

論文 / 著書情報
Article / Book Information

題目(和文)	プロセッサおよびSoCの低電力機能設計
Title(English)	Low power functional design of microprocessor and SoC
著者(和文)	伊藤雅之
Author(English)	Masayuki Ito
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第8821号, 授与年月日:2012年3月26日, 学位の種別:課程博士, 審査員:前島 英雄
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第8821号, Conferred date:2012/3/26, Degree Type:Course doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Low Power Functional Design of Microprocessor and SoC

Masayuki Ito

Tokyo Institute of Technology

March, 2012

TABLE OF CONTENTS

	Page
CHAPTER 1: Introduction	1
1.1 Expectations about Low Power Microprocessor and SoC	1
1.2 Low Power Functional Design and Its Value	4
1.3 Power Reduction of Microprocessor and SoC.....	6
1.4 Outline of the Thesis.....	9
CHAPTER 2: Low Power Design of Functional Units	13
2.1 Introduction	13
2.2 Multiplier by Operand Decomposition.....	16
2.2.1 Multiplier Architectures.....	16
2.2.2 Previous Work	20
2.2.3 Low Power Multiplication Algorithm	20
2.2.4 Implementations and Evaluations.....	28
2.3 Square Rooting by Iterative Multiply-additions.....	33
2.3.1 Newton-Raphson Method.....	33
2.3.2 New Square Rooting Algorithm	34
2.3.3 Comparisons	39
2.4 Efficient Initial Approximation for Division.....	42
2.4.1 Conventional Multiplicative Division	42
2.4.2 New Initial Approximation Method	46
2.4.3 Speed-up Technique on Convergence	48
2.4.4 Comparisons	50

2.5 Summary of Chapter 2.....	53
CHAPTER 3: Low Power Design of CPU and Cache Memory.....	55
3.1 Introduction	55
3.2 Resume Standby SRAM.....	56
3.3 Core Standby for CPU	59
3.4 Summary of Chapter 3.....	65
CHAPTER 4: SoC Design for Dynamic Power Reduction.....	67
4.1 Introduction	67
4.2 Dynamic Clock Control.....	69
4.2.1 Background.....	69
4.2.2 Dynamic Clock Stop for Functional Modules	70
4.2.3 Dynamic Clock Stop for Bus Modules	72
4.2.4 Evaluation Results	73
4.3 Partial Clock Activation for Mobile SoC	75
4.3.1 Background.....	75
4.3.2 Dynamic Power Reduction by Partial Clock Activation	76
4.3.3 Evaluation Results	78
4.4 Summary of Chapter 4.....	79
CHAPTER 5: SoC Design for Static Power Reduction.....	81
5.1 Introduction	81
5.2 Components for Multi-power Domains.....	83

5.3 Hierarchical Power Domain Control of Multi-core SoC.....	90
5.3.1 Background.....	90
5.3.2 Multi Power Domain SoC Architecture and Implementations.....	92
5.3.3 IP-MMU	97
5.4 Independent Power-off Control of CPU and SRAM of Multi-core SoC.....	102
5.4.1 Background.....	102
5.4.2 Multi-core SoC Architecture	103
5.4.3 CPU Power Modes	105
5.4.4 Implementation of Multi-core SoC.....	106
5.4.5 Evaluation Results	110
5.5 Summary of Chapter 5.....	112
CHAPTER 6: Conclusions	113
6.1 Summary of the Thesis	113
6.2 Perspective.....	117
ACKNOWLEDGEMENTS.....	119
REFERENCES	121
RELATED PAPERS AND PRESENTATIONS.....	125

CHAPTER 1: Introduction

1.1 Expectations about Low Power Microprocessor and SoC

The design of complex chips has undergone a series of revolutions during the last twenty years. In the 90's, the exponential increase in chip density drove the adoption of RTL (Register Transfer Level) design and logic synthesis by EDA (Electronic Design Automation) tools providing a dramatic increase in a designer's productivity. In this era, power has been a second order concern in chip design, following first order issues such as cost, area, and frequency. Figure 1.1 shows a microprocessor chip developed in 250nm process technology [1]. This chip's development target is mainly embedded use for game processing applications and requires both high-performance and low-power. The low-power motivation is mainly for packaging and cooling costs. Submicron technology has posed a new set of design problems. We can implement millions of gates on a reasonably small die, leading to a power density and total power dissipation that are at the limits of what packaging, cooling, and other infrastructure can support.

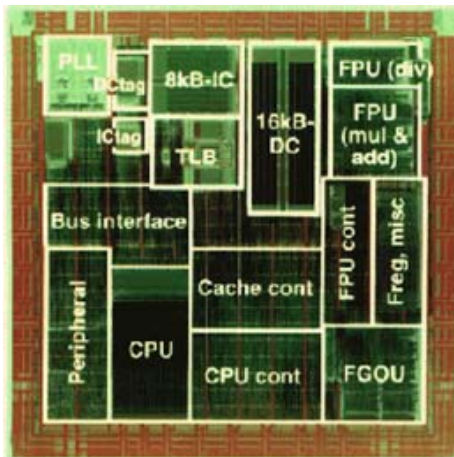
In the past decade, the era of tens or hundreds of million gates on one SoC, the result was that IP (Intellectual Property) and design reuse became accepted as the only practical way to design large chips with relatively short development time. Nowadays, every SoC design employs substantial IP in order to take advantage of the ever increasing density offered by deep sub-micron technology. As process technology shrunk to 90nm and 65nm designs, leakage current is nearly as large as dynamic current under some conditions. These changes are having a significant effect on how chips are designed. Design for low power has started to change how designers approach complex SoC

designs. For most SoC designs, the power budget is the most important design goal of the project. Exceeding power budget can be fatal to a project, whether it means moving from a cheap plastic package to an expensive ceramic one, resulting in unacceptably poor reliability due to excessive power density, or failing to meet the required battery life.

Already, the total power consumption of microprocessor chips presents a significant problem for servers. Infrastructure costs such as power and cooling can equal the cost of the computers themselves. For battery-powered hand-held devices, the absolute power numbers are smaller but the problem is just as serious. For battery-powered devices, which comprise one of the fastest growing segments of the electronics market, the leakage in deep submicron processes is a major problem.

To combat this, designers are using aggressive approaches at every step of the design process, both by software and by hardware, and from architecture to implementation. These approaches include power gating, where blocks are powered down when not in use, and multi-threshold libraries that can trade-off leakage current for speed. Battery life for these devices has declined as features have been added faster than power has been reduced. For virtually all applications, reducing the power consumed by SoCs is essential in order to continue to add performance and features and grow these businesses. The power density of the highest performance chips has grown to the point where it is no longer possible to increase clock speed as technology shrinks. As a result, designers are designing multi-processor chips instead of chips with a single, ultra-high speed processor. Figure 1.2 shows a heterogeneous multi-core SoC for mobile chip [2] that was developed with 65nm process. For the development of this SoC, the power

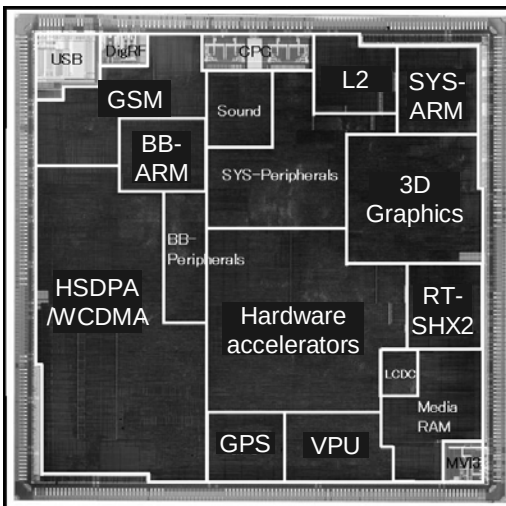
consumption number is by far the most important target, as it directly gives significant impact on the specification for the final product.



Chip Features

Process	0.25um
CPU	SH4 200MHz
Size	7.6mm × 7.6mm
Performance	360 MIPS
Power	1.2W
Voltage	1.8V

Figure 1.1 Processor chip in 1998



Chip Features

Process	65nm
CPU	SH-X2 500MHz ARM1176 500MHz ARM926 167MHz
Size	9.3mm × 9.3mm
Performance	≅ 2000 MIPS
Power	< 100mW
Voltage	1.2V

Figure 1.2 Mobile SoC in 2009

1.2 Low Power Functional Design and Its Value

Design hierarchy starts from the specification level to implementation level. This thesis focuses on functional-design level techniques for low power consumption for microprocessors and SoCs. How to achieve the specification is the main point in this functional-design level and any individual technique can contribute to a large power consumption reduction, which increases the value of microprocessors and SoCs. In the upper specification level (architecture-level in case of microprocessor design case), there are no big differences when microprocessors or SoCs achieve the specified target at a similar performance level. The next level of design hierarchy is functional-design level, or micro-architecture level in case of microprocessor design. Lower design hierarchy levels are logic design level, circuit design level and layout-design level. At these levels, each individual low power technique is very difficult as most of the designs can be automatically handled by EDA tools.

The value of microprocessors and SoCs from the viewpoint of power consumption can be summarized into three categories, namely maximum power, average power, and idle power. Power reduction for these three categories enhances the value of microprocessors and SoCs.

Maximum power is dissipated power when the maximum number of active gates is reached. Maximum power should be estimated at the worst-case even if it is unrealistic. Reducing the maximum power will produce such values as cost reduction by adopting

cheap plastic packaging instead of expensive ceramic ones and cheap power supply IC (Integrated Circuit). It may also be possible that a cooling fan is no longer necessary.

Average power is dissipated power when gates are activated at an average rate and is combined with leakage power. Average power decides the battery life of mobile products. Reducing the average power will produce such values like prolonged music playback time per battery charge. This will make the product more attractive to the end-user and thus there is a possibility that the chip can be sold at a higher price.

Idle power is dissipated power when the chip is in idle state and the main source of this is leakage. Idle power determines the battery life of mobile products when not in use. Reducing the idle power will produce such values as longer standby time of mobile phones per battery charge. This power reduction may directly show an improvement in the final product specification and thus there is a possibility that the chip will be adopted by many handset makers and can be sold at a higher price.

1.3 Power Reduction of Microprocessor and SoC

Power now has become the most important aspect in the design of general purpose processors and SoCs. Digital CMOS circuits consume relatively less power and static CMOS logic is widely used because of its robustness, and its suitability for design automation [3]. In static CMOS, the dynamic power dissipation of a circuit depends primarily on the number of logic transitions per unit time [4]. The total power for a processor or an SoC design consists of dynamic power and static power. Dynamic power is the power consumed when the device is active, that is, when signals are changing values. Static power is the power consumed when the device is powered up but no signals are changing values. In CMOS devices, static power consumption is due to leakage. How to reduce power dissipation is the theme of this thesis.

The first and primary source of dynamic power consumption is switching power; the power required to charge and discharge the output capacitance on a gate. We can describe the dynamic power by the following expression:

$$P_{dynamic} \propto \alpha \cdot F \cdot C \cdot V^2,$$

where α is average switching activity, F is frequency, C is total capacitance, and V is supply voltage. There are a number of techniques at the architecture, logic design, and circuit design level that can reduce the power for a particular function implemented in a given technology. These techniques focus on the voltage and frequency components of the equation, as well as reducing the data-dependent switching activity.

There are a variety of design techniques for minimizing switching activity, which effectively lowers switching activity for the gates involved. Another approach for lowering dynamic power is clock gating. Driving the frequency to zero drives the power to zero. Some form of clock gating is used on many SoC designs.

The effective way to reduce dynamic power is to reduce the supply voltage, as $P_{dynamic}$ is proportional to V^2 . Over the past several decades, as semiconductor technology scaled, Vdd has been lowered from 5V to 3.3V to 2.5V to 1.2V and will be lower in the future. The trouble with lowering Vdd is that it tends to lower Ids, the drive current of the transistor, resulting in slower speeds. In order to maintain performance, we should lower Vt, the threshold voltage, but the result is that we raise leakage current. Until recently, this was a reasonable technique, since static power from leakage current was so much lower than dynamic power. But with 90nm or lower process technologies, we are getting to the point where static power can be as big a problem as dynamic power, and we need to tackle this conflict.

Remember that the dynamic power is also proportional to C, the capacitance to the gates involved. In order to reduce C for dynamic power dissipation, one of the effective ways is to adopt a good micro-architecture or to adopt a good algorithm and achieve the specification by a smaller number of transistors.

We can describe the static power by the following expression:

$$P_{static} \cong I_{leak} \cdot V,$$

where I_{leak} is leakage current, and V is supply voltage. There are several approaches to minimizing leakage current. One technology is known as multi-Vt, using high Vt cells wherever performance goals allow and low-Vt cells where necessary to meet timing. Another technique is to create multiple-power domains inside one chip and shut down the power supply of domains when they are not active. This approach is known as power gating. There exists a third technique that has a big effect: achieving the same function and performance by a smaller area, in other words, by a smaller number of transistors.

1.4 Outline of the Thesis

This thesis explores how power consumption, both dynamic power and static power, can be reduced at the functional-design level in the development of microprocessors and SoCs. This thesis is organized as follows after the introduction in Chapter 1.

Chapter 2 discusses low-power functional design of arithmetic units. Since multipliers are rather complex circuits and typically must operate at a high system clock rate, reducing logic transitions and thus the power dissipation of multipliers is a key to satisfying the overall power budget. Division and square rooting are also important functions that are usually supported at instruction-level and use iterative multiplication. In this chapter, after an introduction in section 2.1, we propose a low-power multiplier by a novel operand decomposition scheme in section 2.2. Then in section 2.3, we propose a square rooting algorithm by iterative multiply-additions. Then in section 2.4, efficient initial approximation schemes for division are discussed. In each section, we review the conventional methods first and propose our methods. We also compare these methods from the viewpoint of power consumption together with other viewpoints such as gate size or performance. A summary is shown in section 2.5.

Chapter 3 discusses low-power functional design of CPU-cores using low leakage onchip-SRAM. In this chapter, after an introduction in section 3.1, low-power resume-standby SRAM is introduced in section 3.2. Then, a new CPU power saving mode called core-standby mode is described in section 3.3. In Core standby mode, CPU logic is

powered-off while cache and internal-SRAM keeps powered to “resume” level at which the data can be retained. Section 3.4 summarizes this chapter.

Chapter 4 discusses dynamic power reduction techniques for SoCs. Clock control is an important issue to reduce dynamic power consumption. In this chapter, after an introduction in section 4.1, dynamic clock control scheme is described in section 4.2. The main point here is that a human designer specifies at functional design level when to stop clock delivery for a group of logic. For the more sophisticated dynamic power reduction of clock-gating, partial clock activation method is described in section 4.3. The clock activation is linked with the corresponding power-domain activation and we explain this technique with our mobile SoC in section 4.3. Section 4.4 summarizes this chapter.

Chapter 5 discusses static power reduction techniques for SoCs. Leakage power amount and ratio increases in sub-100nm generation SoCs. One good solution is power domain separation to shutdown unused circuits. After an introduction in section 5.1, we explain some necessary components for multi-power domain technique in section 5.2. In section 5.3, we describe our multi-power domain SoC. Our single chip SoC contains plural sub-systems and power domain separation that amounts to twenty. Power for unused domains should be off even for a short period for leakage reduction. In section 5.4, we describe our homogeneous multi-core SoC. In order to achieve leakage power savings for the multi-core SoC, the 8 CPU cores and 8 RAMs are assigned to split power supply domains so that the power supply of unused CPU cores can be cut off. For the power-off, “RAM resume” mode can be selected for fast restart on wake up. Section 5.5 summarizes this chapter.

Chapter 6 concludes this thesis. The thesis is summarized with the perspective for this research theme, low-power functional design techniques of microprocessors and SoCs.

CHAPTER 2: Low Power Design of Functional Units

2.1 Introduction

Nowadays, many digital signal processing systems are targeted at portable, battery-operated systems, so power dissipation is the primary design constraints. Since multipliers are rather complex circuits and typically must operate at a high system clock rate, reducing logic transitions and thus the power dissipation of multipliers is a key to satisfying the overall power budget for functional units.

In this chapter, we first propose a novel low power multiplication algorithm using operand decomposition. Unlike Booth's encoding schemes [5] that modify only one operand to reduce the number of partial products, our scheme decomposes both a multiplicand and a multiplier. A final product is generated through two multiplications. The decomposition is done to reduce the number of ones in total partial products. As a result, logic transitions are reduced, achieving lower power dissipation.

Division and square root are essential operations in many scientific and engineering applications. With availability of high-speed multiplication units, multiplicative algorithms have become advantageous to fast calculation of division and square root. In general, such methods begin with an initial approximation to the desired value, which is then improved by an iterative algorithm through some multiplications and additions. Therefore, efficient initial approximation methods and fast converging algorithms hold the key of achieving the desired precision faster at lower hardware cost. Here, performance and power are in trade-off relation; when we get initial approximation

value of higher precision using larger lookup tables, we can achieve high performance at the cost of larger power of lookup tables. We deal with division and square root operations on the mantissa parts of floating point numbers. We assume an arithmetic unit on which one multiplication $(A \times B) + C$ can be executed in one cycle. This operation takes no more time than a multiplication by incorporating addition into the adder tree of the multiplier. We propose three new methods, an initial approximation method for division, a higher order converging division algorithm accelerated through table look-up, and a new converging algorithm for square root. They are all suitable for implementation on a multiply-add unit.

Many multiplicative square root algorithms have been presented [6]. Among them, the multiplicative Newton-Raphson method has been widely used. It involves three multiplications per iteration. In section 2.3, we propose a new algorithm for square root. It calculates $\sqrt{X}$ in a different manner from conventional converging algorithms. It does not directly refine an initial approximation to the final result, but refines an initial approximation to the difference between the exact $\sqrt{X}$ and an approximation to $\sqrt{X}$. The iterative formula of our algorithm consists of only one multiply-accumulate operation, which can be executed in one cycle on a multiply-add unit.

For the generation of initial approximations, lookup tables are commonly used. The simplest way is directly reading an initial approximation to the desired value through table look-up using some most significant bits of an operand as an index. Besides this direct approximation method, a linear approximation method can be employed. In this case, the table entry is the two coefficients of the linear function. Evaluation of the initial

approximation from the linear function requires one multiply-accumulate operation, and the result is roughly twice as many bits of accuracy as that achieved by the direct approximation method. In section 2.4, we propose an efficient approximation method to form an initial approximation to the reciprocal of a divisor. It requires only one multiply-accumulate operation and its accuracy is higher than that achieved by the original linear approximation.

Multiplicative division algorithms, represented by the Newton-Raphson method and Goldschmidt's algorithm use iterations to refine an approximation to the desired result. The typical rate of convergence doubles with each iteration. An extended algorithm of the Newton-Raphson method has been presented, where an arbitrary higher order convergence can be achieved [7]. In section 2.4, we also develop a scheme to speed up the convergence of the algorithm, and propose an accelerated higher order converging algorithm. The combination of this converging algorithm and our initial approximation method achieves very high performance. A single iteration of the converging algorithm is enough for double-precision division.

This chapter is organized as follows. In section 2.2, we propose a low-power multiplier by operand decomposition. Then in section 2.3, we propose a square rooting algorithm by iterative multiply-additions. Then in section 2.4, we discuss efficient initial approximation for division with a higher order converging algorithm. In each section, we review the conventional methods first and propose our methods. We also compare these methods from the viewpoint of power consumption together with other viewpoints such as gate size or performance. A summary of this chapter is shown in section 2.5.

2.2 Multiplier by Operand Decomposition

2.2.1 Multiplier Architectures

In this subsection, we first review regular topologies for multipliers. They can be classified as either array topologies or tree topologies [8][9]. Then, we present some existing architectural and algorithmic approaches to reduce the consumed power.

Array Multipliers

A simple array multiplier consists of rows of (3,2) counters. In the simple array, each row of (3,2) counters adds a partial product (PP) to the partial sum, generating a new partial sum and a sequence of carries. Figure 2.1 shows a block diagram for a single array multiplier. The delay of the array to produce the final partial sum and carry is $(2n-4)$ XOR delays for an n -bit multiplier.

The delay required to generate the result for the simple array can be halved by performing two additions in parallel. The final two partial sums and two carries are combined using a (4:2) compressor. Figure 2.2 shows a block diagram for a double array multiplier. The delay of the double array to produce the final partial sum and carry is $(n-1)$ XOR delays.

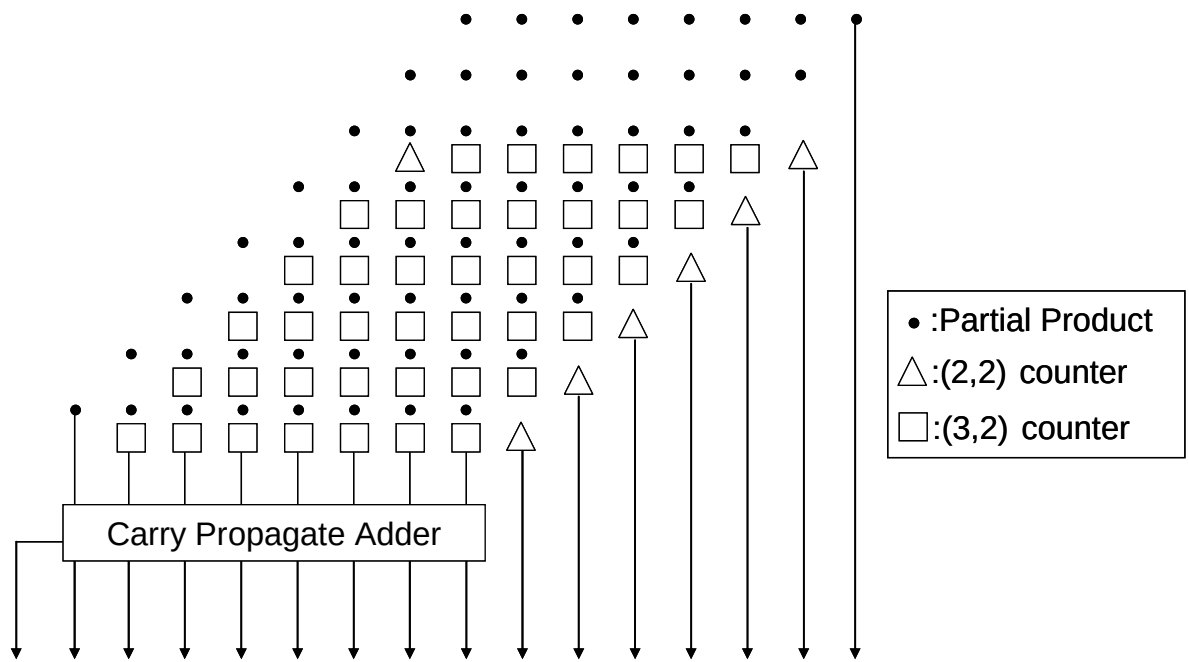


Figure 2.1 An 8-bit single array multiplier

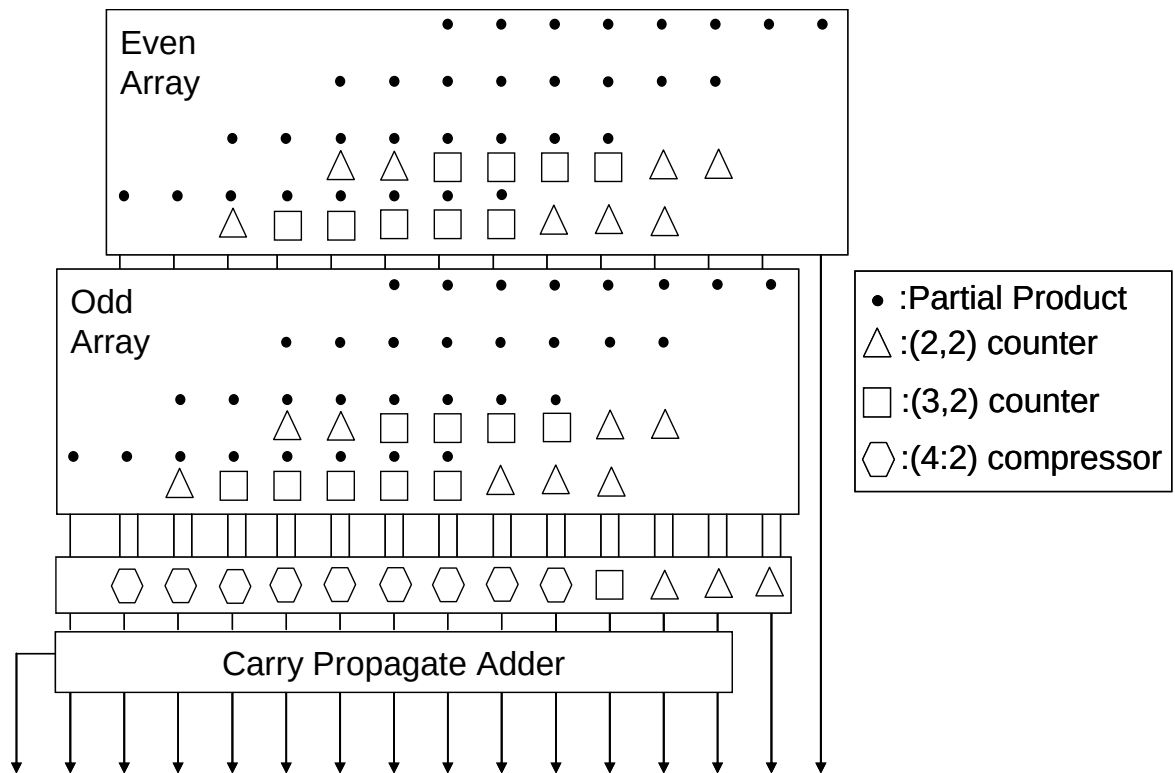


Figure 2.2 An 8-bit double array multiplier

Tree Multipliers

Trees are very fast structures for summing PPs. The Wallace tree topology based on (3,2) counters [10] has an irregular structure which causes many spurious transitions due to imbalances of arrival time for each counter. A (4:2) compressor based multiplier forms a binary tree topology which leads to a fast, symmetric and regular design. Thanks to this regularity, spurious transitions can also be reduced [11]. Figure 2.3 shows a block diagram for a (4:2) compressor based multiplier. The delay of this multiplier to produce the final partial sum and carry is $3\log_2(n/2)$ XOR delays for an n -bit multiplier.

