit cost of the chip can be reduced. The main disadvantage of FlexSys is that the final circuit and its performance is defined at the design level and is fixed after chip manufacture.

Another similar approach, but with much more flexibility is the use of a special register to specify which cores are going to have their power lines cut off. As we have seen (Equation 1), reducing the switching activity can also lead to power reductions. Here there are three basic approaches: *Reducing wasteful switching activity*[12, 2]; *Optimizing signal protocols and encoding*[9]; and *Optimizations by algorithm selection*, where different implementations of a given function by different hardware algorithms lead to completely different cores, in terms of power consumption, area and computation time.

The effect of algorithm selection in core's characteristics can be seen in Table 2 for different implementations of multipliers (Table 1).

3. The Functional Redundancy System Design Approach

Silicon capacity is doubling every 18 months allowing feasibility of SoC implementations. The design of such complex systems in reasonable time is not possible without reuse of components and system parts. The reuse of components have created the IP (Intellectual Property) community, with IP providers, IP tool makers, IP integrators and so on. In the actual design approach, cell libraries of IP cores determine the final density, performance, and power of most IC designs. Choosing the right cell library for a project can have a significant impact on the characteristics of the circuit designed, and thus the success of a product. Technical and business considerations normally determine the choice for a certain library.

In this section we describe the Functional Redundancy approach for system design[3]. It is heavily IP-oriented and shows to be an effective solution for dynamic exchange of system characteristics even after chip tape-out. This methodology is best suited for dynamic systems in which high performance and low power consumption is a much more important concern than die size.

The basic idea is to have in the same circuit, different instances (IP cores) with same functionality, but with different performance-energy consumption characteristics. An RTOS aware of the functional redundancy present in the circuit will make the dispatch of instructions to the highest performance or to the lowest power consuming cores, or to some other, with a suitable combination of characteristics, according to the system load. For instance, in case of a processor endowed with functional redundancy, running a text editor, low speed, low power consuming IP cores should be used, while for intensive multimedia applications, high performance ones should be chosen for executing the task.

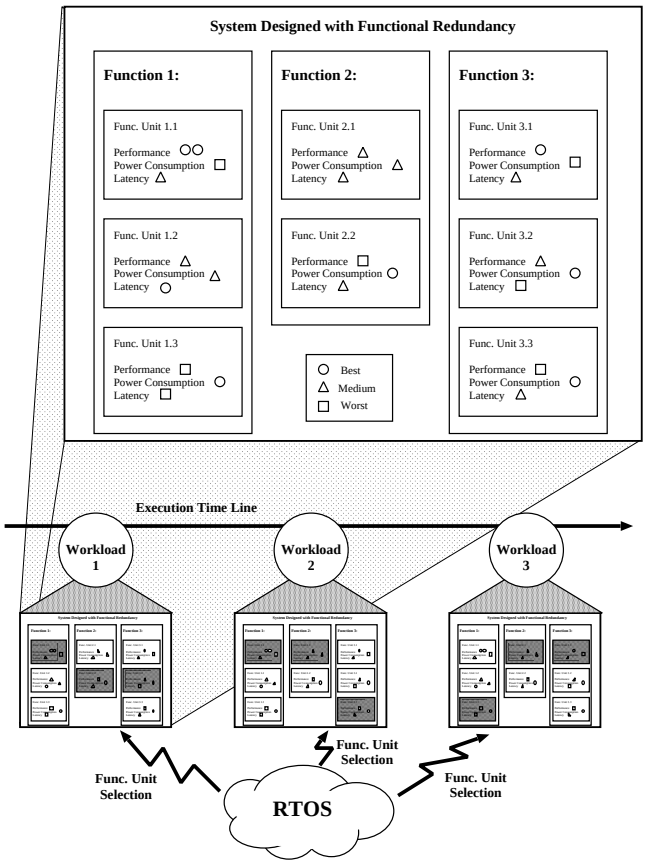

Figure 2. Functional Redundancy: as the execution workload changes, the RTOS will schedule different parts of the circuit in order to deliver the required performance and meet timing constraints, minimizing power consumption and maximizing system's flexibility.

A general system with Functional Redundancy (three different cores for system function 1; two different cores for function 2; and three cores for function 3) is shown in Figure 2. In a given moment, the system will be in one of all possible different configurations, in which one core of each function will be active. During task's execution the workload will change (Workload 1, Workload 2, and so on). When the RTOS detects the change of workload it can dynamically specify each cores will be used to execute the task's program. Doing so it is possible to obtain minimum power consumption, maximizing system's flexibility to adapt itself to the environment requirements. *Recompila-*

Circuits	Algorithms
Mult-1	A array multiplier with Booth's algorithm
Mult-2	A Wallace tree multiplier with Booth's algorithm
Mult-3	A multiplier with 7-3 parallel counters
Mult-4	A multiplier with redundant binary adders

Table 1. Specifications of multipliers

Circuits	Area[μm^2](# of cells)	Delay[ns]	Toggle count/Cycle	Energy/Cycle [pJ]
Mult-1	666462.7 (786)	23.76	926.95	65.48
Mult-2	800846.2 (1,067)	15.77	779.78	65.40
Mult-3	978017.0 (1,344)	12.98	604.37	63.97
Mult-4	896409.4 (1,440)	13.95	861.42	87.15

Table 2. Switching activity of multipliers

tion of code may not be necessary[1]*, as portions of circuit instead of instructions are scheduled by the OS, during task's execution.*

Differently from the approach presented in Subsection 2.1, the supply voltage of the IP core is not changed. The difference of cell's characteristics is due to the hardware algorithms used to implement a given function. The internal structure of the IP core determines its switching activity, and consequently, their power consumption (Equation 1). In order to demonstrate the effects of algorithm selection, we show, in Table 2 the differences in power consumption, area and switching activity for four different implementations of a multiplier.

IP cores selection Designers usually face a strict compromise between performance and power consumption of a circuit, while taking into account die size of the IC. As we have mentioned, chip area is no longer a design burden. However, choosing a certain unique combination of IP cores that could, by average, satisfy the performance and power requirements of an application domain becomes very difficult as the number of cores increase.

According to the minimum performance requirements and maximum power consumption, the solution space of efficient IP cores that should make part of a cell library can be specified, Figure 3. A reasonable number of cores may be selected from this library in order to implement a real project, minimizing TAT.

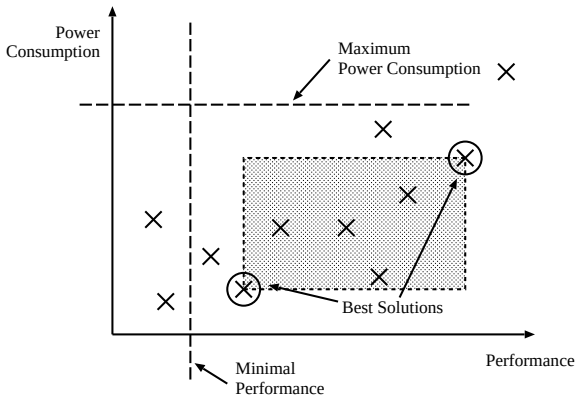

Figure 3. Minimum acceptable performance and power consumption limit the set of different IP cores to be used (gray region).

4. Cores with High Functionality

The effectiveness of Functional Redundancy as a design approach to improve power consumption in dynamic real-time systems was demonstrated for low functionality IP cores[3]. For such architectures, an operation, like a multiply instruction, will be directed to one of the redundant Functional Units in order to minimize power or maximize performance. This is the scheduling of operations to low functionality cores, in which the RTOS has higher control of the performance and power consumption of the system. Another benefit is that recompilation may not be necessary[2].

However, distinct hardware implementations of higher functionality cores can expose very discrepant computation times, latencies, area characteristics, and so on, giving room for much more aggressive implementations and system optimizations.

Let us take as a reference the Fast Fourier Transform (FFT), one of the most widely used digital signal pro-

[1]This is true for superscalar architectures or even VLIW machines endowed with some dynamic compilation scheme (in hardware).

[2]VLIW systems should be endowed with some sort of dynamic compilation mechanism for FU's with different latencies.

Processor	Datapath width (bits)	1024-pt Exec Time (μsec)	Power (mW)	Clock Freq (MHz)	Norm Area (mm^2)
Dassault Electronique	12	10.2	15,000	25	240
Cobra, Col. State	23	9.5	7,700	40	1,104+
CNET, E. Bidet	10	51	300	20	100
M. Wosnitza, ETH	32	80	6000	66	167
Spiffee[1], $V_{dd} = 3.3V$	20	30	845	173	25
Spiffee[1], $V_{dd} = 1.1V$	20	330	9.5	16	25

Table 3. Comparison of characteristics for 1024-Point Complex FFT's.

cessing (DSP) algorithms. As with most DSP algorithms, FFT's make frequent accesses to data in memory. A program implementation of such function results in low performance, high energy consumption (mainly due to memory accesses), whereas specific FFT circuits can greatly optimize these variables. The characteristics of some implementations for a 1024-point complex FFT is illustrated in Table 3.

In this section we analyze the problems related to the use of cores with higher functionalities, i.e. IP cores whose implementation is optimized for a given function or computational task, like cores for signal processing filters, encoder/decoders for audio and video, and so on.

IP Integration and Compiler Support Building SoCs is much more than matching pieces in a Lego game. That is why we have not seen much SoCs built up to now, but we believe that these problems will be solved soon. However the use of IP cores is not only limited to interface and protocol problems. Cores with same functionality may be based on completely different computation models, e.g. control flow, data-flow, and so on. When taking cores from a commercial library or generating a custom one, the core's description must be provided for automatic compiler generation.

In order to get all benefits of functional redundancy as a system design approach the task's load must be infered during execution. The compiler or the user, who better knows the application it is designing, must provide sufficient information to the RTOS about variables which influence time of computation and computation load of different sections in a task. Supose we have a signal processing task that implements a filter and has a loop controlled by a variable n. During compilation it may not be possible to know the value or the range n can have, only at execution time. Thus, during execution of the task, the OS just has to infer n to make a better decision about the system's core configuration. Another example is code with dynamic references.