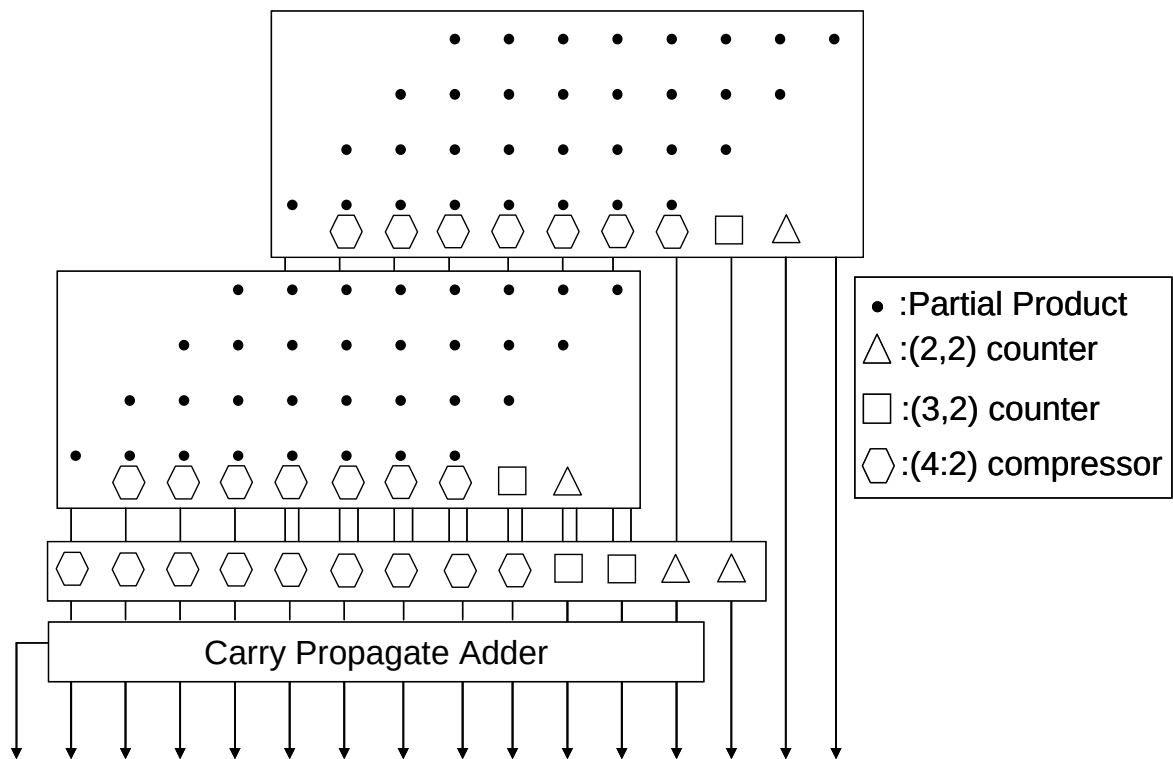


Figure 2.3 An 8-bit (4:2) compressor based multiplier

2.2.2 Previous Work

Reducing the power consumption for multiplication can be tackled at different levels of the design hierarchy. Algorithmic level approaches can give a significant overall power reduction, but only a few approaches have been reported. Some existing solutions at this level for low-power multipliers try to reduce the switching activity by observing operands.

A dynamic operand exchange method observes the sign bits of the operands of two consecutive multiplications and conditionally exchanges the operands [12]. A dynamic operand complementation scheme utilizes the identity $(-1)^2 = 1$ in addition to the dynamic operand exchange [13]. For these two schemes, the required logic for judgment and selection for operand exchange is not trivial and the worst case delay is increased.

If the multiplication is divided into clusters of smaller multipliers with registers, clock gating techniques can be applied to clusters that produce a result of zero [14]. This approach is limited to small multipliers or applications that naturally contain a high probability of inputs with several consecutive zeros, where clusters can be turned off frequently.

2.2.3 Low Power Multiplication Algorithm

In this subsection, we propose a novel low power Multiplication Algorithm for Switching Activity Reduction through operand decomposition (MASAR for short). For

simplicity throughout this subsection, we try to compute unsigned n -bit integer multiplications; operands X and Y have n bits and a product $M = X \times Y$ has $2n$ bits. Let

$$X = [x_{n-1}x_{n-2} \cdots x_1x_0], \text{ and}$$

$$Y = [y_{n-1}y_{n-2} \cdots y_1y_0].$$

We decompose operand X and Y into 4 numbers A , B , C , and D to reduce the number of ones in partial products. For this purpose we generate four numbers below in Step 1 of MASAR:

$$A = [-1a_{n-1}a_{n-2} \cdots a_i \cdots a_1(a_0 + 1)],$$

$$B = [b_{n-1}b_{n-2} \cdots b_i \cdots b_1b_0]$$

$$C = [c_{n-1}c_{n-2} \cdots c_i \cdots c_1c_0], \text{ and}$$

$$D = [d_{n-1}d_{n-2} \cdots d_i \cdots d_1d_0], \text{ where}$$

$$a_i = \overline{x_i} \wedge \overline{y_i}, b_i = x_i \wedge y_i, c_i = \overline{x_i} \wedge y_i, \text{ and } d_i = x_i \wedge \overline{y_i} \text{ for } (n-1 \geq i \geq 0), \text{ with}$$

the property that

$$X \times Y = (C \times D) - (A \times B). \quad (1)$$

To help understand this equation (1), we introduce a temporal number $T = -A$.

Then,

$$T = [t_{n-1}t_{n-2} \cdots t_1t_0], t_i = x_i \vee y_i.$$

It is easy to see that $X = B + D$, $Y = B + C$, and $T = B + C + D$. Then

$$X \times Y = (B + D) \times (B + C)$$

$$= (C \times D) + (B + C + D) \times B$$

$$= (C \times D) + (T \times B)$$

$$= (C \times D) - (A \times B),$$

And the equation (1) is proved. There is another important relation between these four numbers. Since a_i , b_i , c_i and d_i are generated by ANDing x_i and y_i or their inversions, $a_i + b_i + c_i + d_i$ is always 1 for each i . When we denote the probability of $a_i = 1$ by $\Pr(a_i)$,

$$\Pr(a_i) = \Pr(b_i) = \Pr(c_i) = \Pr(d_i) = 1/4, \text{ while}$$

$$\Pr(x_i) = \Pr(y_i) = 1/2.$$

In Step 2 of MASAR, we generate and compress PPs for each multiplication, $C \times D$ and $A \times B$. About twice as many PPs as in conventional multipliers are generated. Let

$$P = A \times B = [p_{n-1,n-2} \cdots p_{i,j} \cdots p_{1,0}], \text{ and}$$

$$Q = C \times D = [q_{n-1,n-2} \cdots q_{i,j} \cdots q_{1,0}].$$

We generate $p_{i,j}$ and $q_{i,j}$ by

$$p_{i,j} = (a_i \wedge b_j) \vee (b_i \wedge a_j), \text{ and}$$

$$q_{i,j} = (c_i \wedge d_j) \vee (d_i \wedge c_j)$$

$$\text{for } (n-1 \geq i > j \geq 0).$$

This partial product reduction can be done by simple OR logic because only one of a_i and b_i is 1, and only one of c_i and d_i is 1. Note also that $p_{i,i} = q_{i,i} = 0$ for $(n-1 \geq i \geq 0)$ by the same reason. Now, the total number of PPs with MASAR is almost the same as conventional multipliers. Here, $\Pr(p_{i,j}) = \Pr(q_{i,j}) = 1/8$ while the probability of one for each partial product of conventional multipliers is $1/4$. Thus the number of logic transitions is reduced.

We need to account for the complementary terms generated by the most significant and least significant bits of A . The n -th position of A is always -1 and additional n PPs are generated. The 0-th position of A is $(a_0 + 1)$ instead of a_0 , which requires a simple modification to the logic to generate the corresponding $(n+1)$ PPs. Tables 2.1 and 2.2 give the total number of PPs and the average number of ones in PPs with and without MASAR. The numbers for MASAR are after PP reduction in Step 2. Figure 2.4 shows an example of our operand decomposition and Figure 2.5 shows the MASAR PP reduction scheme.

After PPs are generated and reduced, PPs for each P and Q are summed up by either an array or tree topology. Then the final partial sum and carry from both P and Q can be combined using (4:2) compressors. Since $-P + Q$ is required instead of $P + Q$, the (4:2) compressors should be modified so that two input signals from P are negated. +2 must also be added as we negate two numbers. This complement can be handled easily by the final carry propagate adder. Figures 2.6 and 2.7 show block diagrams for an array multiplier with MASAR and a tree multiplier with MASAR. Positive PPs and

negative PPs shown in these diagrams correspond to the values $(C \times D + B \cdot 2^n)$ and $(-A \times B - B \cdot 2^n)$, respectively.

Table 2.1 Total number of PPs

Type/Width	8	16	32	64
Conventional	64	256	1024	4096
MASAR	66	258	1026	4098

Table 2.2 Average number of ones in PPs

Type/Width	8	16	32	64
Conventional	16	64	256	1024
MASAR	12	40	144	544

$$\begin{aligned}
 X &= \{ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \} = d'55 \\
 Y &= \{ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \} = d'69 \\
 \\
 A &= \{ -1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \} = d'-119 \\
 B &= \{ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \} = d'5 \\
 C &= \{ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \} = d'64 \\
 D &= \{ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \} = d'50 \\
 \\
 55 \times 69 &= 64 \times 50 + 119 \times 5
 \end{aligned}$$

Figure 2.4 An example of operand decomposition

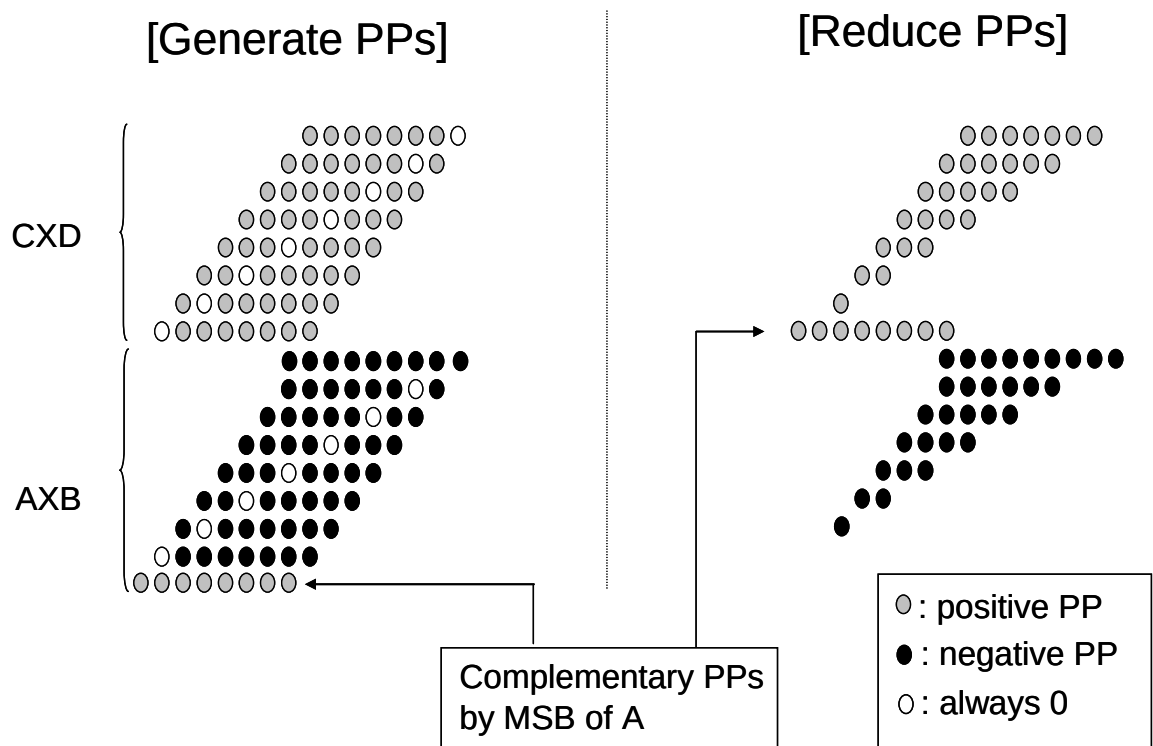


Figure 2.5 PP reduction by MASAR

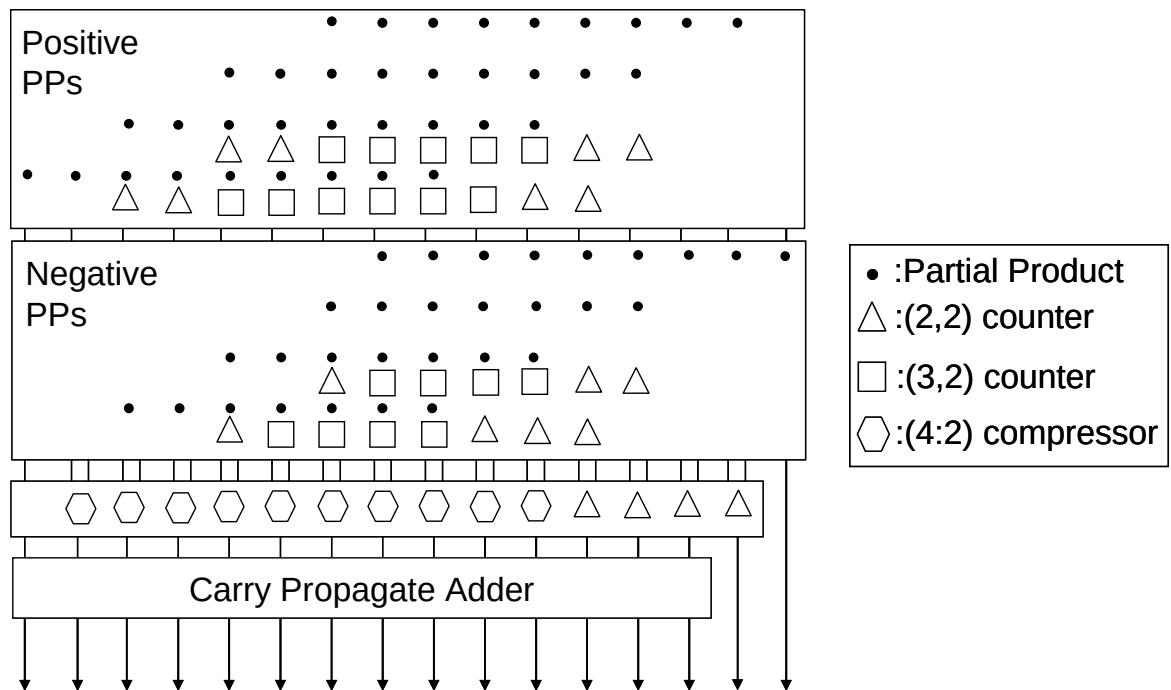


Figure 2.6 An 8-bit array multiplier with MASAR

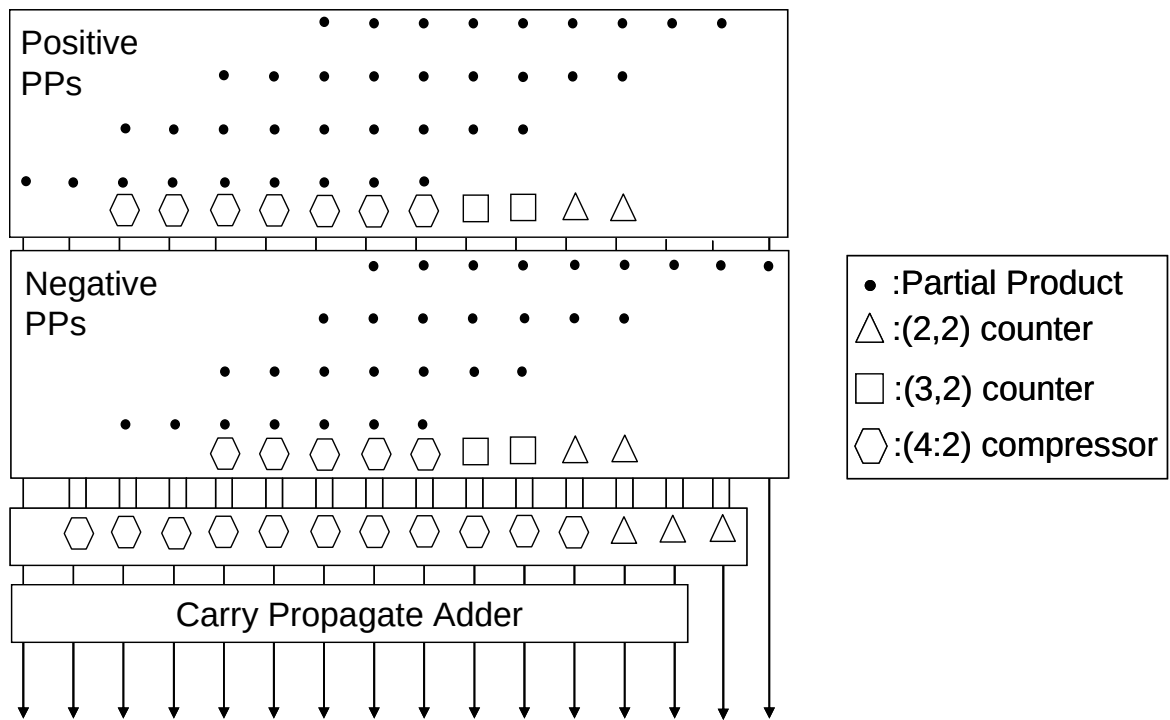


Figure 2.7 An 8-bit tree multiplier with MASAR

2.2.4 Implementations and Evaluations

In this subsection, we compare the proposed multiplier to conventional multipliers in terms of dynamic power, area, and delay. We have applied MASAR to both array multipliers and tree multipliers. Since MASAR generates two arrays, a double array multiplier (D-array) is compared with an array multiplier with MASAR (M-array). For tree multipliers, (4:2) compressor based multipliers without MASAR (C-tree) and with MASAR (M-tree) are compared. Note that MASAR only affects the PP generation and reduction stages and the remainder of the multiplier logic in each comparison are implemented in the same manner. Kogge-Stone adders [15] are adopted as the final carry propagate adder in the multipliers for our evaluations.

Logic Transitions and Dynamic Power

In order to evaluate dynamic power savings with MASAR, we first specified the multipliers in C language, with the logic shown in the Figure 2.8. The partial products were generated using AND gates, or AO22 gates for MASAR. (AO22 is a complex logic gate with four inputs, for computing p_{ij} and q_{ij} .) The remaining logic was implemented using counters and compressors shown in Figure 2.8. The final Kogge-Stone adder was specified using XOR, AND, and AO21 logic. Then the average number of gates that change output value per multiplication was counted. Each multiplier was subjected to 1000 pseudo-random input patterns. We initialized these multipliers to 0 only before the first pseudo-random data. Table 2.3 shows the resulting total number of logic transitions.

This shows that MASAR saves 12% and 15% in array multipliers of 32-bits and 64-bits, respectively, and 12% and 18% for tree multipliers of 32-bits and 64-bits, respectively.

Then Verilog RTL was written and verified using Synopsys VCS HDL Simulator. We synthesized each multiplier with Synopsys Design Compiler using 0.13um high-speed standard cell library at the worst case process corner. We collected switching activity data, without glitches, for these gate-level netlists with the same random data as used in C simulation. The switching activities were then back-annotated to calculate the dynamic power assuming 100MHz clock frequency, at 85°C, with supply voltage of 1.08V.

Figure 2.9 shows power consumption results. MASAR saves 13% and 10% in array and tree multipliers of 64-bit respectively. Thus, MASAR is effective for both array and tree multipliers for sizes of 32-bit width and larger. The actual dynamic power savings correspond fairly closely to reduction in the number of logic transitions calculated by simulation in C in Table 2.3. Note that MASAR can also reduce the number of spurious logic transitions as it reduces the probability of ones in PPs.

Area and Delay

The additional logic required for Step 1 of MASAR is the generation of A , B , C , and D and the complementary logic for the most significant bit and least significant bit of A in Step 1. Recall that $p_{i,j} = (a_i \wedge b_j) \vee (b_i \wedge a_j)$, and $q_{i,j} = (c_i \wedge d_j) \vee (d_i \wedge c_j)$ in Step 2 of MASAR. Generally $n(n-1)$ OR gates are required for the generation of $p_{i,j}$ and $q_{i,j}$ in addition to generating PPs by $2n(n-1)$ AND gates. Here, we can use AO22 gates for

compressing PPs to reduce the number of gates. Thanks to this scheme, the increase in cell area required for implementing the whole multiplier with MASAR can be kept small. It is difficult to reduce the number of gates in conventional multipliers by the same scheme; partial product generation logic and the first level of additions cannot be merged efficiently, because fan-outs of these PPs are at least 2. Table 2.4 presents the corresponding cell area for the multipliers. The required cell area for MASAR is about 10% for multipliers of 16 bits or larger width.

MASAR only affects the critical path delay with the changes in the logic used to generate the partial products. The logic on the critical path of the remainder of the multiplier is the same. Therefore, we synthesized the partial product logic with a tight delay constraint to compare the differences. The additional delay for MASAR is about 0.2ns, which amounts to as much as 8% for the 8 bit multipliers and as little as 2% and 4% overhead for the 64 bit array and tree multipliers respectively. Library wire load models were used for the synthesized netlist delay analysis.