Regarding code generation for different kinds of cores the following problems arise:

- **Execution Architecture Schemes** This characteristic may lead to uncompatibility of code, scheduling problems and stalls due to data dependency. For instance, a redundant system with FFT cores with a four-stage pipeline and another with two stages.

- **Program Code Size** In order to satisfy protocol and structural differences between redundant cores we may generate code for each core separately. Although it can lead to better performance and scheduling of instructions and function calls, it may increase code size to unacceptable terms. In this case a Branch-and-Bound like algorithm could satisfy the required constraints.

5. Summary and Discussions

The use of Functional Redundancy is not attached to any computation model (SISD, SIMD, MISD or MIMD) nor the use or not of clock in the circuit nor any special processor architecture, like Superscalar, VLIW, etc., so the technique can be applied to any of these systems. However, as the parallelism increases, like in SIMD or MISD, greater challenges for efficient OS controlled scheduling of instructions and tasks arise. Also it does not conflict with any other techniques presented in Section 2, like variable voltage hardware and power supply cut off and its derivatives. Indeed, it can improve even more the power consumption of the system if used in combination with any of those techniques.

Different hardware implementations, specially for higher functionality cores, can expose very discrepant computation times, parallelism and latencies. The compiler or the system architecture has to be able to handle different latencies of IP cores, in order to maintain correctness of the program and not to include other bottlenecks due to data dependency, limiting performance. We do not have an idea of how big is this impact, being this one of our future works.

The flexibility to change system characteristics on-the-fly, making it possible to obtain even lower power consumptions while meeting the time constraints of a task, should be a much important system specification than circuit area

6. Conclusions

In this paper we discussed the impact of the use of high functionality IP cores on software development for a system designed with Functional Redundancy. Functional Redundancy allows the exchange of system characteristics, orchestrated by a RTOS, even after system integration and chip fabrication. The basic idea is to have in the same circuit, different instances (IP cores) with same functionality, but with different performance-energy consumption characteristics. An RTOS aware of the functional redundancy present in the circuit will try to schedule parts of the circuit (cores) in order to execute a given task's section of code.

The main contribution of this approach is to make it possible to allocate on-the-fly a given set of cores aiming at high performance computation or low power consumption according to the actual task load in dynamic systems. So the designed system can expose various performance and power consumption characteristics from the possible combinations of cores with respect to each specific section of code or function in the system.

The use of high functionality IP cores impose some drawbacks due to different cores' computation models, protocols of communication and execution architecture schemes. Although different object codes can be generated to accommodate these differences, achieving better performance, program code size may become a big problem.

If the number of tasks or functions demanding different performance/power characteristics or the number of redundant core instances are big[3], the problem of scheduling may become untreatable, so low complexity scheduling algorithms or heuristics are necessary to explore the new system flexibility disclosed by our approach. We are also going to tackle this problem.

References

[1] B. M. Baas. A low-power, high-performance, 1024-point FFT processor. *IEEE Journal of Solid-State Circuits*, 34(3):380–387, March 1999.

[2] D. Brooks and M. Martonosi. Dynamically exploiting narrow width operands to improve processor power and performance. In *Proc. of International Symposium on High-Performance Computer Architecture (HPCA-5)*, January 1999.

[3] V. M. G. Ferreira and H. Yasuura. Functional redundancy for dynamic exploitation of performance-energy comsumption trade-offs. To be presented in the 13^{th} Symposium on Integrated Circuits and Systems Design, Brazil, September, 2000.

[4] J. Heeb. Next generation intel StrongARM technology overview. In *Proc. of International Symposium on Low-Power and High-Speed Chips*, April 2000.

[5] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava. Power optimization of variable voltage core-based systems. In *Proc. of the 35^{th} Design Automation Conference*, pages 176–181, June 1998.

[6] A. Inoue and H. Yasuura. Flexible system lsi for low power design and low cost implementation of system-on-chip. In *Proc. of 6^{th} Asia Pacific Conference on Chip Design Languages (APCHDL'99)*, pages 117–123, October 1999.

[7] T. Ishihara and H. Yasuura. Optimization of supply voltage assignment for power reduction on processor-based systems. In *Proc. of the Workshop on Synthesis and System Integration of Mixed Technologies (SASIMI97)*, pages 51–58, December 1997.

[8] T. Okuma, T. Ishihara, and H. Yasuura. Real-time task scheduling for a variable voltage processor. In *Proc. of The 12^{th} International Symposium on System Synthesis (ISSS'99)*, pages 24–29, November 1999.

[9] P. R. Panda and N. D. Dutt. Reducing address bus transitions for low power memory mapping. In *Proc. of ED&TC96*, pages 63–67, 1996.

[10] M. Srivastava, A. P. Chandrakasan, and R. W. Brodersen. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. In *IEEE Transactions on VLSI Systems*, volume 4, pages 42–55, 1996.

[11] Transmeta Inc. The crusoe processor. http://www.tensilica.com.

[12] T. Xanthopoulos and A. Chandrakasan. A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization. In *Proc. Symposium on VLSI Circuits*, June 1999.

[3] Normally the number of instances will not be so large as then, area would become a design constraint.