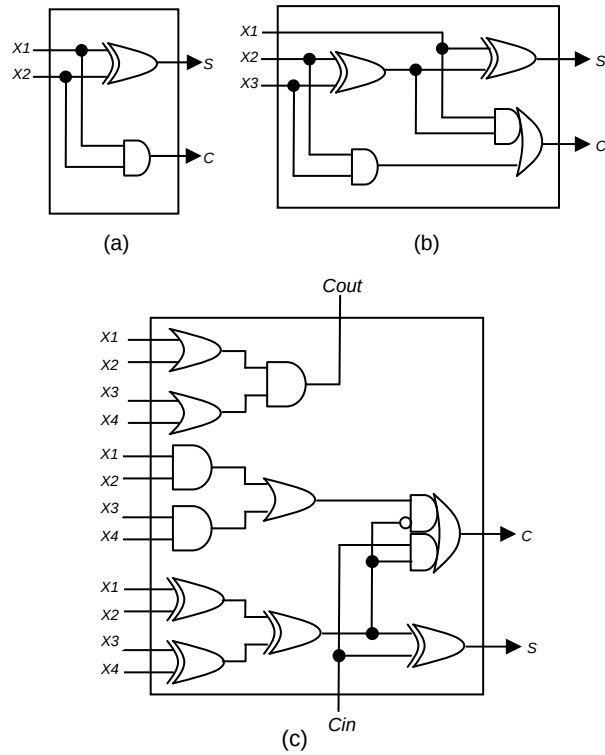


Figure 2.8 Circuits for (a) a (2,2) counter, (b) a (3,2) counter, (c) a (4:2) compressor

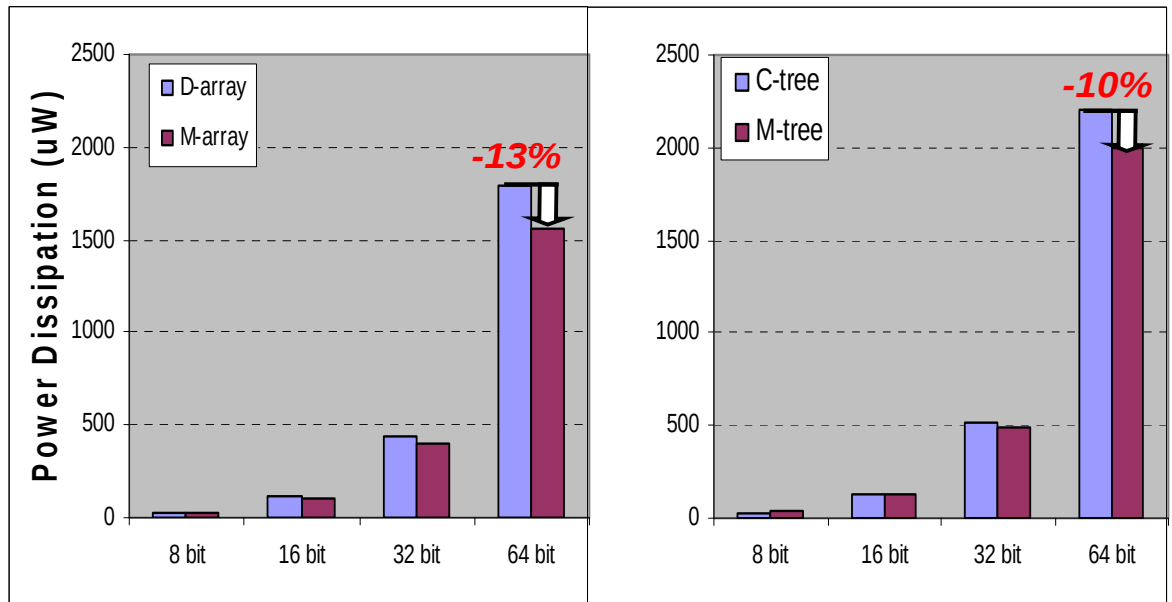


Figure 2.9 Evaluation results of power consumption

Table 2.3 Average number of logic transitions

Type/Width	8	16	32	64
D-array	146	588	2344	9267
M-array	169	572	2069	7864
Ratio (%)	115.8	97.3	88.3	84.9
C-tree	156	671	2722	10770
M-tree	178	663	2400	8850
Ratio (%)	114.1	98.8	88.2	82.2

Table 2.4 Cell area (μm^2)

Type/Width	8	16	32	64
D-array	5315	21380	86108	343890
M-array	6098	23837	95310	376737
Ratio (%)	114.7	111.5	110.7	109.6
C-tree	5212	22056	89310	356790
M-tree	6207	24309	98270	389215
Ratio (%)	119.1	110.2	110.0	109.1

2.3 Square Rooting by Iterative Multiply-additions

We discuss multiplicative square rooting in this section. Among many converging algorithms for square root, the multiplicative Newton-Raphson method for calculating $1/\sqrt{X}$ is widely used. It requires three multiplications per iteration. We first review the multiplicative Newton-Raphson method in subsection 2.3.1. In subsection 2.3.2, we propose a new square rooting algorithm. The iterative formula of our algorithm consists of one multiply-accumulate operation and can be executed in one cycle on a multiply-add unit. It has a property of linear convergence. In subsection 2.3.3, we compare our algorithm to the multiplicative Newton-Raphson method.

2.3.1 Newton-Raphson Method

The multiplicative Newton-Raphson method for calculating $1/\sqrt{X}$ begins with an initial approximation G_0 to the reciprocal of the square root of given X , where $X = [1.x_1x_2 \cdots x_mx_{m+1} \cdots x_b]$. After iterating the converging formula n times, $\sqrt{X}$ can be obtained by the additional multiplication $G_n \times X$.

In the case of the direct approximation method for $1/\sqrt{X}$, we use an m -bits-in, t -bits-out look-up table for producing G_0 . For square root m bits for the index consist of the $(m - 1)$ most significant bits of the mantissa X , i.e., $[x_1x_2 \cdots x_{m-1}]$, and the last bit of the exponent part of the original floating point number. The error $\delta_d = G_0 - 1/\sqrt{X}$ is bounded as

$$|\delta_d| = G_0 - 1/\sqrt{X} < 2^{-m-1} + 2^{-t-2} \quad (2)$$

and the required table is of size $2^m \times t$ bits.

The multiplicative Newton-Raphson iterative formula for calculating $1/\sqrt{X}$ is

$$G_i = \frac{G_{i-1}}{2} \times (3 - X \times G_{i-1}^2), \quad (3)$$

where G_i converges to $1/\sqrt{X}$ quadratically. Each iteration (3) can be executed in three cycles as follows:

1. $Q_{i,1} = X \times G_{i-1}$
2. $Q_{i,2} = (3 - Q_{i,1} \times G_{i-1})$
3. $G_i = \frac{1}{2} \cdot (Q_{i,2} \times G_{i-1})$

After the final (n-th) iteration, one additional multiplication $G_n \times X$ is needed,

which can be calculated in step3 of the final iteration by $\frac{1}{2} \cdot (Q_{n,2} \times Q_{i-1})$ instead of

$\frac{1}{2} \cdot (Q_{n,2} \times G_{n-1})$ Since n iterations of the formula (3) involve 3n multiplications, 3n cycles

are required in adopting the direct approximation.

2.3.2 New Square Rooting Algorithm

In this subsection we propose a new converging algorithm for calculating $\sqrt{X}$, which can be derived from the divisional Newton-Raphson method calculating $\sqrt{X}$ by the following iterative formula:

$$P_i = \frac{1}{2} \cdot (P_{i-1} + \frac{X}{P_{i-1}}). \quad (4)$$

Here, P_0 is an initial approximation to $\sqrt{X}$. This method requires one division in each

iteration. We divide the first iteration of (4), $P_1 = \frac{1}{2} \cdot (P_0 + \frac{X}{P_0})$, into two formulas using

$$K = 2P_0.$$

$$S = \frac{1}{4} - \frac{1}{K^2} \times X \text{ and} \quad (5)$$

$$P_1 = \frac{K}{2} - K \times S. \quad (6)$$

If we can calculate $S_\infty = \frac{1}{2} - \frac{\sqrt{X}}{K}$ from $S_0 = \frac{1}{4} - \frac{1}{K^2} \times X$, the exact $\sqrt{X}$ is obtained by

$\frac{K}{2} - K \times S_\infty$ and iterations of the converging formula (4) are no longer in need. Then,

S_∞ and S_0 hold the following relation:

$$S_\infty - \frac{1}{2} = -\sqrt{\left| \frac{1}{4} - S_0 \right|}. \quad (7)$$

Note that $S_\infty - \frac{1}{2} = -\frac{\sqrt{X}}{K}$ and $\frac{1}{K^2} \times X = \frac{1}{4} - S_0$. (7) shows one of the two solutions of

the following quadratic equation of S_∞ :

$$S_\infty^2 - S_\infty + S_0 = 0. \quad (8)$$

From (8) we can derive an asymptotic equation of S :

$$S_i = S_{i-1}^2 + S_0, \text{ where } \lim_{i \rightarrow \infty} S_i = \frac{1}{2} - \frac{\sqrt{X}}{K}. \quad (9)$$

The new converging algorithm can be summarized as follows using $L = \frac{1}{K^2}$. It has a property of linear order convergence and we call it Algorithm LOC.

[Algorithm LOC]

$$\text{step1: } S_0 = \frac{1}{4} - L \times X$$

step2: for $i = 1$ to n do

$$S_i = S_{i-1}^2 + S_0$$

$$\text{step3: } P = \frac{K}{2} - K \times S_n$$

Here, P is the final result obtained by the algorithm.

In Algorithm LOC, K is an approximation to $2\sqrt{X}$, because $K = 2P_0$, where P_0 is an approximation to $\sqrt{X}$. It is read through table look-up addressed with some upper bits of X . $L (= \frac{1}{K^2})$ must have the same precision as P . It can also be read out of a look-up table addressed with the same bits of X as for the table for K .

Unlike initial approximations for the Newton-Raphson method, we keep both K and $K' = \sqrt{2}K$ and use K' instead of K in step3 of Algorithm LOC, when the exponent part of the original floating point number is odd. Note that L is irrelevant to the exponent part. Therefore, in Algorithm LOC, we use the m most significant bits of X for m -bits-in tables. The iterative formula of Algorithm LOC step2 is a simple multiply-accumulate operation that can be executed in one cycle on a multiply-add unit. The operations of step1 and step3 also can be done in one cycle for each. Hence, when step2 is iterated n

times, the number of required cycles is $(n + 2)$ in total. Let ξ_i be the error of $\frac{K}{2} - K \times S_i$ from $\sqrt{X}$ when step2 is iterated i times;

$$\xi_i = \sqrt{X} - \left(\frac{K}{2} - K \times S_i\right). \quad (10)$$

Then,

$$\xi_0 = \frac{K}{4} \left(1 - \frac{2}{K} \sqrt{X}\right)^2. \quad (11)$$

$$\xi_i = \xi_{i-1} \cdot \left(1 - \frac{2}{K} \sqrt{X}\right) + \frac{\xi_{i-1}^2}{K}. \quad (i \geq 1) \quad (12)$$

(12) shows that LOC has a property of linear convergence. Its coefficient is $\left(1 - \frac{2}{K} \sqrt{X}\right)$. As K is an approximation to $2\sqrt{X}$, a high-precision approximation fastens the speed of convergence. It decreases the initial error ξ_0 at the same time. Thus, the convergent speed depends on the precision of K . When the look-up tables are addressed with the m most significant bits of X , the table for K should contain the value $K = \sqrt{w} + \sqrt{w + 2^{-m}}$ for the subinterval $[w, w + 2^{-m})$, where $w = [1.x_1x_2 \cdots x_m]$. Unless a truncation error of K is considered, the coefficient $\left(1 - \frac{2}{K} \sqrt{X}\right)$ is less than 2^{-m-2} , which shows that the number of correct bits is increased by $(m + 2)$ bits with each iteration.

Now, let the table width for K be t . Then, $\left(1 - \frac{2}{K} \sqrt{X}\right)$ satisfies the following inequality:

$$|1 - \frac{2}{K} \sqrt{X}| < \frac{1}{w} \cdot 2^{-m-2} + \frac{1}{2\sqrt{w}} \cdot 2^{-t-1}. \quad (13)$$

From (11), (12) and (13), the total error $\xi_n = P - \sqrt{X}$ is

$$|\xi_n| < \frac{1}{2} (2^{-m-2} + 2^{-t-2})^{n+2}. \quad (14)$$

We show that the convergent speed of the algorithm can be accelerated through another table look-up. The iterative formula of step2 in Algorithm LOC is

$S_i = S_{i-1}^2 + S_0$. We can approximate

$$S_2 = S_0 \times (1 + S_0) + 2\hat{S}_0^3, \quad (15)$$

where $\hat{S}_0^3$ is an approximation to S_0^3 read through table look-up using some upper digits of S_0 as an index. $(1 + S_0)$ can be obtained easily by adding 1 to S_0 using only small additional hardware. Now we show the accelerated LOC algorithm, Algorithm ALOC.

[Algorithm ALOC]

$$\text{step1: } S_0 = \frac{1}{4} - L \times X$$

$$\text{step2.1: } S_1 = S_0 \times (1 + S_0) + 2\hat{S}_0^3$$

step2.2: for $i = 2$ to n do

$$S_i = S_{i-1}^2 + S_0$$

$$\text{step3: } P = \frac{K}{2} - K \times S_n$$

Suppose the table for $\hat{S}_0^3$ is of size $2^l \times l$ bits ($4 \leq l \leq m$). Then, ξ_1 by Algorithm ALOC becomes

$$\begin{aligned}
|\xi_1| &= \left| \frac{K}{2} - K \cdot (S_0 + S_0^2 + 2\hat{S}_0^3) - \sqrt{X} \right| \\
&< 2^{-3s-l+3}
\end{aligned} \tag{16}$$

This value is 2^{-l+3} times smaller than $|\xi_1|_{\max}$ in Algorithm LOC. It means that ALOC produces $(l-3)$ more correct bits than LOC using $l \times 2^l$ bits table.

2.3.3 Comparisons

In this subsection, we compare our converging algorithms (LOC and ALOC) to the multiplicative Newton-Raphson method (NR in short) with the direct table-lookup with equal numbers of cycles C on a multiply-add unit.

Table 2.5 shows the result. The precision or the number of correct bits of the result can be evaluated by $\lfloor -\log_2 |G_n \cdot X - \sqrt{X}| \rfloor$ for the NR method and $\lfloor -\log_2 |S_n - \sqrt{X}| \rfloor$ for our method. Here, the table width t for each method is assumed to be large enough not to diminish the accuracy of the final result. The numbers (m, t) for each method are renamed as (m_n, t_n) , (m_l, t_l) , and (m_a, t_a) for NR, LOC, and ALOC, respectively. The result indicates that the proposed algorithm can form square roots faster than the Newton-Raphson method with look-up tables of nearly same size.

When we calculate double-precision (53-bit) square roots for the IEEE standard, $|G_n \cdot X - \sqrt{X}| < 2^{-53}$ or $|S_n - \sqrt{X}| < 2^{-53}$ must be satisfied. Table 2.6 shows how large look-up tables are required with C cycles for each method. From Table 2.6, if we compare these methods at $C = 6$, which means 6 cycles for calculating double-precision

square roots, m_n must be 13 for DA+NR, while $m_l = 7$ for LOC. Required table sizes are 120K bits for NR and 14.4K bits for LOC. This shows that LOC can form double-precision square roots with about 1/8 the table size required for the NR method. This means that the proposed method reduces the leakage power of lookup tables to 1/8. In this case, ALOC is no better than LOC, because the same $m_a = 7$ is required. Table 2.6 also shows that the necessary iterations of the multiplicative Newton-Raphson method is at least 2, otherwise the required table size becomes impractically large. This means the NR method needs at least 6 cycles in forming double precision square roots. By Algorithm ALOC, however, $C = 5$ can be achieved using 29.6K bits table and $C = 4$ using 121K bits table.

From the viewpoint of performance, the proposed linear converging algorithm can create double-precision square roots faster than the Newton-Raphson method. In contrast to the complicated quadratic converging iterative formula involving three multiplications, our linear converging iterative formula is only one multiply-accumulate operation. It contributes to high-speed square root operations.

Since the proposed square rooting algorithm uses a high-speed multiplier, the low-power multiplier proposed in the previous section 2.2 can also be applied and the both power reduction effects can be achieved.

Table 2.5 Number of correct bits by each square root algorithm

Method	C=2	C=3	C=4	C=5	C=6	C=7	Table size
NR	–	$2m_n+1$	–	–	$4m_n+2$	–	$2^{m_n} \times t_n$
LOC	$2m_l+4$	$3m_l+6$	$4m_l+8$	$5m_l+10$	$6m_l+12$	$7m_l+14$	$2^{m_l} \times (t_l + 2b)$
ALOC	–	$3m_a+1+3$	$4m_a+1+5$	$5m_a+1+7$	$6m_a+1+9$	$7m_a+1+11$	$2^{m_a} \times (t_a + 2b) + 2^1 \times 1$

Table 2.6 Table size for double-precision square roots with C cycles

Method	C=3	C=4	C=5	C=6
NR	1728Mbit ($m_n=26, t_n=27$)	–	–	120Kbit ($m_n=13, t_n=15$)
LOC	7.9Mbit ($m_l=16, t_l=17$)	472Kbit ($m_l=12, t_l=12$)	58Kbit ($m_l=9, t_l=10$)	14.4Kbit ($m_l=7, t_l=9$)
ALOC	0.99Mbit ($m_a=13, t_a=15$)	121Kbit ($m_a=10, t_a=13$)	29.6Kbit ($m_a=8, t_a=11$)	N.A. (same as LOC)

2.4 Efficient Initial Approximation for Division

We discuss multiplicative division on a multiply-add unit in this section. We concentrate on forming the reciprocal of a given divisor $Y = [1.y_1y_2 \cdots y_my_{m+1} \cdots y_b]$, where $1 < Y < 2$. We first review the conventional initial approximation methods and a higher order converging algorithm [7] in subsection 2.4.1. Then we propose an efficient initial approximation method in subsection 2.4.2. We also propose an accelerated higher order converging algorithm by developing a speed-up technique on convergence in subsection 2.4.3. In subsection 2.4.4, we compare our methods to the conventional methods.

2.4.1 Conventional Multiplicative Division

Initial Approximation

Multiplicative division methods begin with an initial approximation R_0 to the reciprocal of the divisor Y . After R_0 is formed, an iterative formula of a converging algorithm is used to achieve the desired precision. The simplest way of forming R_0 is reading an approximation to $1/Y$ directly out of a look-up table [16][17]. Suppose the table width be t bits. As the table is addressed with the m most significant bits of Y , an initial approximation R_0 is in the range $0.5 < R_0 < 1$ and has the form $[0.1r_1r_2r_3 \cdots r_t]$. To give the best value for the subinterval $[p, p + 2^{-m})$, where $p = [1.y_1y_2 \cdots y_m]$, the

look-up table should contain the value $\frac{1}{2}(\frac{1}{p} + \frac{1}{p+2^{-m}})$. The worst error occurs on the first subinterval $p = 1$. The total error ε_d considering the truncations error due to storing only t bits in the table is

$$|\varepsilon_d| = \left| R_0 - \frac{1}{Y} \right| < 2^{-m-1} + 2^{-t-2} \quad (17)$$

and the table is of size $2^m \times t$ bits.

We can select any combination of m and t . When we set $t = m$, ε_d satisfies $|\varepsilon_d| < \frac{3}{2} \cdot 2^{-m-1}$. When $t = m+3$, $|\varepsilon_d| < \frac{17}{16} \cdot 2^{-m-1}$. m and t should be selected so that the required table size becomes smallest as the approximation satisfies the necessary accuracy. This is common to the approximation methods in this thesis.

Besides this direct approximation, a linear approximation method can be employed. In this case, a linear function $-C_1 \times Y + C_0$ is adopted to approximate $1/Y$ and the calculation of it requires one cycle on a multiply-add unit. The two coefficients C_0 and C_1 are read out of a look-up table addressed with the m most significant bits of Y and have the length of t bits for each.

Let $E(Y)$ be the error function:

$$E(Y) = -C_1 \times Y + C_0 - \frac{1}{Y}. \quad (18)$$

Differentiating (18) yields $E'(Y) = \frac{1}{Y^2} - C_1$. Hence,

$$E(Y)_{\max} = E\left(\frac{1}{\sqrt{C_1}}\right) = C_0 - 2\sqrt{C_1} . \quad (19)$$

To minimize $|E(Y)_{\max}|$, the errors of both endpoints of each subinterval, i.e., $E(p)$ and $E(p + 2^{-m})$, should have the same value and it should be equal to $-E(Y)_{\max}$. From these conditions we get

$$C_1 = \frac{1}{p \cdot (p + 2^{-m})}, C_0 = \frac{p + 2^{-m-1} + \sqrt{p \cdot (p + 2^{-m})}}{p \cdot (p + 2^{-m})}, \text{ and} \quad (20)$$

$$E(Y)_{\max} < \frac{1}{p^3} \cdot 2^{-2m-3} \quad (21)$$

The total error of the linear approximation ε_l considering truncation errors due to storing only t bits of the two coefficients in the table is

$$|\varepsilon_l| < 2^{-2m-3} + 2^{-t} \quad (22)$$

and the table is of size $2^m \times t \times 2$ bits.

For example, setting $t = 2m+3$ yields $|\varepsilon_l| < 2^{-2m-2}$ and the table is of size $2^m \times (4m+6)$ bits.

A polynomial approximation using a polynomial of degree $n \geq 2$ may be used [18][19]. It needs n multiply-accumulate operations and the look-up table must keep $(n + 1)$ coefficients of the polynomial. It is not efficient as an initial approximation method, because the error decreases faster by adopting higher order converging algorithms.

Multiplicative Division Algorithm

The most commonly used multiplicative division algorithm is the Newton-Raphson method shown below using R_0 as an initial approximation to $1/Y$.

$$R_i = R_{i-1} \times (2 - R_{i-1} \times Y), \quad (23)$$

where R_i converges to $1/Y$ quadratically. The Newton-Raphson method (23) has been extended in the following way [7]:

$$\text{step1: } D_i = (1 - R_{i-1} \times Y),$$

$$\text{step2: } R_i = (1 + D_i + D_i^2 + D_i^3 + \dots D_i^{k-1}) \times R_{i-1}$$

Note that this algorithm has a property of k_i -th order convergence. We rewrite this higher order converging algorithm for implementation on a multiply-add and we call it Algorithm HOC. (n is the number of iterations.)

[Algorithm HOC]

for $i = 1$ to n do

$$\text{step1: } D_{i,0} = (1 - R_{i-1} \times Y)$$

step2.1: for $j = 1$ to $k_i - 2$ do

$$D_{i,j} = D_{i,0} \times D_{i,j-1} + D_{i,0}$$

$$\text{step2.2: } R_i = R_{i-1} \times D_{i,k_i-2} + R_{i-1}$$

To analyze the error of Algorithm HOC, let $\varepsilon_{i,j}$ be the error of R_i when step2.1 is iterated j times;

$$\varepsilon_{i,j} = \frac{1}{Y} - (R_{i-1} \times D_{i,j} + R_{i-1}). \quad (24)$$

Then,

$$\varepsilon_{i,0} = Y \cdot \varepsilon_{i-1,k_{i-1}-2}^2 (i \geq 1) \text{ and} \quad (25)$$

$$\varepsilon_{i,j} = D_{i,0} \cdot \varepsilon_{i,j-1} = \varepsilon_{i-1,k_{i-1}-2} \cdot Y \cdot \varepsilon_{i,j-1} (i, j \geq 1). \quad (26)$$

(24) and (25) show that Algorithm HOC has a property of k_i -th order convergence when step2.1 is iterated $(k_i - 2)$ times. The number of required cycles for the i -th iteration is also k_i . If we set $k_i = 2$ for all i 's, the algorithm is the original Newton-Raphson algorithm (23). In the case of $k_i = 4$, it is equivalent to the case where Newton-Raphson iterative formula (23) is iterated twice. In order to achieve the best accuracy for some given number of cycles C , k_i must be 3 for all i 's if C is divisible by 3, otherwise the only exception should be $k_i = 2$ or 4 for only one i . Therefore, we choose such an optimized combination of k_i 's in Algorithm HOC.

2.4.2 New Initial Approximation Method

In this subsection, we propose a new initial approximation method. We adopt a modified linear function

$$A_1 \times (2p + 2^{-m} - Y) + A_0$$

instead of $-C_1 \times Y + C_0$ for the approximation to $1/Y$. Recall that $p = [1, y_1 y_2 \cdots y_m]$.

We find the following relation between C_1 and C_0 in the linear approximation (20).

Then $A_1 \times (2p + 2^{-m} - Y)$ can form almost the same value as $-C_1 \times Y + C_0$ by setting

$A_1 = C_1$. We can decide A_0 to improve the approximation.

We get the value $(2p + 2^{-m} - Y)$ as follows. Let

$$q = Y - p.$$

Then q has the form

$$q = [0.00 \cdots 0.y_{m+1}y_{m+2}y_{m+3} \cdots]$$

We get the relation below:

$$(2p + 2^{-m} - Y) = p + 2^{-m} - q = [1.y_1y_2y_3 \cdots y_m\tilde{y}_{m+1}\tilde{y}_{m+2}\tilde{y}_{m+3} \cdots], \quad (27)$$

Where $\tilde{1} = 0$ and $\tilde{0} = 1$. Thus $(2p + 2^{-m} - Y)$ can be obtained only by inverting less significant bits than y_m in Y .

Now we show the appropriate value of A_0 . A_0 should be an approximation to

$$\frac{1}{Y} - A_1 \times (2p + 2^{-m} - Y), \quad \text{where } A_1 = C_1 = \frac{1}{p \cdot (p + 2^{-m})}. \quad \text{We use the following}$$

transformation:

$$\begin{aligned} A_0 &= \frac{1}{Y} - A_1 \times (2p + 2^{-m} - Y) \\ &= \frac{1}{p + q} - \frac{(p + 2^{-m}) + (p - Y)}{p \cdot (p + 2^{-m})} \\ &= \frac{1}{p} \cdot \left(1 + \frac{q}{p}\right)^{-1} - \frac{(p + 2^{-m}) - q}{p \cdot (p + 2^{-m})} \\ &= \left(\frac{1}{p} - \frac{q}{p^2} + \frac{q^2}{p^3} - \cdots\right) - \left(\frac{1}{p} - \frac{q}{p \cdot (p + 2^{-m})}\right) \\ &= \frac{q \cdot (q - 2^{-m})}{p^3} + O(2^{-3m}) \end{aligned}$$

Since A_0 depends on both p and q , the table for A_0 should be addressed with the m_p most significant bits of p and the m_q most significant bits of q . Suppose the table width for A_0

and A_1 be t_0 and t_1 , respectively. Note that $|A_0| < 2^{-m-2}$. Then, the total error of the modified linear approximation ε_m is shown below:

$$|\varepsilon_m| < (3 \cdot 2^{-m_p-1} + 2 \cdot 2^{-m_q} + 2^{-t_0-1}) \cdot 2^{-2m-2} + 2^{-t_1-1}. \quad (28)$$

In order to make the table size for A_0 nearly equal to that for A_1 , we set

$$m_p = \left\lfloor \frac{m}{2} \right\rfloor, \text{ and } m_q = \left\lfloor \frac{m}{2} \right\rfloor.$$

Adopting $t_0 = \left\lceil \frac{m}{2} \right\rceil + 1$, and $t_1 = \left\lceil \frac{5m}{2} \right\rceil + 4$ result in $|\varepsilon_m| < 2^{-2.5m}$ and the total table size is $2^m \times (3m + 5)$ bits. (Here, we fix the table widths t_0 and t_1 as to m , because they do not diminish the accuracy of the final result.) This result shows that the proposed method is more effective than the linear approximation when $m > 4$. For instance, when $m = 10$, the proposed method gives about three bits better approximation using smaller look-up tables than the linear approximation.

The proposed modified linear approximation method can also be applied in producing an approximation for multiplicative square root [20].

2.4.3 Speed-up Technique on Convergence

By the analysis of subsection 2.4.1, Algorithm HOC gives the best performance when its convergent rate is cubic. In this subsection we propose an accelerated HOC algorithm by developing a scheme to speed up the convergence of Algorithm HOC. Since the iterative formula of step2.1 in Algorithm HOC is $D_{i,j} = D_{i,0} \times D_{i,j-1} + D_{i,0}$. We can

calculate $D_{i,2}(= D_{i,0}^3 + D_{i,0}^2 + D_{i,0})$ in one cycle on a multiply-add unit in the following way:

$$D_{i,2} = D_{i,0} \times (1 + D_{i,0}) + \hat{D}_{i,0}^3, \quad (30)$$

where $\hat{D}_{i,0}^3$ is an approximation to $D_{i,0}^3$ read through table look-up using some upper digits of $D_{i,0}$ as an index. $(1 + D_{i,0})$ can be obtained easily by adding 1 to $D_{i,0}$ using only small additional hardware. Now we show the accelerated HOC algorithm, Algorithm AHOC.

[Algorithm AHOC]

for $i = 1$ to n do

$$\text{stepl: } D_{i,0} = (1 - R_{i-1} \times Y)$$

$$\text{step2.1: } D_{i,1} = D_{i,0} \times (1 + D_{i,0}) + \hat{D}_{i,0}^3$$

step2.2 for $j = 1$ to $(k_i - 2)$ do

$$D_{i,j} = D_{i,0} \times D_{i,j-1} + D_{i,0}$$

$$\text{step2.2: } R_i = R_{i-1} \times D_{i,k_i-2} + R_{i-1}$$

Here, $\hat{D}_{i,0}^3$ is an approximation to $D_{i,0}^3$. Suppose the table for $\hat{D}_{i,0}^3$ is of size $2^l \times l$ ($3 \leq l \leq m$) bits. When $D_{i,0}$ is guaranteed to be $0 < D_{i,0} < 2^{-s}$ the s most significant bits of $D_{i,0}$ are all 0's. We use l bits from the $(s + 1)$ -th to the $(s + l)$ -th position of $D_{i,0}$, as an index for the look-up table. In the case $D_{i,0} < 0$, these bits should be inverted before addressing the table.

Let $\eta_d < 2^{-s-l}$ be the error due to using only l bits of $D_{i,0}$ for addressing the table.

Then,

$$\left| \hat{D}_{i,0}^3 - D_{i,0}^3 \right| = (D_{i,0} + \eta_d)^3 - D_{i,0}^3 < 3D_{i,0}^2 \cdot 2^{-s-l} + O(2^{-3s-2l}) \quad (31)$$

Since the truncation error of $\hat{D}_{i,0}^3$ is less than $2^{-3s-l-1}$, $\varepsilon_{i,1}$ by Algorithm AHOC becomes

$$\left| \varepsilon_{i,1} \right| = \left| \frac{1}{Y} - R_i \times (1 + D_{i,0} + D_{i,0}^2 + \hat{D}_{i,0}^3) \right| < 2^{-3s-l+2}. \quad (32)$$

The worst error of (32) is 2^{-l+2} times smaller than $\left| \varepsilon_{i,1} \right|_{\max}$ by Algorithm HOC. It means AHOC produces $(l - 2)$ more correct bits using a table of size $l \times 2^l$ bits than HOC in forming R_i . In the case of $l \cong s$, AHOC achieves $(k_i + 1)$ -th order convergence with k_i cycles for $(k_i \geq 3)$. Recall that HOC gives k_i -th order convergence with k_i cycles.

2.4.4 Comparisons

In this subsection, we compare the proposed modified linear approximation (ML in short) to the direct approximation (DA in short) and the linear approximation (LA in short), and the proposed Algorithm AHOC to Algorithm HOC on a multiply-add unit. We compare four schemes, HOC with DA (DA+HOC), HOC with LA (LA+HOC), HOC with ML (ML+HOC) and AHOC with ML (ML+AHOC). We show how much precision can be obtained with equal numbers of cycles C . Table 2.7 shows the result. The

precision can be evaluated by $\left\lceil -\log_2 \left| R_n - \frac{1}{Y} \right| \right\rceil$, which shows the number of correct bits

of R_n . Here, table width t for each method is assumed to be large enough not to diminish the accuracy of the final result. The numbers (m, t) for each method are renamed as (m_d, t_d) , (m_f, t_f) , $(m_l, 3m_l + 5)$, and $(m_a, 3m_a + 5)$ for DA+HOC, LA+HOC, ML+HOC, and ML+AHOC, respectively. From Table 2.7, more accuracy can be produced with fewer cycles by the proposed method (ML+HOC or ML+AHOC) than the conventional method (DA+HOC, LA+HOC) with the look-up tables of nearly same size. It shows that the proposed method can reduce the number of cycles and achieves fast division on a multiply-add unit.

When we calculate double-precision (53-bit) quotients for the IEEE standard, R_n must satisfy $\left| R_n - \frac{1}{Y} \right| < 2^{-54}$ because the final multiplication by the dividend X , where $1 \leq X < 2$, is needed. (We do not consider extra bits for rounding.) Table 2.8 shows how large look-up tables are required with C cycles to obtain $1/Y$ by each method. From Table 2.8, in order to achieve the precision of 54 bits with $C = 3$, $m_d = 18, m_f = 13, m_l = 11$ for DA+HOC, LA+HOC, and ML+HOC, respectively. Corresponding required table sizes are 4352K, 448K, and 76K bits, respectively. By the proposed modified linear approximation, we can reduce the required table size into about 1/6 of that by the linear approximation, which is only 1/9 size of that by the direct approximation. In the case of $C = 4$, ML+AHOC requires 3.3K bits table, while ML+HOC requires 7.3K bits. By Algorithm AHOC, we can save more than half of the table size. These results show that the proposed method reduces the table size significantly, which contributes to the leakage power reduction of lookup tables for the same ratio.

Since the proposed division schemes, both initial approximation and converging algorithm, use a high-speed multiplier, the low power multiplier proposed in section 2.2 can also be applied and both power reduction effects can be achieved.

**Table 2.7 Number of correct bits by each initial approximation method
with C cycles**

Method	C=1	C=2	C=3	C=4	C=5	Table size
DA+HOC	m_d	$2m_d+1$	$3m_d+2$	$4m_d+3$	$6m_d+5$	$2^{m_d} \times t_d$
LA+HOC	$2m_f+2$	–	$4m_f+5$	$6m_f+8$	$8m_f+11$	$2^{m_f} \times 2t_f$
ML+HOC	$2.5m_l$	–	$5m_l$	$7.5m_l$	$10m_l$	$2^{m_l} \times (3m_l+5)$
ML+AHOC	–	–	–	$7.5m_a+1-2$	$10m_a+1-2$	$2^{m_a} \times (3m_a+5)+2^l \times l$

Table 2.8 Table size for double-precision reciprocals with C cycles

Method	C=3	C=4	C=5
DA+HOC	4352Kbit ($m_d=18, t_d=17$)	112Kbit ($m_d=13, t_d=14$)	4.0Kbit ($m_d=9, t_d=8$)
LA+HOC	448Kbit ($m_f=13, t_f=28$)	9.5Kbit ($m_f=8, t_f=19$)	1.6Kbit ($m_f=6, t_f=13$)
ML+HOC	76Kbit ($m_l=11$)	7.3Kbit ($m_l=8$)	1.4Kbit ($m_l=6$)
ML+AHOC	–	3.3Kbit ($m_l=7, l=4$)	N.A. (same as ML+HOC)

2.5 Summary of Chapter 2

We have proposed a novel low power multiplication algorithm for reducing switching activity (MASAR for short) through operand decomposition. Functional simulation to estimate the number of logic transitions in C language corresponds well with the dynamic power savings demonstrated by the synthesized netlists. The power savings with MASAR are 13% and 10% respectively for the 64-bit array and tree multipliers. The area overheads for MASAR are 10% for the 16, 32 and 64 bit multipliers. Our algorithm is effective for both array and tree multipliers of 32 bits or larger widths. The proposed algorithm can be practically applied to many digital systems, where minimizing power consumption is important. There is a substantial reduction in the number of logic transitions and a corresponding power savings, with only a small delay and area increase.

For the calculation of square root, we have proposed a new converging algorithm. The iterative formula of our algorithm consists of one multiply-accumulate operation and can be executed in one cycle on a multiply-add unit. When we calculate double-precision square roots in 6 cycles, our method requires only 1/8 the size of look-up tables compared to the multiplicative Newton-Raphson method, and thus achieves the same ratio of leakage power of the lookup tables.

We have proposed multiplicative division on a multiply-add unit. An efficient initial approximation method and fast converging algorithm are important to achieve the desired precision with small number of cycles and with small look-up tables. We have

proposed an efficient initial approximation method with an accelerated higher order converging algorithm for division. When we calculate double precision division at 4 clock cycles, our method requires only $1/6$ the size of the lookup table compared to the linear approximation method, and thus achieves the same ratio of leakage power reduction of the lookup table.

Since the proposed square rooting and division schemes are all based on a high-speed multiplier, the low power multiplier proposed in section 2.2 can also be applied and the power reduction effects can all be achieved at the same time.

CHAPTER 3: Low Power Design of CPU and Cache Memory

3.1 Introduction

From the market trends for battery-operated mobile processors, ever-increasing multi-media applications with heavy data traffic such as videophone, 3D Java games, music-play etc. together with heavy OS handling require a high-performance processor with higher frequency. At the same time, requirements on power dissipation are getting more and more severe. For example, less leakage current for long “wait-for-call” time specification of cellular phone is mandatory.

In order to satisfy both these two requirements, CPU core including cache memory and the first-level local-RAM should be implemented using lower-Vt transistors to achieve high frequency for high peak performance in need. Then leakage current of CPU core should be cut down by controlling supply voltage.

Since it is common to implement large capacity of SRAM for cache memory and local-memory, such SRAM consumes a large portion of the overall power of CPU core. Low leakage SRAM is mandatory and we can make the best use of such SRAM. As for logic parts of CPU core, the only way to cut down leakage power is power supply control in case of idle state.

In this chapter, low-power resume-standby SRAM is introduced in section 3.2. Then, new CPU power saving mode called “Core standby mode” is described in section 3.3. In Core standby mode, CPU logic is powered-off while Cache and internal-SRAM keeps powered to “resume” level at which the data can be retained. Finally section 3.4 summarizes this chapter.

3.2 Resume Standby SRAM

We adopt low leakage SRAM that has two operation modes [21]: One is active mode where read and write requires high-speed access. The other is standby mode, where only data retention is required. Leakage current reduction in active mode is achieved by supply voltage reduction when not-selected. Leakage current reduction in case of standby mode is done by supply voltage reduction to data-retention level.

As shown in Figure 3.1, this SRAM module is divided into 4 banks, and each bank has three controlled power lines: V_{ssm} , V_{ddw} , and V_{ssa} . The module has one more controlled power line V_{ssc} . The power line controllers control voltages of the power lines.

When the memory is not accessed, the leakage current can be reduced by lowering the voltage of PMOS source line V_{ddw} , because of a body-bias effect. Since the PMOSs have large gate width and use low V_t for high-speed, suppressing the leakage current through the PMOS is quite effective. In addition V_{ddw} is connected only to the word drivers to reduce the load. Therefore, V_{ddw} voltage changes fast enough to activate only the selected bank.

The V_{ssm} , the source line of the driver MOS of the memory cells, is controlled to reduce leakage when cells are not being accessed. When the voltage is controlled to be about 0.3V, the memory-cell leakage is cut by 90%, yet the cells can retain their data.

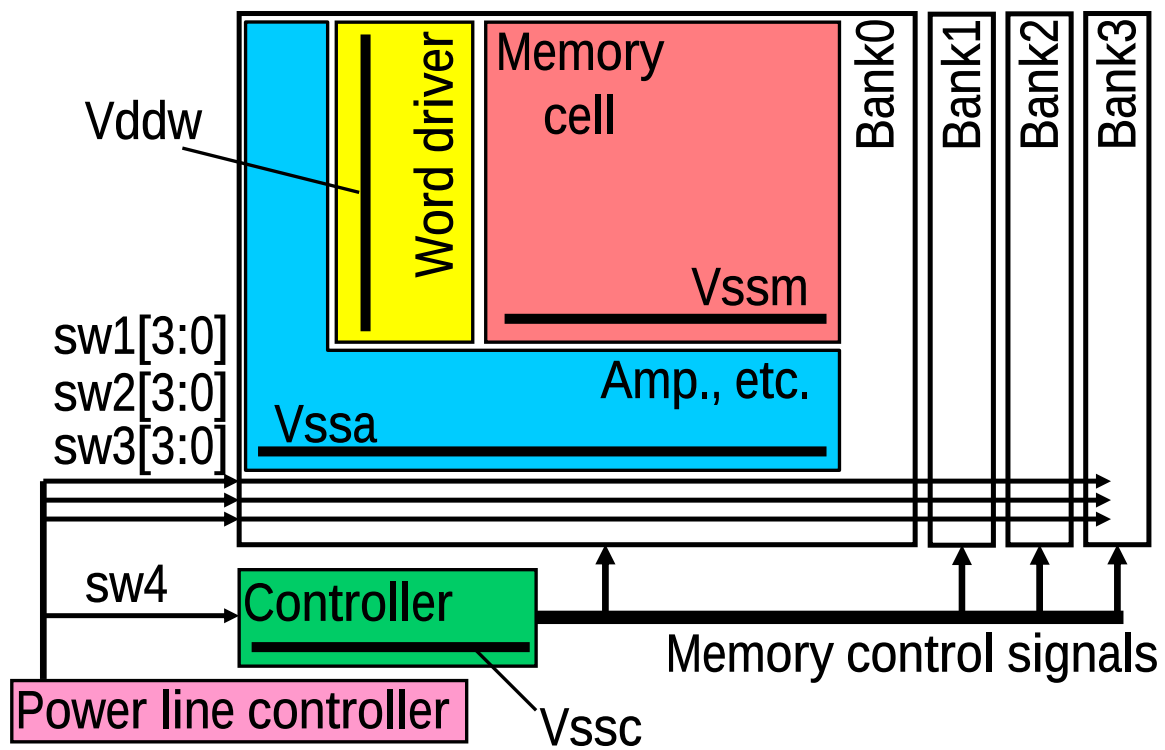


Figure 3.1 Low leakage SRAM implementation

Figure 3.2 is a leakage evaluation result of the low-leakage SRAM. This data is measured by 256KB SRAM at 130nm LP process at room temperature at 1.2V. Compared to the conventional SRAM, leakage power in the active mode is down to 75%, and in the standby mode is down to only 5% level of 50uA.

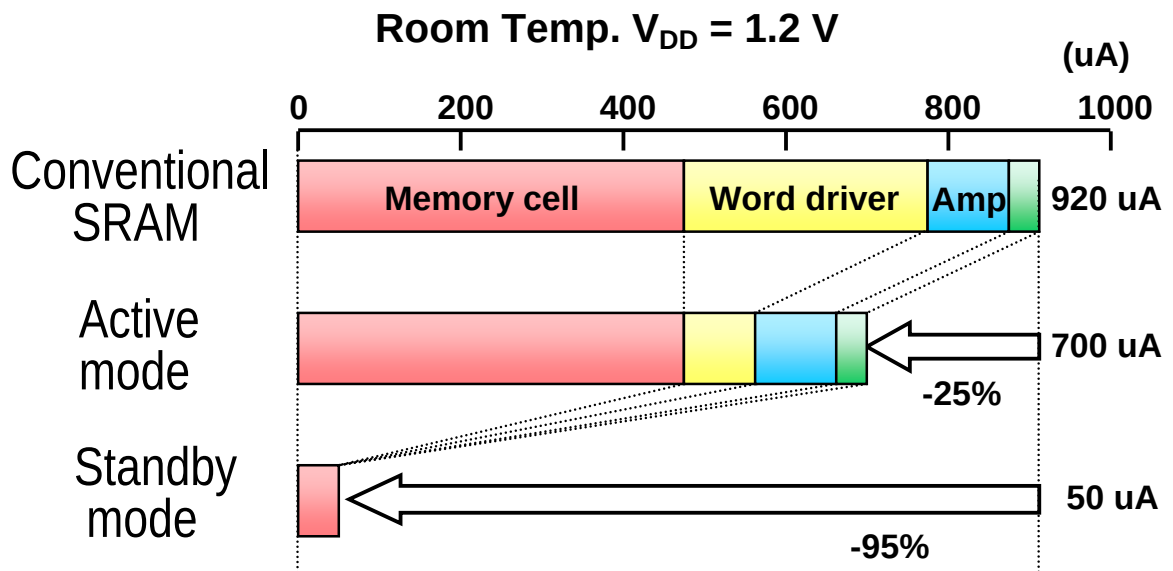


Figure 3.2 Low leakage SRAM evaluation result

3.3 Core Standby for CPU

Core standby CPU [22] is a good CPU component for multi-power domain SoC. In order to achieve high frequency for CPU, we tend to use more low-Vt transistors and consume much leakage current in this CPU power domain. However we cannot power-off the whole CPU as we are going to lose the contents for the cache and RAM. These shortcomings made us to introduce a new power-save mode called core standby.

In the core standby mode separate power domains for CPU logic and CPU RAM areas are created. The CPU enters into core standby mode by the control of software. In this mode, power supply for the logic domain is switched off and the RAM domain goes into a low-leakage state called Resume state, where only data retention is possible. As the data in RAMs and caches are retained during the core standby mode, the CPU can resume the operations quickly on wake-up from this mode compared to power off mode.

Logic area and RAM area of CPU are separated as shown in Figure 3.3. Each RAM block has an RS port, which means, “Resume Standby”. RS signal is asserted during the core standby mode. In order to achieve quick transit and recovery to and from the core standby, we use one cycle access local RAM called ILRAM for “core-standby transit code”, “resume boot code”, and “stack” for CPU resources such as system control registers. Figure 3.4 shows an ILRAM area separation for core-standby support. By these schemes, we minimize the overhead of transit time and recovery time.

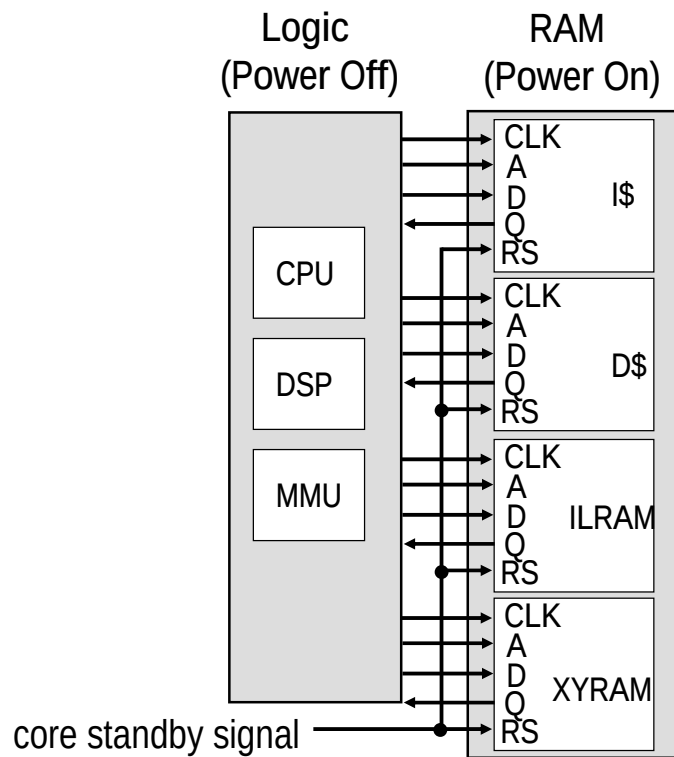


Figure 3.3 Core standby configuration

Resume boot codes to exit from core-standby
Core standby entry codes to enter into core-standby
Stack area for CPU architecture resources

Figure 3.4 ILRAM usage for core-standby

Figure 3.5 shows a flow chart for core standby mode entry. When all CPU tasks finishes, the CPU becomes idle and power management decides to transit to core-standby mode. Then the CPU jumps to “core-standby entry code” that is located on ILRAM. Then the CPU stacks its architecture resources such as system control registers to “stack area” in ILRAM. Then the CPU sets its reset vector to core-standby exit code address for the preparation of awake from core-standby mode. Then core-standby flag is set and SLEEP instruction is executed to transit to core-standby mode.

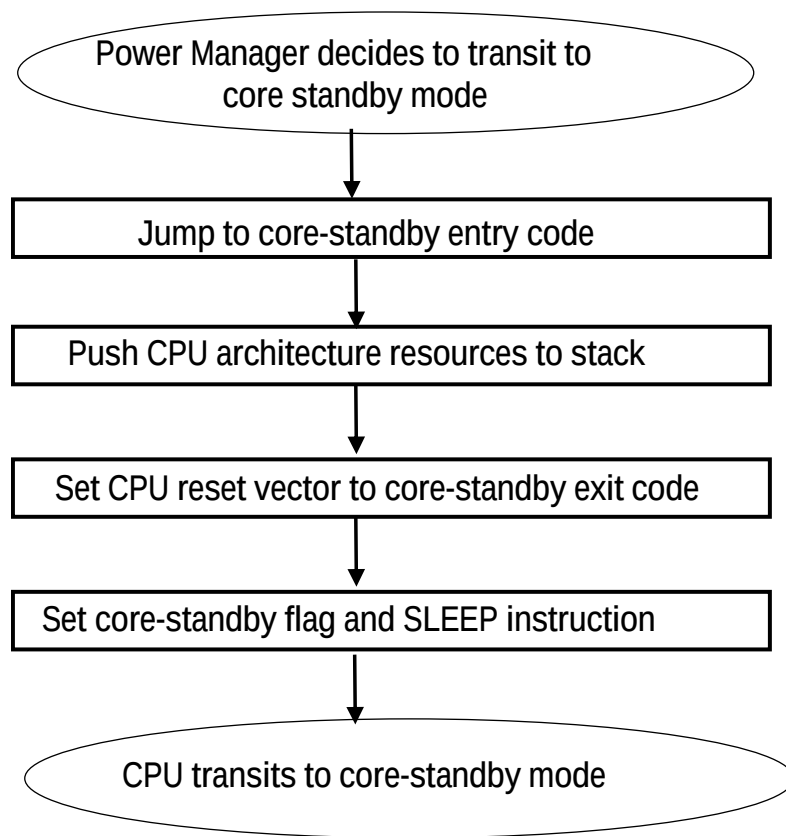


Figure 3.5 Flow chart for core-standby entry

Figure 3.6 shows a flow chart for core standby mode exit. When a system controller outside of CPU receives an interrupt that awakes the CPU, it releases core-standby mode and gives a reset to the CPU. Then the CPU boots and jumps to “resume-boot code” that is located on ILRAM. The CPU executes these instructions and pops architecture resources from the stack. Then the CPU branches to interrupt handler to deal with the corresponding interrupts received.

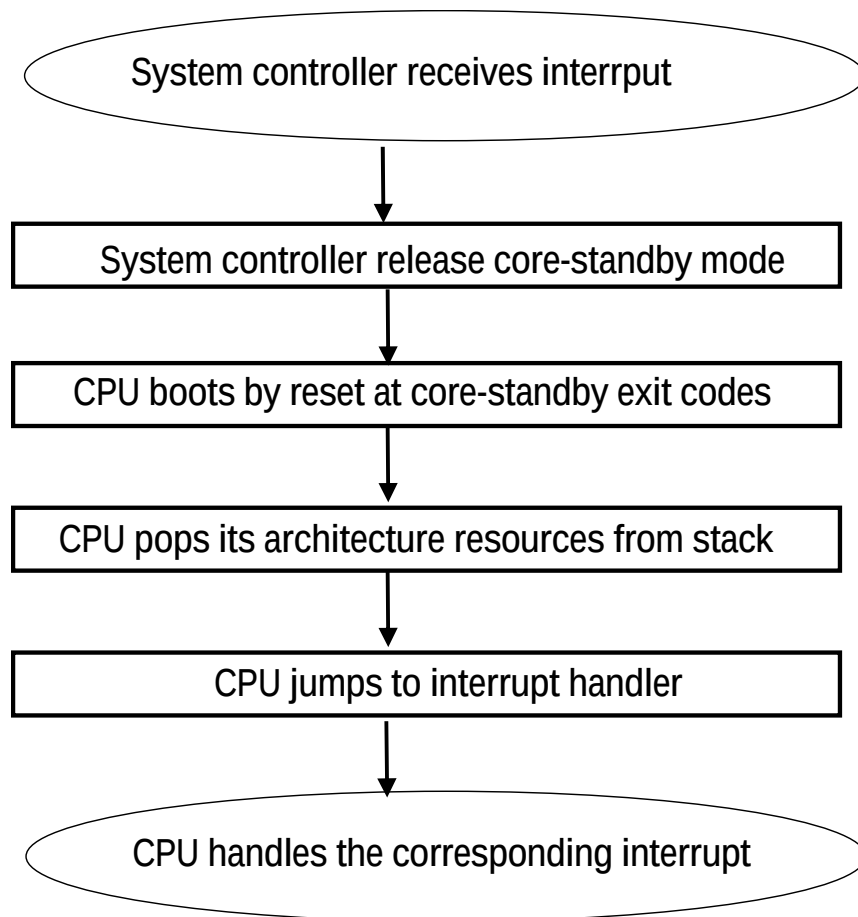


Figure 3.6 Flow chart for core-standby exit

Figure 3.7 shows a resume SRAM implementation. With the assertion of RS, the VSS level of the RAM domain is raised to ARVSS level by voltage adjustment circuit as shown in Figure 3.8 so that the leakage power of the RAM module is minimized. During the “RS” assertion, the RAM modules use RS signal and tie “word line” for memory array to LOW and fix the other input signals from logic domain which might be intermediate voltage level. Thanks to this function, no isolation cells are required between logic and RAM domain. Since this path is very timing critical, we have achieved core standby implementation with very little delay overhead.

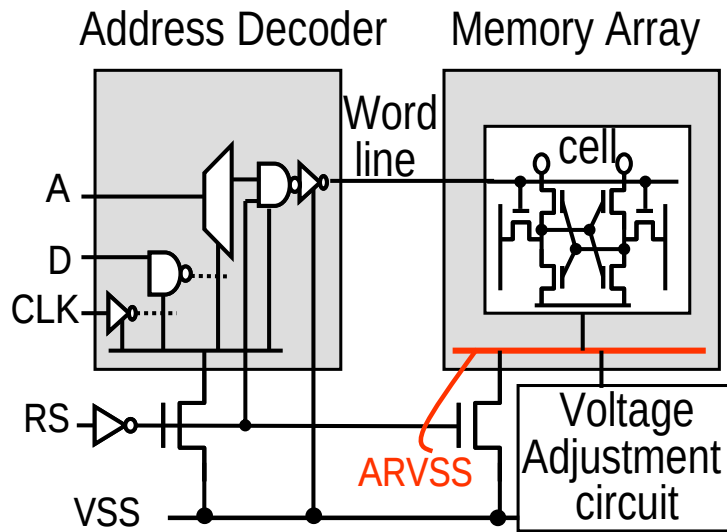


Figure 3.7 Core-standby circuit implementation

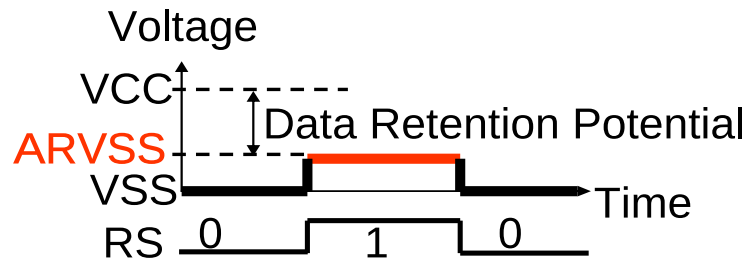


Figure 3.8 Core-standby voltage

Table 3.1 shows evaluation results of core standby mode. The results are compared with the sleep mode and power-off mode. The transition time and recovery time to and from the core standby mode is 5 μ s and 7 μ s including resource push and pop time and power switch transition time. They are about 1 μ s for sleep mode. In case of power-off mode, dirty cache data needs write back before power-off. After reboot, cache and RAM refill is necessary. So overhead for transition and recovery time of power-off mode is usually an order of 100 μ s. The measured leakage power on silicon at room temperature is about 1.0mA in the sleep mode while 0.02mA in the core standby mode. These results show that the core standby mode cuts almost all leakage current with a slight overhead time for resource push and pop. In the case of mobile SoC, as CPUs are placed in idle state for the most of the time in the real usage, the core standby mode contributes greatly to the leakage power reduction.

Table 3.1 Evaluation result of core stanby

CPU power-modes	Normal	Sleep	Core Standby	Power Off
Clock	On	Off	Off	Off
Power supply (Logic)	On	On	Off	Off
Power supply (RAM)	On	Resume	Resume	Off
Transition Time (*1)	-	1.0 μ s	5.0 μ s	> 20.0 μ s(*3)
Recovery Time (*1)	-	1.0 μ s	7.0 μ s	> 40.0 μ s(*4)
Leakage Power (*2)	1.28 mA	1.03 mA	0.02 mA	0.00 mA

(*1) Including resource push/pop time and power switch transition

(*2) Measured at room temperature, 90nmLP

(*3) Cache flush required before power-off

(*4) Cache refill overhead after power-on

3.4 Summary of Chapter 3

We have introduced high-performance and low-power CPU using low-leakage SRAM with resume-standby support and a new CPU power saving mode called core-standby mode. In the core standby mode, CPU logic is powered-off while Cache and internal-SRAM keeps powered to resume level at which the data can be retained.

Evaluation results show that the transition time and recovery time to and from the core standby mode is 5 μ s and 7 μ s including resource push and pop time and power switch transition time. With only this slight overhead for transition and recovery time, we can achieve high CPU core frequency by using lower-V_t transistors. The measured leakage power on silicon at room temperature is about 1.0mA in the sleep mode while 0.02mA in the core standby mode. These results show that the core standby mode cuts almost all leakage current.

It is highly dependent on applications that how much percentage of time CPU is in idle state, but one main target of application processor of mobile chip show that CPU is placed in idle state for the most of the time in the real usage of a cell phone. In the future, the trend of higher frequency in peak performance and lower leakage in idle state is essential, and the core standby mode contributes greatly to the leakage power reduction in almost all the fields.

CHAPTER 4: SoC Design for Dynamic Power Reduction

4.1 Introduction

As the process technology progresses, complicated systems can now be achieved on one-chip SoC. For such an SoC, dynamic power consumption control is an important issue because if none of low power techniques are applied, the SoC consumes much dynamic power even when SoC does not handle any program, i.e. CPU is in idle state and all the IPs are stopped. Recent SoCs apply many levels of clock-gating techniques.

Figure 4.1 shows typical clock-gating techniques. For functional IPs, static clock-gating by setting clock control registers can be effective, because software knows when to use such IPs and it is easy to release clock stop register to supply clock before using the IP and set clock stop register after finishing the function. This static clock-gating can also be applied for some units of CPU core especially for coprocessors. Another commonly-used clock-gating technique is dynamic clock control by the hardware. This is normally applied by logic synthesis tool by the analysis of RTL descriptions.

However for ever-increasing complications of SoCs, these clock-gating techniques are not sufficient. One reason is that logic synthesis tool cannot apply clock-gating by understanding functional meaning. This should be done by human designer at functional design level and by clearly describing at RTL when to stop clock delivery for the group of logic in C-driver level shown in Figure 4.1. We will explain such techniques in section 4.2.

For the further sophisticated dynamic power reduction of clock-gating, partial clock activation method is effective where clock system including PLLs is separated and

only activated when necessary. The clock activation is linked with the corresponding power-domain activation. For this purpose, the clock system should be reorganized and is more related to application scene such as music playback. We will explain this technique that is applied to mobile SoC in section 4.3.

Finally section 4.4 summarizes this chapter.

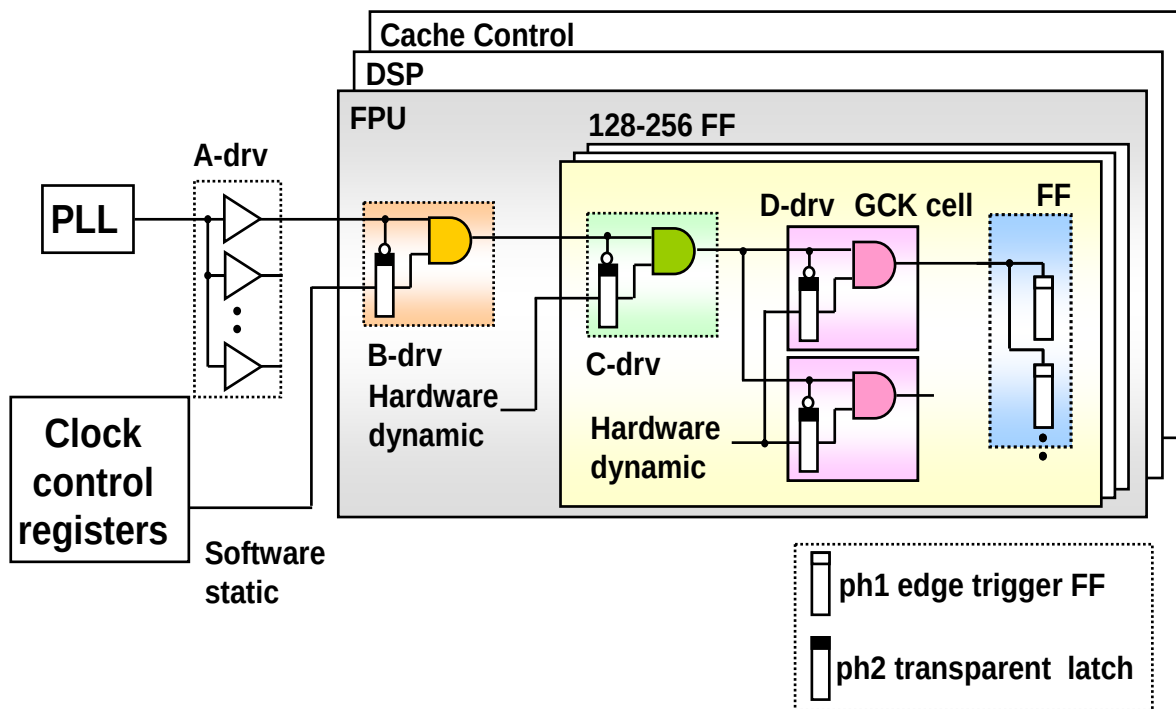


Figure 4.1 Conventional clock-gating techniques

4.2 Dynamic Clock Control

4.2.1 Background

In order to eliminate waste dynamic power, it is very important to stop clock supply for the parts except CPU that does not need to work. From the power analysis of SH-mobileG1 [24], more than 60mA was consumed when all the CPUs and functional modules were stopped and clock delivery for these modules were stopped. We found that about 40mA was consumed by bus related modules. They are bus-routers, bus-bridges, and bus controllers. The problem here is that the conventional software module-stop scheme is not sufficient for dynamic power saving. As shown in Figure 4.2, the clock delivery for unused module is stopped by programming module standby register conventionally by the software control.

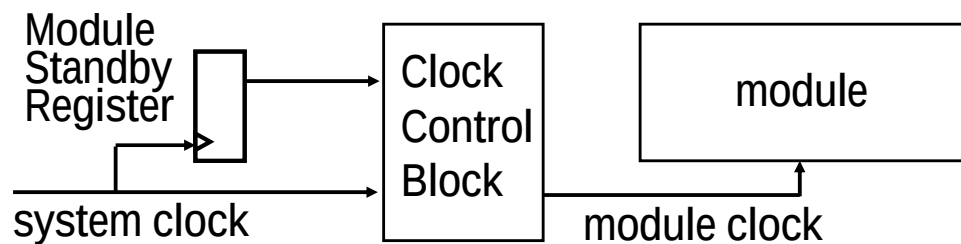


Figure 4.2 Conventional software module-stop

4.2.2 Dynamic Clock Stop for Functional Modules

As for functional modules, conventional software module-stop scheme shown in Figure 4.2 is usually effective, as the software knows when to start and stop the module. However, for a very large-IP like video processing unit or VPU for short, this control is not sufficient. We have introduced dynamic module-stop scheme at sub-block level for VPU. Figure 4.3 shows a VPU encoding process flow. 10 sub-blocks of the VPU execute each processing in parallel at 7-stage-pipeline. The 6-th time slot in Figure 4.3, for example, two VPU sub-blocks namely “Rate-Control” and “Deblocking Filter” process the n -th repetition of the VPU encoding. At the same time, “DMA Write” sub-block processes the previous $(n-1)$ -st repetition, and “Entropy Encoding” and “Motion Compensation” sub-blocks process the next $(n+1)$ -st repetition. One time slot for VPU encoding pipeline is fixed to 1280 clock cycles, and each sub-block starts processing simultaneously and ends one by one and waits for the next time slot to start.

This control feature enables to control clock delivery at sub-block level. Since each sub-block of the VPU finishes processing one by one and waits for the next time slot, we introduce watch and control block that checks each sub-block’s pipeline status and stops clock delivery just after its processing is completed. This mechanism is shown in Figure 4.4. This architecture enables to control clock delivery for each sub-block at each time slot and eliminates waste dynamic power dissipation.

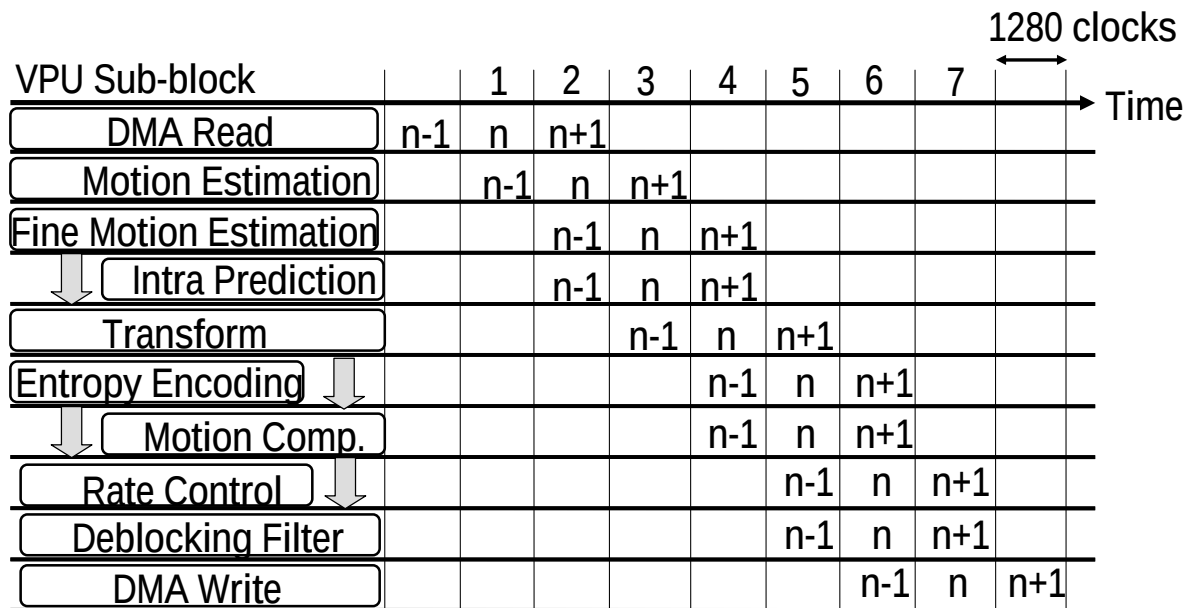


Figure 4.3 VPU encoding process flow

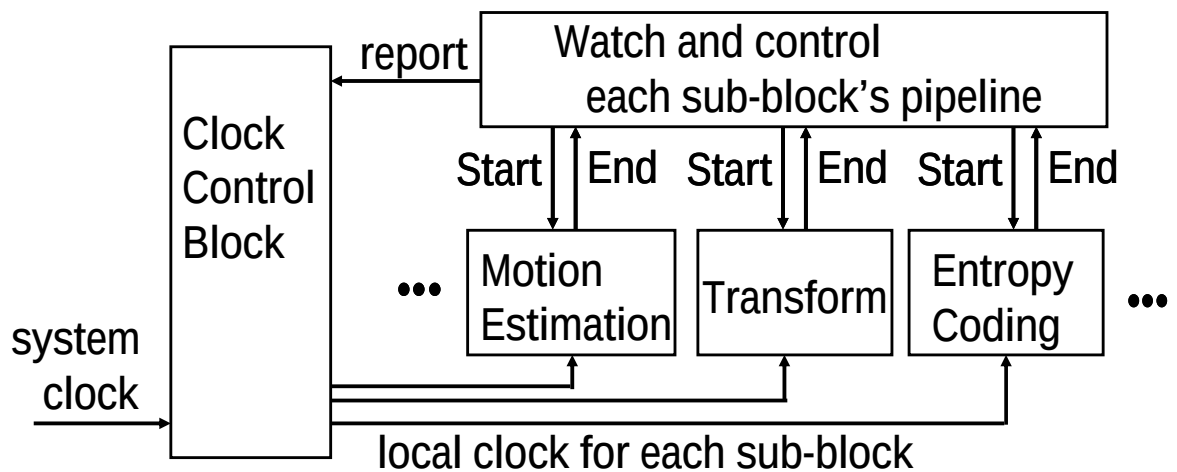


Figure 4.4 Clock control mechanism for VPU

4.2.3 Dynamic Clock Stop for Bus Modules

We have applied dynamic module-stop technique to bus-related modules as well, as they consumed much waste power. Bus-related modules cannot adopt the conventional software module-stop scheme shown in Figure 4.2 unless all the modules related to the bus is stopped. Figure 4.5 explains dynamic module-stop technique using Local Media Bridge or LMB for short as an example. We have designed small clock control circuits for trigger detection logic and a down counter. By using this method, LMB module-level clock delivery is stopped when there is no bus transaction on the bus. When a new bus request arrives and is detected by the detection logic, clock supply restarts by the hardware automatically. A programmable down counter specifies when to stop clock delivery after one bus transaction is completed. The timing chart of Figure 4.5 shows an example of the case where the down counter is set to 4. Two additional cycles are required for a bus request during clock-stop condition. No additional latency is inserted for continuous bus transactions by programming the counter with an appropriate value.

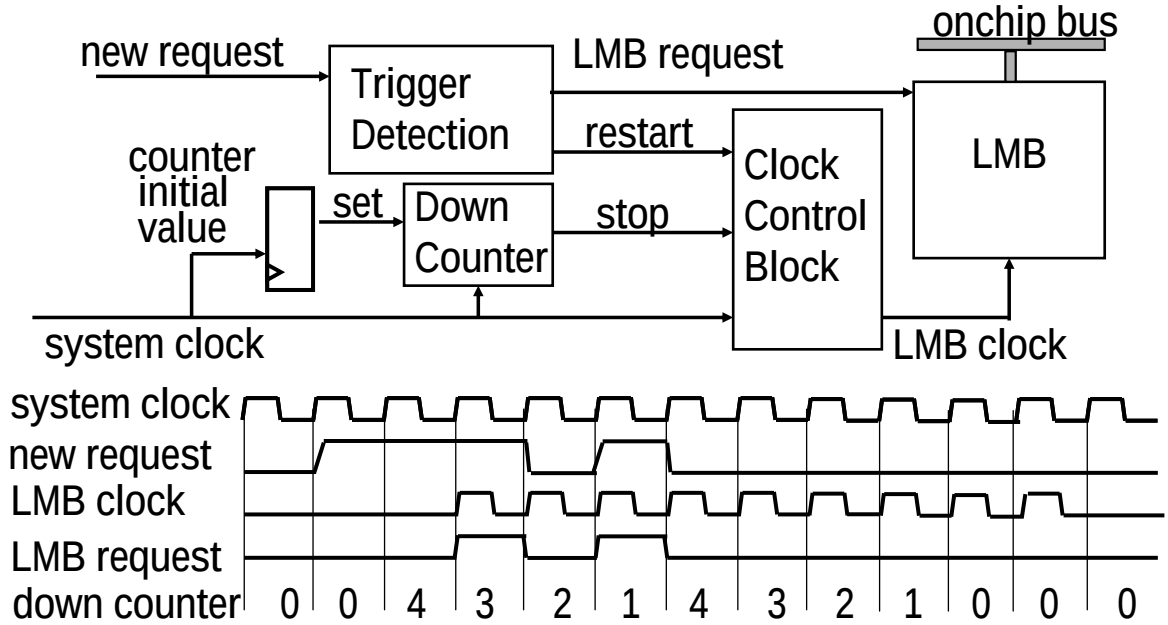


Figure 4.5 Clock control mechanism for LMB

4.2.4 Evaluation Results

Figure 4.6 shows evaluation results of dynamic module-stop or DMS for short. We have applied DMS to VPU and about 10 bus-related modules. Two power consumption data are shown. The first scenario is video telephony. The left graph shows the data when Dynamic module stop is off and the right one with DMS on. The total power consumption is 149mA and 113mA, respectively. The dynamic module-stop scheme reduces more than 35mA. The second scenario is motion picture playback. The total power is 74mA after DMS is on and the same effects of DMS are reconfirmed. We have shown that both functional modules and bus-related modules can reduce dynamic

power by DMS technique through additional clock control logic. DMS techniques can be applicable to almost all modules wherever clock is supplied. Key point here is that the power saved by DMS should be larger than the additional power consumed by additional logic for DMS.

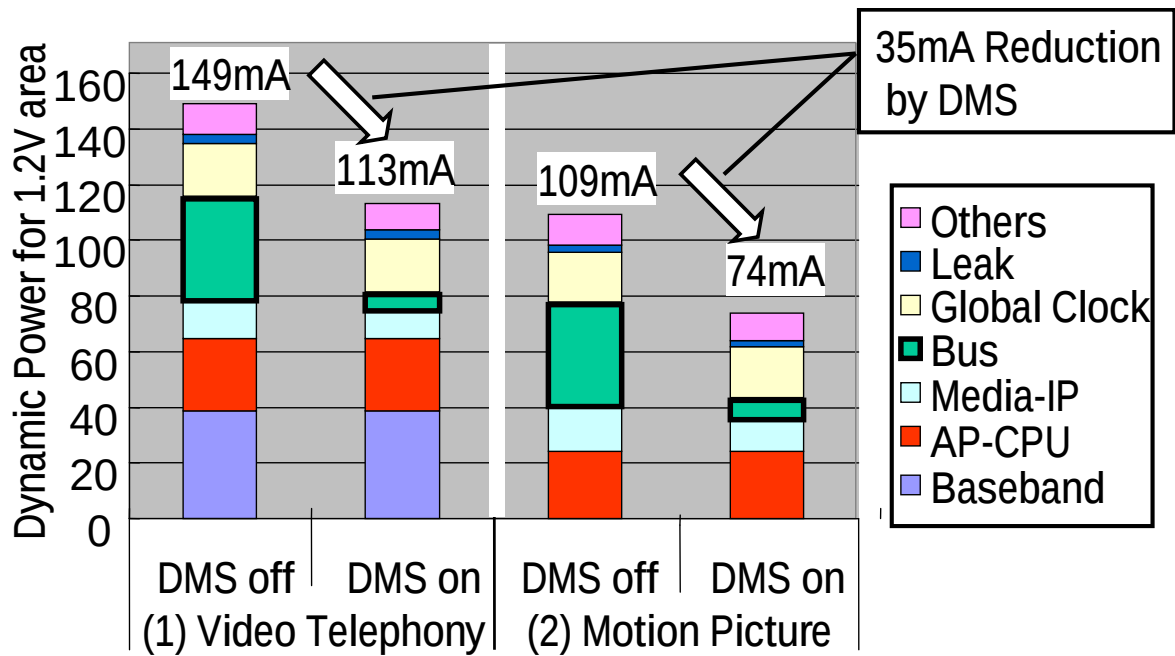


Figure 4.6 Evaluation results for dynamic module-stop

4.3 Partial Clock Activation for Mobile SoC

4.3.1 Background

Cellular phones are used not only for voice communications, but also used for video telephony, 3D Java games and high-class business applications. High-level operating systems, such as Symbian or Linux, are widely adopted in the cellular phone system. A high performance application processor is required to handle such high-level OS together with ever-increasing applications with heavy data traffic. For heavy multimedia handling, media accelerator is suitable along with an application processor. A baseband processor is also required to handle modem sub-system, both the high-performance WCDMA service and the GSM/GPRS service which is generally available worldwide. One-chip integration of a baseband processor and an application processor together with a media accelerator is superior in performance, power consumption, and cost, such processor is widely adopted especially for high performance cellular phone system [22][24]. The dynamic module-stop scheme described in the previous section 4.2 was adopted for onchip-bus router and bus-bridge modules; a clock-driver supplying clock for a module-level is gated by hardware logic during no bus transaction. When a new bus request is detected by the logic, the clock-driver starts clock supply automatically. However, further requirements of power reduction especially for such dedicated applications as music playback are strongly required for mobile SoC.

4.3.2 Dynamic Power Reduction by Partial Clock Activation

We have analyzed our module configuration related to music playback scene. In this scene, CPU runs only 10% of the time for handling files, and it transits to core standby mode for the rest of 90% time. DSP works for music decoding continuously. Clock generation module in always-active domain is constantly working and clocks of most of functional modules in this domain are gated inside modules. We have evaluated power consumption and found that the always-active domain consumed 17.4mA of 28.0mA of total power. Always-running clock dividers and a PLL consume a large amount of power. Clock drivers also consume some portion of power.

In order to reduce the waste power consumption of clock related paths, we have introduced partial clock activation scheme shown in Figure 4.7. Clock trees are separated and only the selected clock tree necessary for the scene is activated. We also separate PLLs into PLL1 and PLL2 so that the PLL1 can stop for the most of the time during music playback scene. Since DSP accesses to BUS for external memory accesses, enhanced dynamic module-stop scheme is applied together with circuit separation including PLLs. This contributes to shut off the clock divider and clock drivers circuit when unnecessary and stops clock supply to some modules inside common area.

Figure 4.8 shows how clocks inside always-active area behave by showing waveforms. In accordance with partial clock activation transition, PLL1 can stop most of the time during music playback scene, and the clock paths delivered to CPU or BUS can be halted when unnecessary.

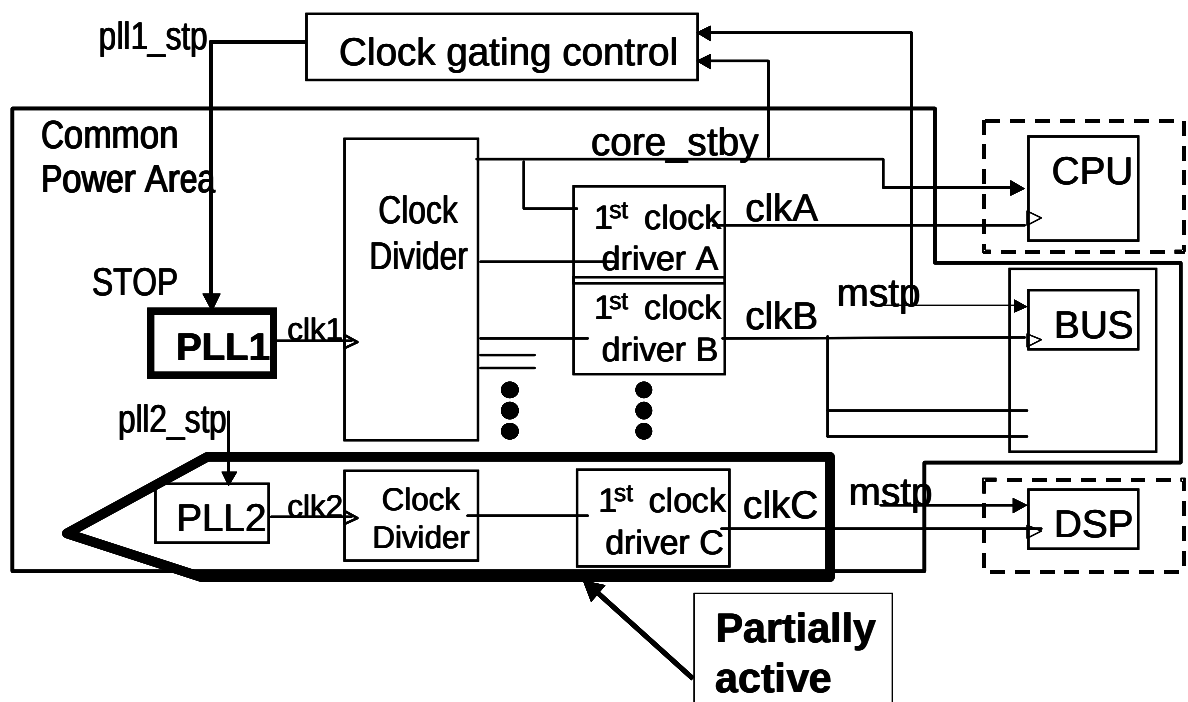


Figure 4.7 Partial clock activation method

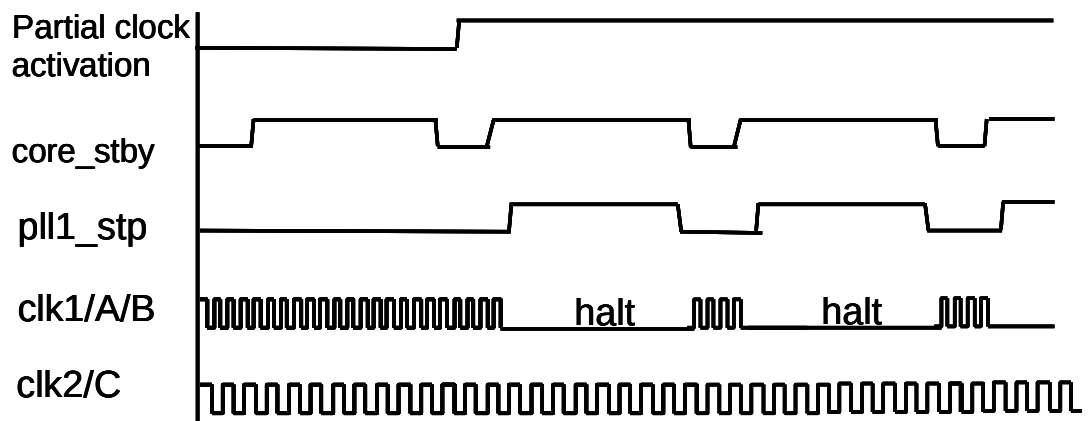


Figure 4.8 Timing chart of partial clock activation

4.3.3 Evaluation Results

Figure 4.9 shows power consumption of music playback scene with and without partial clock activation technique. Power consumption of always-active common area is reduced significantly from 16.7mA to 5.2mA, which means 70% of the power of the common area has been reduced. The total power consumption for music playback scene is now reduced from 28.0mA down to 16.3mA, which means 40% of the total power has been reduced.

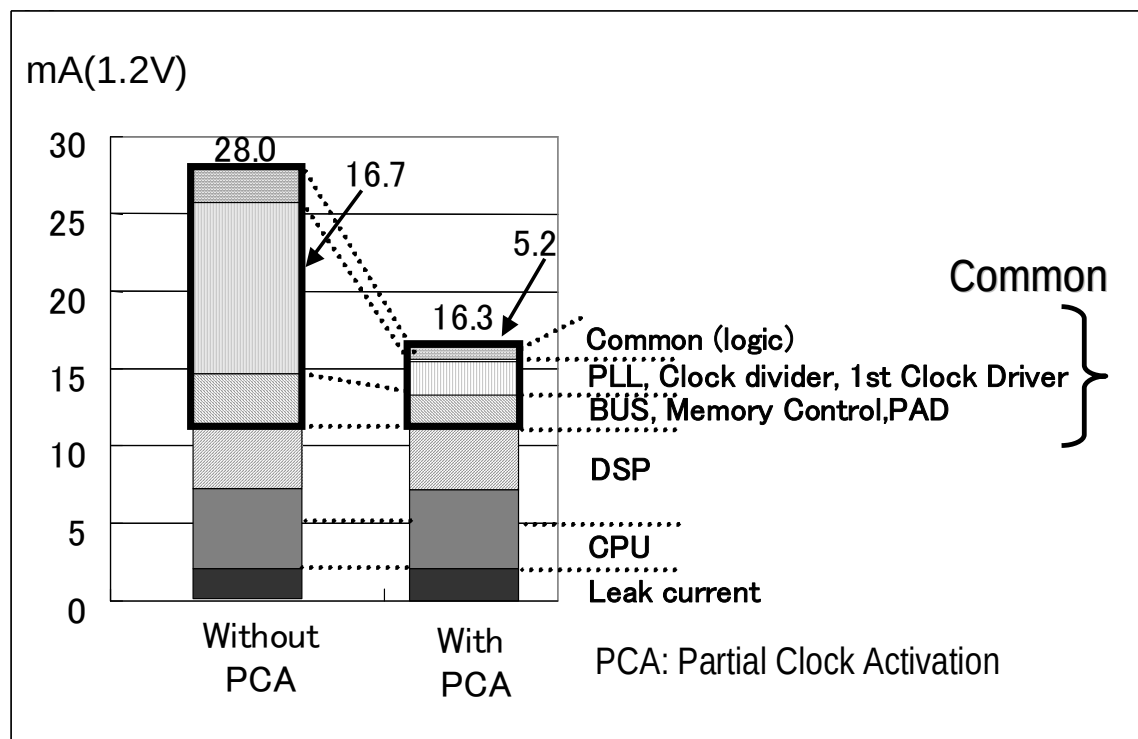


Figure 4.9 Evaluation result of for power consumption in music playback scene

4.4 Summary of Chapter 4

In this chapter clock-gating techniques for dynamic power consumption of SoCs have been discussed. We show that the conventional clock-gating techniques such as static clock-gating by setting clock control registers and dynamic clock control by the hardware applied by logic synthesis tool are not sufficient for the current large multi-functional SoC.

We have presented two clock-gating techniques that can be designed at functional design level. One is dynamic clock control for large IP with dynamic module-stop scheme at sub-block level. The other is dynamic clock control for bus-related modules with dynamic module-stop by new bus transaction request detection. We have applied these techniques to video-processing unit and bus-related modules. Our evaluation results show that the dynamic module-stop techniques reduces current more than 35mA out of 149mA and 109mA for video telephony and picture playback applications, respectively. Dynamic module-stop techniques can be applied to almost all modules that work intermittently and have a certain amount of idle state where clock supply is not needed.

We have applied a more sophisticated dynamic power reduction of clock-gating at more system control level called partial clock activation method, where clock systems including PLLs are separated and only activated when necessary. Active power in always-active area is further reduced to 33%. 40% of LSI power reduction has been achieved for music playback.

CHAPTER 5: SoC Design for Static Power Reduction

5.1 Introduction

As the process technology progresses, complicated systems can now be achieved on one-chip SoC and at the same time leakage power has become dominant. For such an SoC, leakage power consumption control is an important issue because if none of low power techniques are applied, SoC consumes much leakage power as long as voltage is supplied.

Leakage power amount and ratio increases in sub-100nm generation SoC. One good solution is obviously power domain separation to shutdown unused circuits. Required components and techniques for multi power domains are isolation cells called micro-IOs for signals across power domains, backup latches to retain values for fast restart, and power switches and their controller. We will explain such components for multi-power domain techniques [25] in section 5.2.

In section 5.3, we describe our multi-power domain SoC. The conventional partitioning has a few power domains, with each domain powered on and off depending on the operation mode and thus has a star structure. Our single chip SoC contains plural sub-systems and power domain separation amounts to twenty and has a tree structure. Power for unused domain should be off even for a short period to reduce leakage power. We have applied hierarchical power domain management for overhead reduction such as complicated control logic and micro-IO cells.

In section 5.4, we describe our homogeneous multi-core SoC. Both high-performance and low power consumption are required for such fields as digital TV or car

information system. Multi-core SoC is a good solution for this purpose, same as in the way for high performance PC or server. CPU cores can work in parallel and at the same time, power dissipation are getting more and more severe especially for the embedding processors. In order to achieve leakage power saving for multi-core SoC, each CPU core and each internal SRAM are assigned to split power supply domains so that the power supply of unused CPU cores can be cut off to reduce the leakage power. For the power-off, RAM resume mode can be selected for the fast restart on wake up.

Finally section 5.5 summarizes this chapter.

5.2 Components for Multi-power Domains

Figure 5.1 shows an example of two power domains and the constituent components. The left power domain includes processor and SRAM such as cache memory or no-wait cycle access SRAM. This domain will be powered-off and the right power domain will be kept powered. Micro-IOs are isolation cells that are inserted between power domains for the prevention of invalid signal transmission from the left power-off domain. Backup latches keep the register values of the left power domain and restore their values on wake-up by the hardware. Power switches shown by PSW consist of thick-gate oxide high- V_t NMOS and are controlled by power switch controller, PSWC. PSWC is in turn is controlled by power state controller shown as PSC.

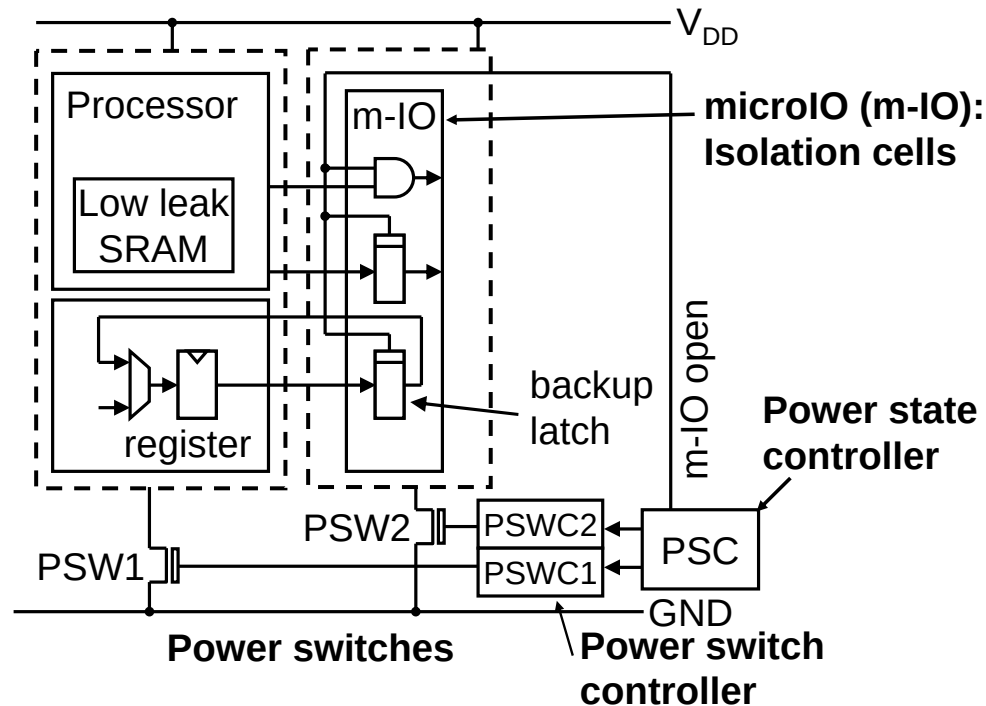


Figure 5.1 Components of multi-power domain SoC

Micro-IOs are essential components for multi power domain separation. As shown in Figure 5.2, signals outputted from power-off domain may become intermediate level. In order to avoid such signal transmission, micro-IOs should be inserted for every signal from power-off domain to power-on domain. The right part of Figure 5.2 shows the connection between two blocks with the micro-IO. Each block has a power switches, and a power switch controller (PSWC). The micro-IO prevents invalid signal transmission by using 4-input AND function, in which three input signals (cds, cdr, and e) are used as control signals. The cds and cdr are driven to low, respectively; when the sender and the receiver block is turned off internally by the power switch. The block designers have to drive “e” to low when the sender block (vdd1) is turned off from outside of the chip. When the two power domains have the same vdd and vss, micro-IOs can be implemented as a normal AND or OR gate, or a latch.

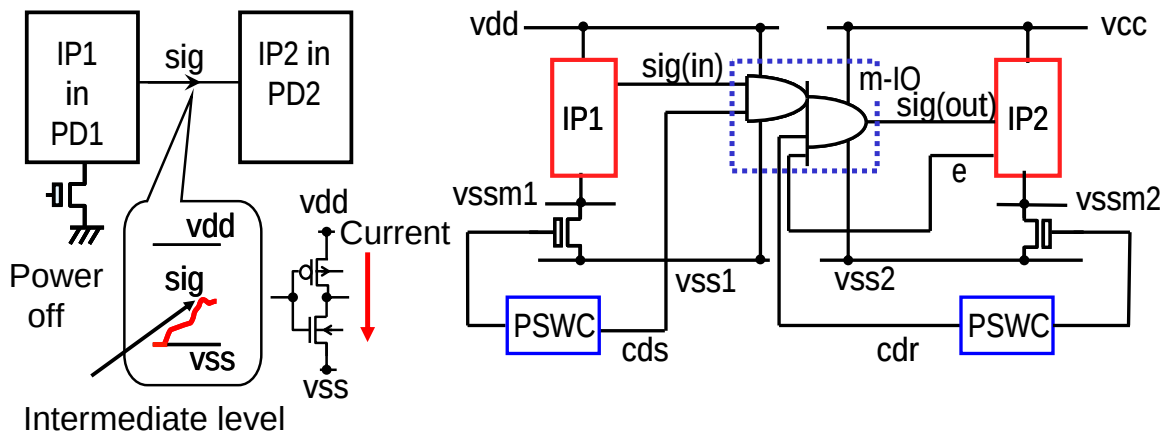


Figure 5.2 Micro-IO

Figure 5.3 shows a configuration of backup latches. Backup latches are placed in common power domain (CPD). Some key information such as the states of the control register, the clock, and the interrupt setting and memory controller settings for achieving a quicker recovery is stored in the backup latches before the domain enters a power-off state. The data of the backup latches are restored to the original flip-flops by the hardware during a recovery operation.

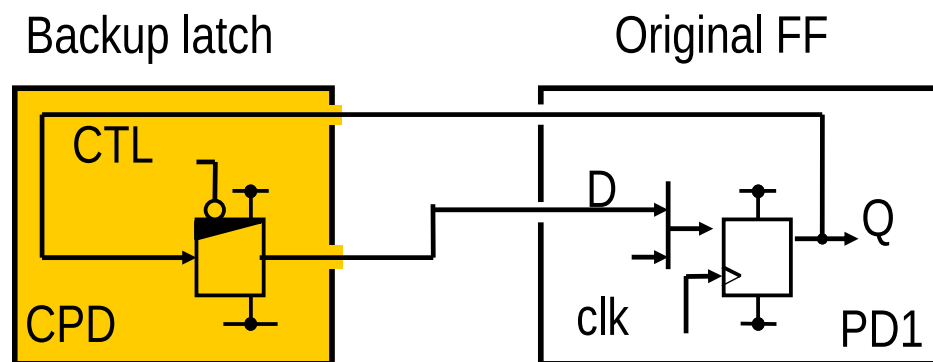


Figure 5.3 Backup latches

The power switch consists of a thick-gate-oxide high-V_t NMOS, which is used for I/O circuitry. The 3.3V I/O supply (VCC) is applied to the gate of the power switches to provide sufficient on-resistance, even under a low-voltage supply for core circuitry (VDD). The PSWC is shown in Figure 5.4 (above). To achieve a fast transition time between the power-off and power-on states without a rush current, we developed an accurate gate-voltage sensing scheme in combination with low-slew-rate driving scheme. A driver increases the voltage of the gates of power switches to 90% of the V_{cc} voltage. This sensing scheme detects that 90% drive of the gate of the power switches using a dynamic comparator, which effectively reduces the time margin. The dynamic comparator can be disabled without requiring the period of the power-on sequence of the power switch. Thus the standby current of the PSWC can be suppressed. Furthermore, the footprint of the dynamic comparator and the reference voltage generator is small compared to the conventional timer circuit, so the area overhead can be reduced to about 1/2 compared to that of the conventional one.

Turning on the power switches may cause a large rush current in the power line. This large rush current induces a voltage drop such as an IR drop and an Ldi/dt drop. When designing the power switch control scheme, we must be careful not to flow this large rush current during the power-on sequence of the power switch. To avoid rush current, a low-slew rate driving scheme is the most effective and our PSWC adopts two stage driving technique as shown in Figure 5.4 (below). The gate of power switch is driven by at a low-slew-rate by using two drivers. In the first stage the gate is driven by a

small driver. After Schmitt buffer detects that the gate is more than the threshold voltage, a large driver drives again to keep it at low impedance.

Figure 5.5 shows the common power domain (CPD) implementation. The CPD prevents the long global signal transmission and reduces the long recovery time with minimal area overhead. As shown in Figure 5.5, the CPD is distributed over the entire SoC. As an example of a long global signal transmission, global clock buffers can be placed in the distributed CPD, while local clock buffers are placed in each power domain. Therefore, having 20 of power domain partitions does not increase the clock skew. Signal repeaters can also be placed in the distributed CPD. In addition backup latches are placed in the CPD near the original FFs.

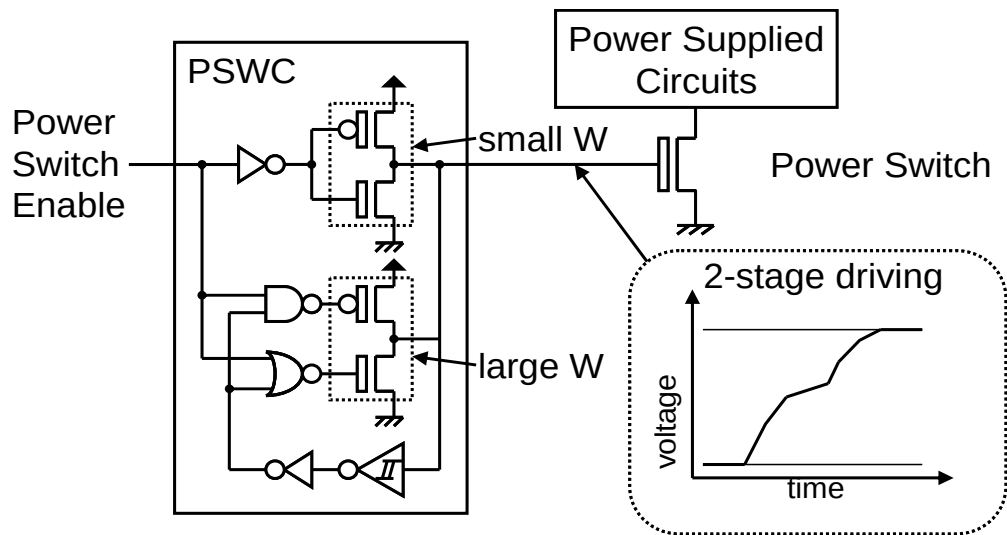
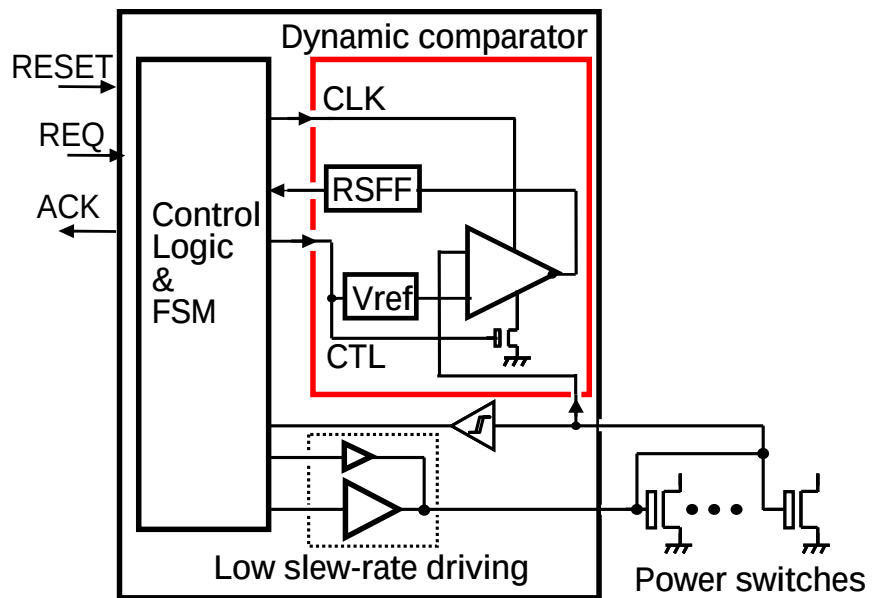


Figure 5.4 Power switch controller and 2-stage driving

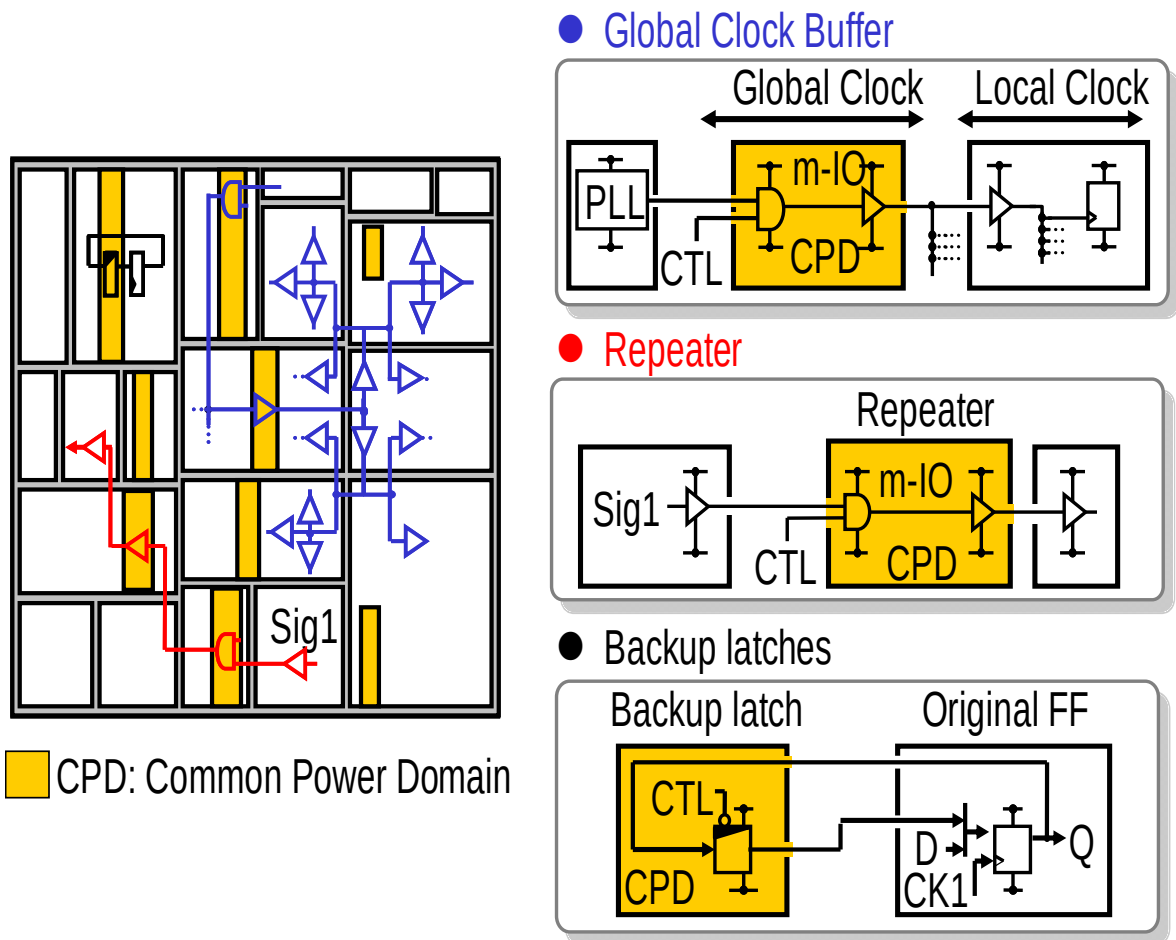


Figure 5.5 Common power domain implementation

5.3 Hierarchical Power Domain Control of Multi-core SoC

5.3.1 Background

Our first generation of one chip integration of an application and a baseband processor, SH-MobileG1, has 3 domains [24]. This 3-domain configuration is very similar to the conventional 3 chip cellular phone system as shown in Figure 5.6. Each domain has a CPU that runs a different OS. SH-mobileG1 has 20 power domain separations for very low leakage current. The major benefits of the 3-domain architecture is to (1) maximize the reuse of huge software assets of conventional cellular phones by keeping the same system architecture; (2) enable separate system development in cellular phone design; (3) introduce different efficient power shut down schemes according to the cellular phone use for the reduction of leakage power consumption; (4) reduce the conflicts between each subsystems; and (5) support separate dynamic clock frequency changes for each domain for the reduction of dynamic power consumption.

Our second generation SH-MobileG2 refines the design in both terms of performance and power dissipation keeping the 3-domain architecture [22]. The design highlights of SH-mobileG2 are (1) CPU core standby mode for two high-performance CPUs, where caches and RAMs are kept powered and logic part is shut-off, is introduced to reduce the leakage power; (2) dynamic clock-stop scheme is introduced, where the clock supply for bus-routers and bus-bridges are dynamically stopped while no bus transaction is in progress to reduce the dynamic power; (3) 512KB media-RAM is integrated which can be split up to 32 interconnect buffer (ICB) pieces which acts as read

fill buffers, write back buffers, or interconnect buffers through the ICB controller to enhance the performance of media IPs.

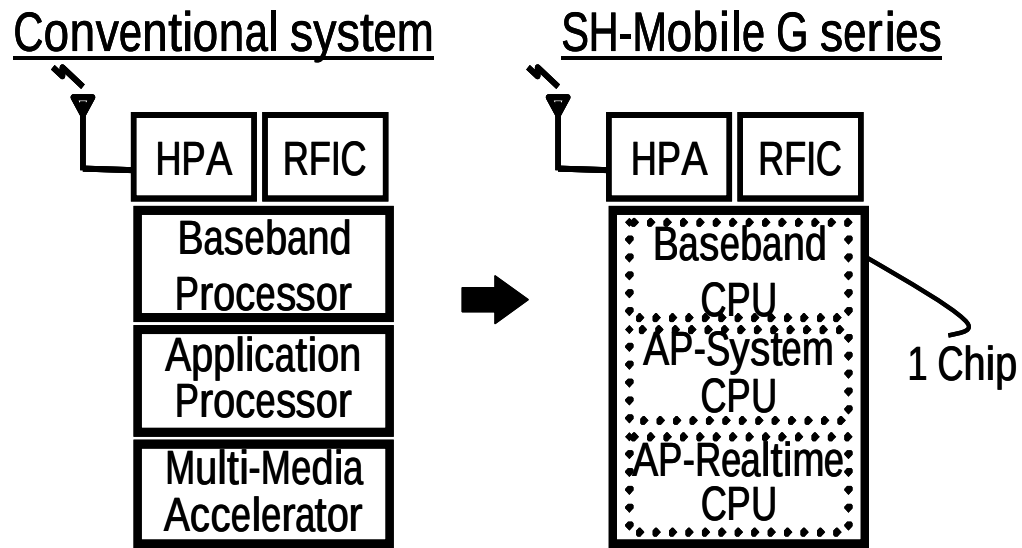


Figure 5.6 One chip SoC of SH-Mobile G series

5.3.2 Multi Power Domain SoC Architecture and Implementations

We have developed the third generation of a single-chip application and a baseband processor, SH-MobileG3, with a further evolved design in both terms of low power consumption and good performance keeping the 3-domain architecture of SH-Mobile G series. SH-mobileG3 has the following key features:

(1) Triple-V_{th} technology in low-power 65nm CMOS process achieves 500MHz for two CPUs.

(2) Power domain is separated into 20 sub-blocks to reduce leakage power. Core standby mode is applied to two high-performance CPUs

(3) Partial clock activation method described in the previous chapter is applied. Unnecessary clock systems are turned off in accordance with CPU cores and other modules activity.

(4) IP-MMU is newly introduced, which translates virtual address to physical address or physical address to physical address, for 18 different kinds of media IPs (2D-graphics, 3D-graphics, capturing, movie, displaying, blending etc), which contributes to reduce the size of external SDRAM.

(5) Interconnect buffer (ICB) is enhanced to incorporate with IP-MMU to give the better performance.

Table 5.1 shows SH-mobileG3 chip features and Figure 5.7 shows SH-mobileG3 chip micrograph. Die size is 9.3mm x 9.3mm. Supply voltages are 1.2V(internal) and 1.8/2.5/3.3V(I/O). SH-mobileG3 integrates 307M transistors in total, 28.2M gates and

30.7M memory bits. In an application part, a large number of media engines are integrated. Powerful 2D/3D graphics engines, video processing unit supporting D1 size (720 x 520) video decoding and encoding, and a camera-IF with image processing unit handles up to 12M pixel camera. A baseband part has a support for dual-baseband system. WCDMA block supports HSDPA service where as GSM block supports EDGE mode.

Figure 5.8 (above) shows CPU and bus architecture. SH-mobileG3 system architecture is multiple OS on heterogeneous multi-CPU cores. In the application part, ARM11 runs high-level OS and SH-X2 runs realtime media applications. In the baseband part, ARM9 controls the baseband IPs including DSPs that handle modem protocols. 2D and 3D graphics IP cores have IP-MMU inside between an on-chip-bus. 16 media IPs are connected to ICB that has another IP-MMU inside. One of the architectural challenges of SH-mobileG3 has super heterogeneous system where CPUs and IP-cores share single page table for virtual address space that was originally used only by OS-control CPU.

The power domain view of SH-mobileG3 is shown in Figure 5.8 (below). A total of 20 hierarchical power domains are defined to provide the useful power down control for the unused domains. The application part and the baseband part are divided into 13 and 6 power domains respectively with one additional common power area (C5). System controllers including PLLs are placed in C5 domain whose power is always on. We have assigned common power domain (CPD) described in the previous section to C5. Then clock buffers, clock dividers, repeater cells, and hardware back-up latches can be located in C5. This makes the implementation of CPD easier, because always-on power area (C5) can be located almost freely on a huge SoC.

Table 5.1 Feature of SH-mobileG3

Die size	9.3mm x 9.3mm
Process	65nm LP
Supply voltage	1.2V (internal), 1.8/2.5/3.3V (I/O)
# of TRs, gate, memory	307M TRs (28.2M Gates, 30.7Mb-RAM, 6.4Mb-ROM)
CPU	ARM926EJ-S,166MHz [modem control] ARM1176JZF-S,500MHz [application] SH-X2,500MHz [multimedia]
2G/3G modem	WCDMA/HSDPA category 8, GSM/GPRS/EDGE
Multimedia hardware engines	VPU(MPEG4/H.264/VC-1 D1 size), VIO (Camera interface 12M pixels), 3D/2D Graphics, GPS, LCDC, Sound Processing Unit(SPU), 512KByte MediaRAM(MERAM)
I/O	617 pins

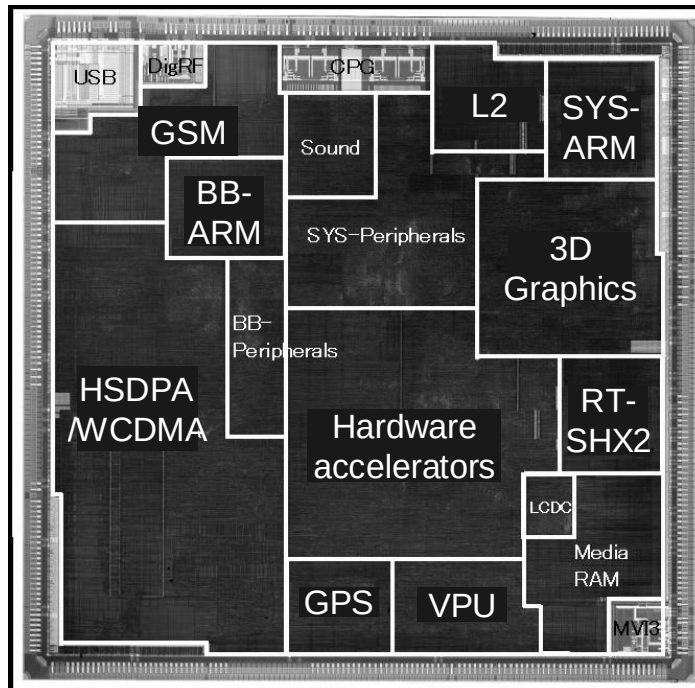


Figure 5.7 Chip micrograph of SH-MobileG3

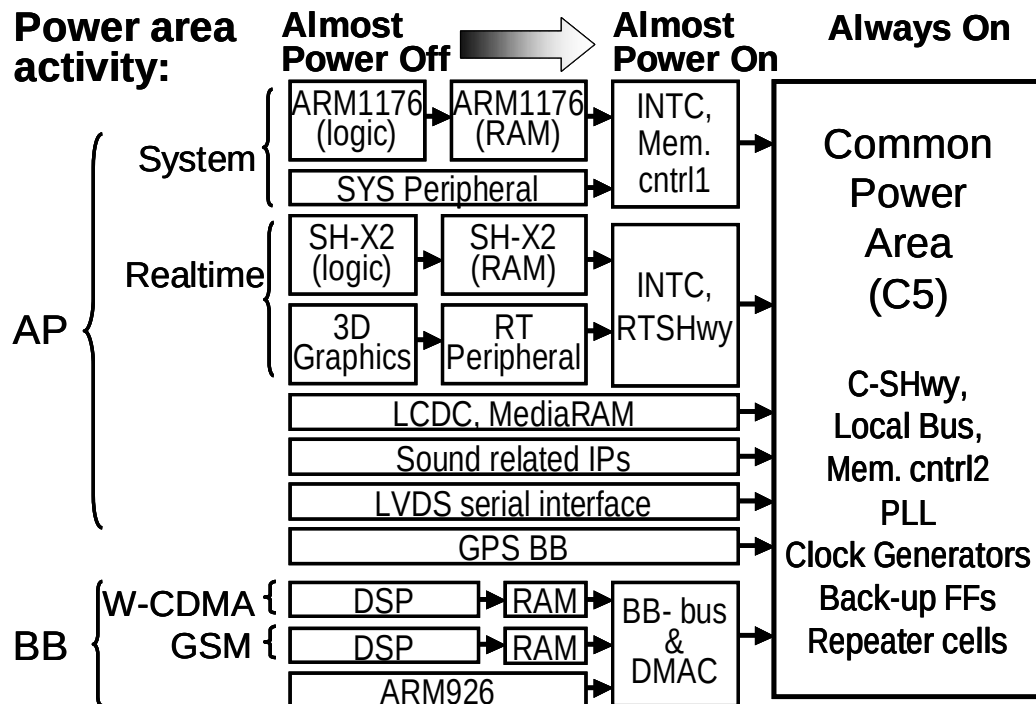
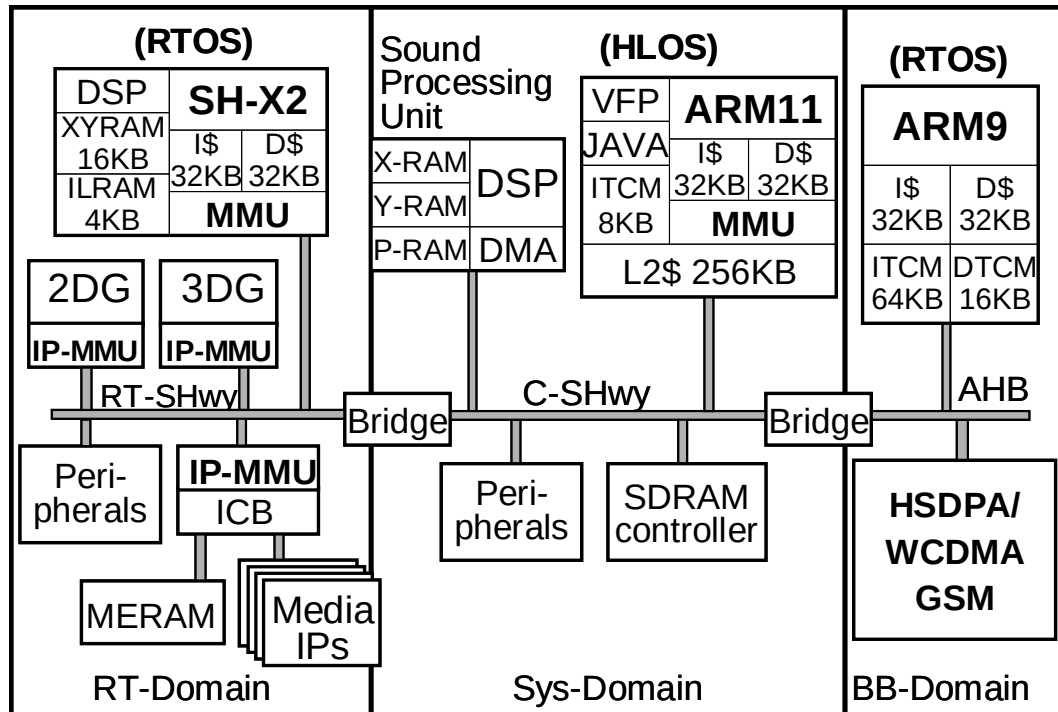


Figure 5.8 CPU/bus architecture and hierarchical power domain of SH-mobileG3

We have applied three power-tree management rules for our hierarchical power domain structure as shown in Figure 5.9. This enables to minimize the number of micro-IO cell insertions and make logic verification easy. The top level of the power distribution hierarchy is a permanently powered-on domain (C5). Rule 1 is that domains on lower levels of the hierarchy can be in the powered-on state only if the domains on the higher levels of the hierarchy are on. Rule 2 is that micro-IO cells are placed in a higher level of the hierarchy when a signal is transmitted from a lower level to a higher one in the hierarchy. Rule 3 is that a signal can transit on the graph of power tree.

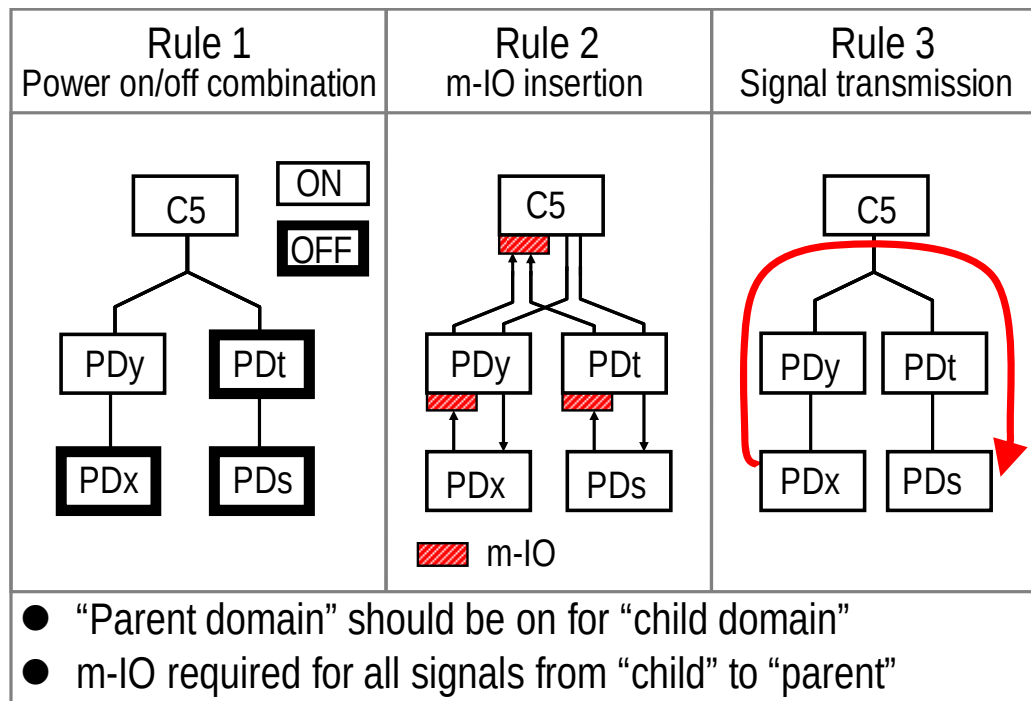


Figure 5.9 Three power-tree management rules

5.3.3 IP-MMU

In the cellular phone system, efficient memory management is required, because ever-increasing diversity of applications require huge size of external memory. By reducing the size of SDRAM, the leakage power for SDRAM can be reduced. The high-level OS controls and maps the physical memory in the cellular phone system by page tables to maintain virtual address translation. Only one CPU in a SoC normally uses this virtual address to physical address translation. Hardware media IPs typically use fixedly allocated memories and physical memories cannot be re-configured even when the media IP finishes its operation. For this reason, large size (20 to 80MB) of external memory is always allocated statically for media-IPs. In order to effectively make use of external memories, SH-mobileG3 has introduced IP-MMU, which is attached to media IPs. As shown in Figure 5.10, virtual address space can be allocated when necessary and can be freed after its operation. Some HW-IPs will use physical addresses by their nature, while other HW-IPs use virtual addresses sharing an IP-MMU. In order to avoid address conflicts of such cases, IP-MMU supports physical address space mapping buffer (PMB) together with translation lookaside buffer (TLB). Here, address space for HW-IPs is separated into two regions: virtual-to-physical and physical-to-physical.

Basic IP-MMU configuration of our design is shown in Figure 5.11(a). Each IP-MMU has 8 micro-TLBs cached from either 16 entries of fully associative PMB or 64 entries of 2way set associative main TLB. The PMB keeps address for physical-to-physical translation and the main TLB keeps address for virtual-to-physical translation. Whether the PMB or the main-TLB is used is decided according to the uppermost bits of

address issued by media IPs. IP-MMU pre-fetches additional page table information for main TLB when main TLB miss happens. This effectively decreases the overhead of TLB fills as media IPs tend to access contiguous memory. 18 HW-IPs of SH-mobileG3 use IP-MMU for address translation. Two HW-IPs, namely 2D-graphics and 3D-graphics, adopt this basic IP-MMU.

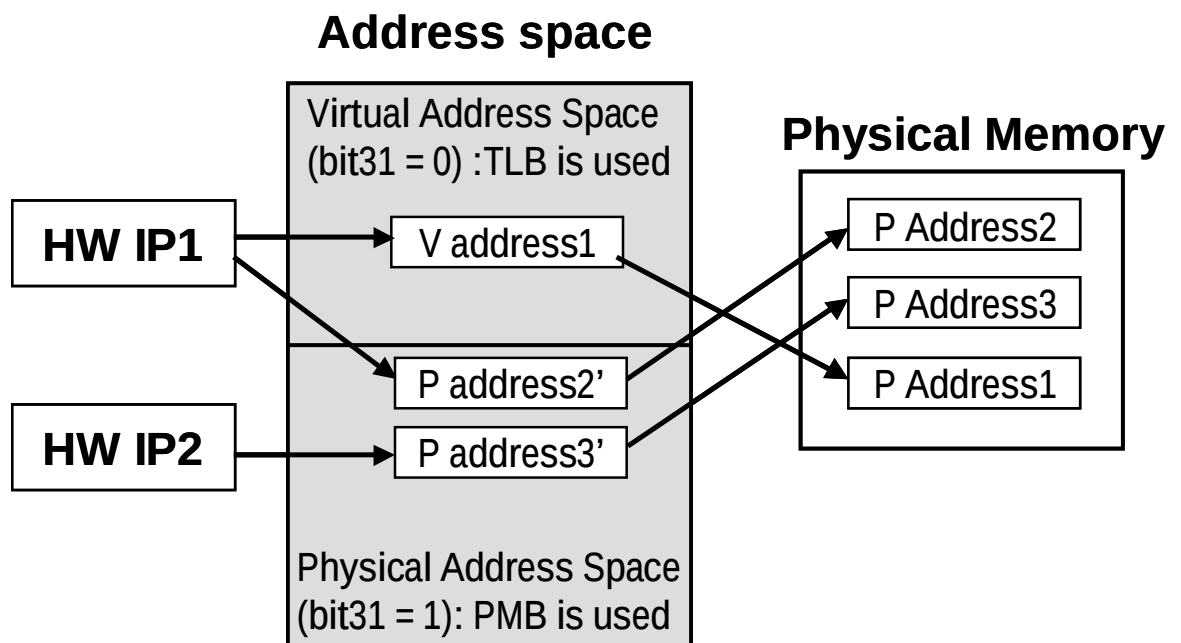


Figure 5.10 Virtual and physical address translation

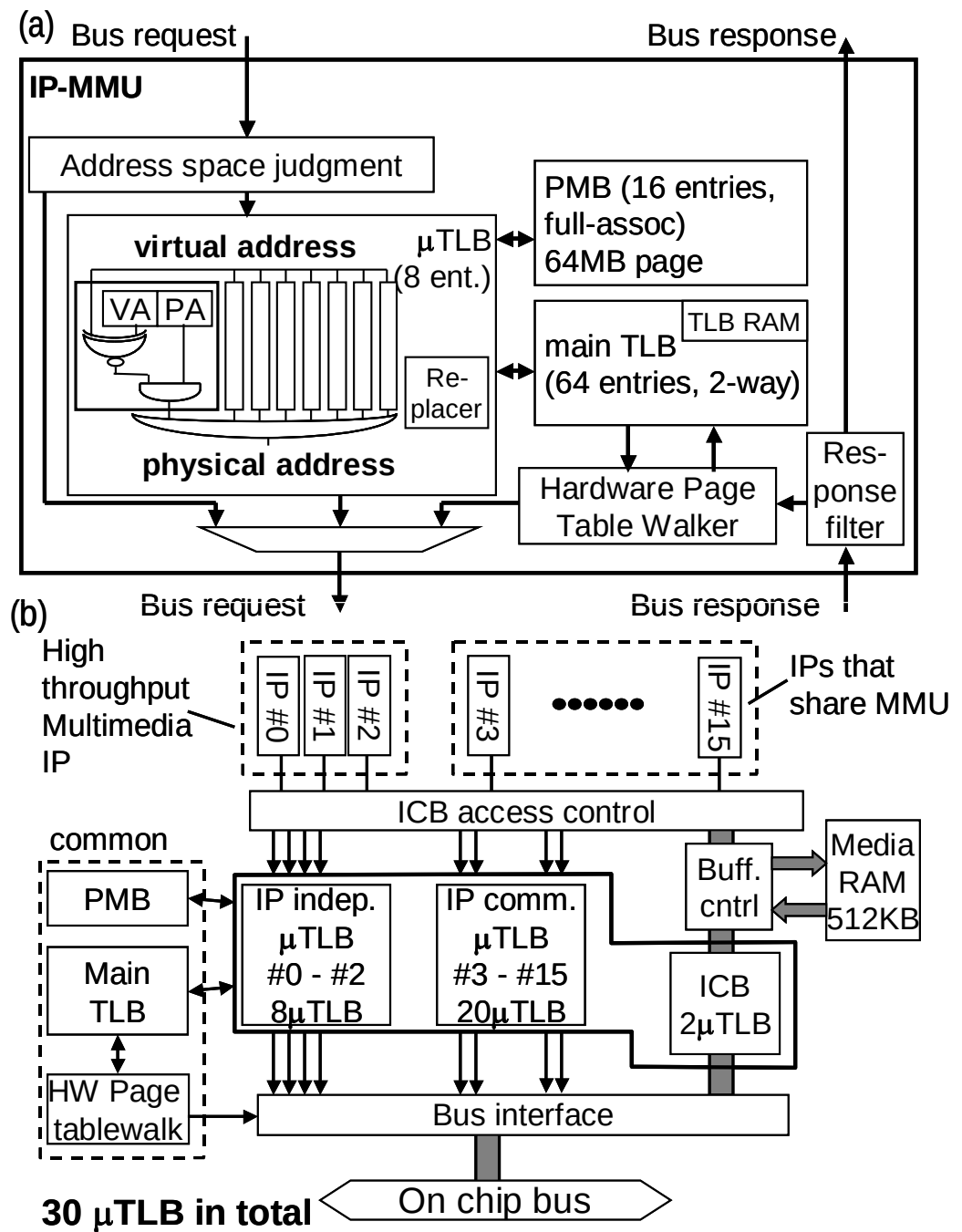


Figure 5.11 (a) Basic IP-MMU configuration
(b) Optimized implementation of MMU in SH-MobileG3

The other 16 HW-IPs are different kinds of media IPs and access patterns by these media IPs are also different. In order to reduce the number of IP-MMUs and yet get the good benefits of the IP-MMUs, 16 HW-IPs are categorized into three and SH-mobileG3 has adopted three optimized IP-MMUs for them as shown in Figure 5.11(b). One IP-MMU is dedicated for three high-throughput media IPs and has 8 micro-TLBs. Another IP-MMU is shared by the other 13 HW-IPs and has 20 micro-TLBs. The other IP-MMU has two micro-TLBs. This IP-MMU is used in a case where HW-IPs make accesses to the on-chip-bus through interconnect buffer (ICB). ICB is the bus bridge with buffering features using 512Kbyte media-RAM. Complicated accesses from HW-IPs can be buffered in ICB and re-ordered into simple line directional (address incremental) access and TLB miss will not be frequent. These three IP-MMUs share one PMB and one main TLB that are also located inside ICB. They have 30 micro-TLBs in total, which is about one-fourth of 128 micro-TLBs if we adopt 16 basic IP-MMUs for each HW-IP instead.

SH-X2 CPU also has MMU, which consists of TLB and PMB very similar to IP-MMU. TLB contents are registered by the software control from the main page table managed by the high-level OS that is working on ARM11 CPU. PMB contents are registered by the software control. SH-X2 also takes care of IP-MMU control such as PMB settings to minimize system performance impact.

By this memory management scheme, not only two heterogeneous CPUs namely ARM11 and SH-X2, but also HW-IPs can share the page table located in the main memory and contribute to the effective use of physical memories. A big interest of this system is the data throughput when media IPs access concentrated at the same time. The

throughput of data does not decrease when the virtual memory translation is enabled and achieves VGA 30fps video cam-coder scenario.

Before using IP-MMU scheme, statically allocated external memory size was 53MB, 35MB for camera display and 18MB for graphics. By utilizing IP-MMU scheme, only 10MB for camera display is required for statically allocation and 43MB memory size can be freed and therefore reduce the same size of external memory. This contributes to leakage power reduction regarding to external SDRAM. As for the video camcoder application performance, D1 (720 x 520) size, 30 frames-per-second for MPEG4/AVC format is confirmed.

5.4 Independent Power-off Control of CPU and SRAM of Multi-core SoC

5.4.1 Background

Looking at the market trends for embedded processors of such fields as digital TV or car information system, both high-performance and low power consumption are required for such processors. Multi-core SoC is a good solution for this purpose, same as in the way for high performance PC or server.

For the higher performance, multi-core SoC with more than 4 CPU cores will be required for the embedding use in the near future. As the number of CPU core increases, good feedback to a compiler has become more and more important to achieve good parallelism. Another key point will be an effective synchronization mechanism among CPU cores working in parallel [26]. On the other hand, requirements on power dissipation are getting more and more severe especially for the embedding processors.

In order to satisfy this requirement, leakage power saving for CPU cores will be a key point in addition to various conventional dynamic power saving schemes. This is because leakage power will be more dominant as the process generation proceeds and also because high-performance multi-cores use leaky high-speed transistors for CPU core sections. Table 5.2 shows our 8-core SoC features. First, the 8-processor core SoC is capable of executing 8640 MIPS at 600MHz for Dhryston2.1. It consumes 2.8W at 1.0V including leakage power. Second, the 8 cores and 8 user RAMs are assigned to split power supply domains so that the power supply of unused CPU cores can be cut off to reduce the leakage power. For the power-off, RAM resume mode can be selected for the

fast restart on wake up. An automatic parallelizing compiler analyzes the parallelism of the program and optimizes total power consumption by controlling each CPU's power mode including power-cutoff. This SoC is fabricated in a 90nm, 8 metal layers, and triple-V_{th} CMOS process technology. The chip size is 104.8mm^2 . Each CPU core size is 6.6mm^2 . Supply voltages are 1.0V – 1.4V for internal and 1.8V/3.3V for IO.

Table 5.2 Eight-core SoC features

Process Technology	90nm, 8-layer, triple-V _{th} , CMOS
Chip Size	104.8mm^2 (10.61mm x 9.88mm)
CPU Core Size	6.6mm^2 (3.36mm x 1.96mm)
Supply Voltage	1.0V–1.4V (internal), 1.8/3.3V (I/O)
Power Domains	17 (8 CPUs, 8 URAMs, common)

5.4.2 Multi-core SoC Architecture

Figure 5.12 shows the basic architecture of our SoC. This SoC consists of two clusters where each cluster has 4 CPU cores. Each processor-core contains a CPU, an FPU, I/D caches and their controller CCN, I/D local memories, and user RAM or URAM.

Inside a cache controller, we implemented barrier registers. Within the cluster, there is a snoop controller that maintains cache coherency among CPU cores. For each CPU core we have a local clock pulse generator and power control registers. 8 CPU cores are connected to on-chip system bus called SuperHyway, which is a packet-based split transaction bus system. Memory controllers such as DDR2-SDRAM, SRAM, and DMA controllers are also connected through SuperHyway. 17 power domains, namely 8 for each CPU core, 8 for URAM, and 1 common, are implemented.

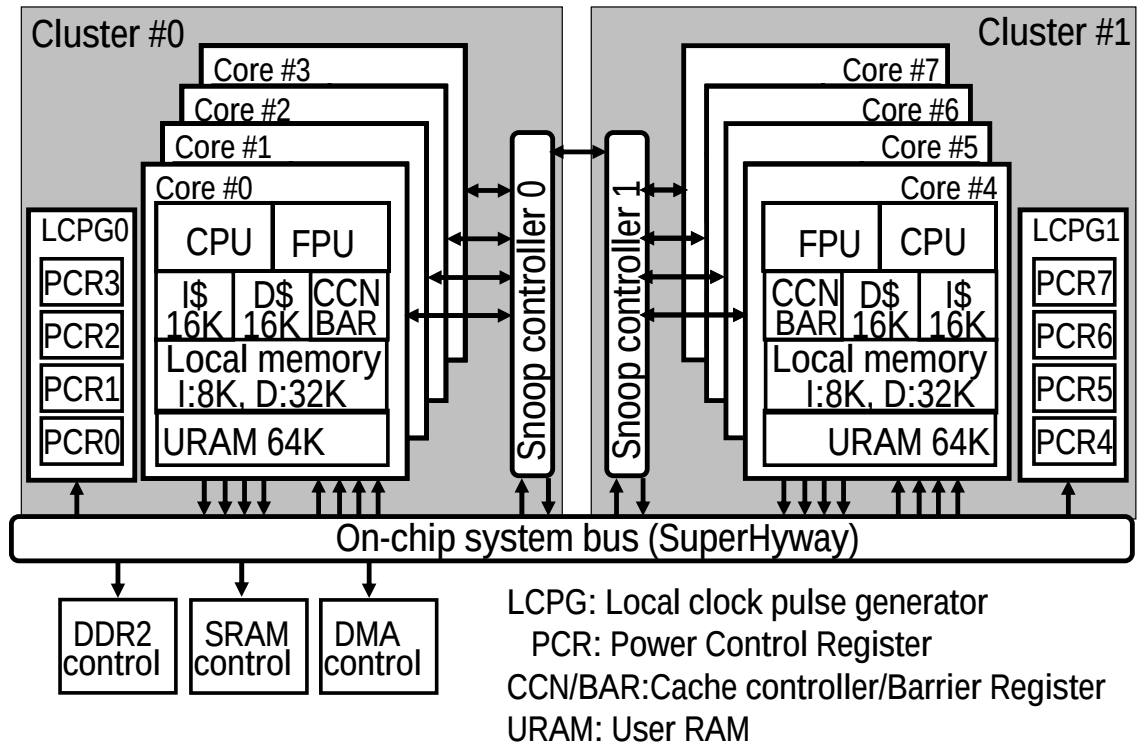


Figure 5.12 Multi-core SoC block diagram

5.4.3 CPU Power Modes

Our previous multi-core SoC [27] supports three power modes for CPU. They are Normal, Light Sleep, and Sleep. In the Light sleep mode, clock supply for CPU is stopped but is continued for cache to maintain cache coherency since other CPUs are working using caches. In the sleep mode, clock supply for all the modules inside the CPU core except URAM is stopped. In order to achieve further power savings for unused CPU cores, we have introduced two additional CPU power modes; they are resume power-off and full power-off. In the full power-off mode, power supply to the CPU core is switched off to eliminate leakage power completely. In the resume power-off mode, power supply for all the modules inside the CPU core is stopped except URAM. As the data in URAM is retained during the resume power-off mode, the CPU can resume the operations quickly on wake-up from the resume power-off mode using data retained in URAM. Here, 8 CPUs are able to select appropriate power modes independently. Note that core-standby mode described in section 3.3 can be technically defined in addition to these five power modes, we have not applied it, because it is too complicated to implement both core-standby mode and Resume Power-off mode at the same time.

Table 5.2 CPU power modes

Power modes	Normal	Light Sleep	Sleep	Resume Power-off	Full Power-off
CPU	active	Clock:off Power:on	Clock:off Power:on	Clock:off Power:off	Clock:off Power:off
Cache	active	active	Clock:off Power:on	Clock:off Power:off	Clock:off Power:off
URAM	active	active	active	Clock:off Power:on	Clock:off Power:off

5.4.4 Implementation of Multi-core SoC

Figures 5.13 and 5.14 show power domain implementation of the SoC and each core, respectively. A total of 17 power domains are identified in the SoC. 8 power domains are allocated to each CPU core. Inside the CPU core power domain, URAM power area is isolated so that the power supply can be independently controlled with CPU core. In addition to these 16 power domains, there is one common power domain that is always-on. Each power domain is surrounded with power switches for each CPU core and each URAM. These power switches are controlled by a power switch controller, VSWC, which are controlled by power control register implemented inside the local clock pulse generator module (LCPG) for each core.

Figure 5.15 shows the power control register, or PCR for short, contains 4 control bits, namely RESET, RESUME, POFF, and LSLEEP. Using these 4 bits, CPU transits to 5 power modes, namely Normal, Light sleep, Sleep, Resume power-off, and full power-off. Only the power state transition from and to Normal state is possible. In order to make a transition to Light-sleep and Sleep states, the CPU executes Sleep instruction with LSLEEP bit set to 1 and 0, respectively. These sleep states come back to Normal state triggered by any interruptions. In order to make a transition to Resume power-off and full power-off states, the CPU executes PCR register updates instruction with RESUME=1 and POFF=1, respectively. These power-down states come back to Normal state triggered by PCR register updates with RESET bit set to 1. These PCR settings can be done from any one of the CPUs awake. Transition time required to shift from normal state to power-

off state is 5us and from the power-off state to go back to normal state is 30us. State transition from and to clock-off state is made immediately without any overhead time. On the other hand, URAM data cannot be retained in “Full Power-off” mode. If URAM data are saved before power-off and are filled after power-on, overhead time for transition and recovery time is usually an order of 100us.

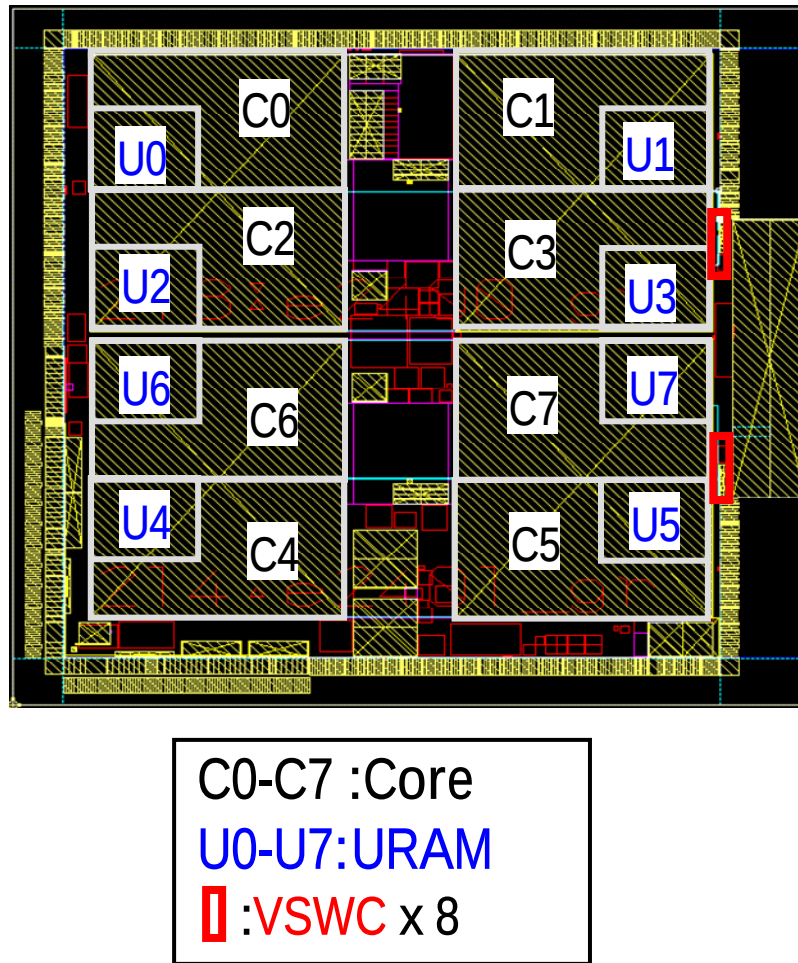


Figure 5.13 Power domain implementation of 8-core SoC

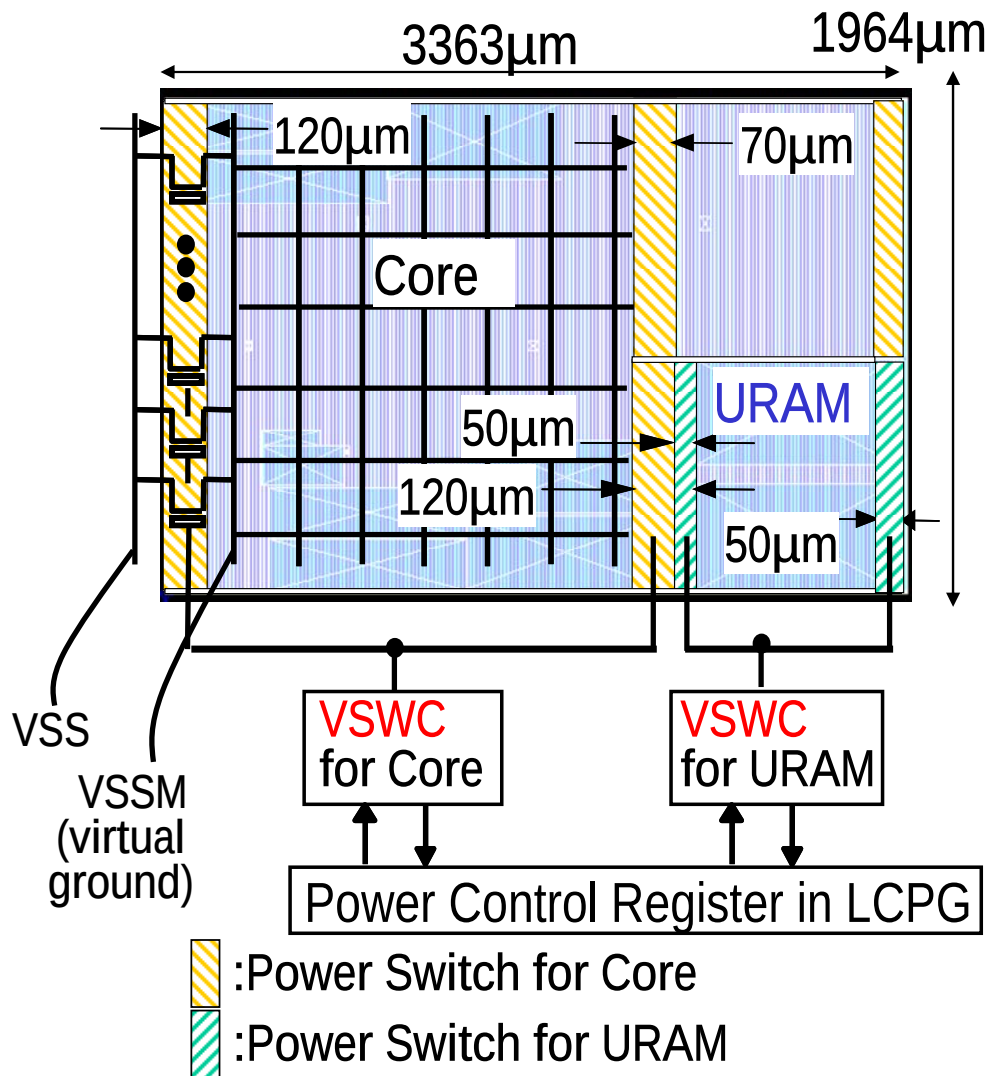


Figure 5.14 Power domain implementation of each CPU core

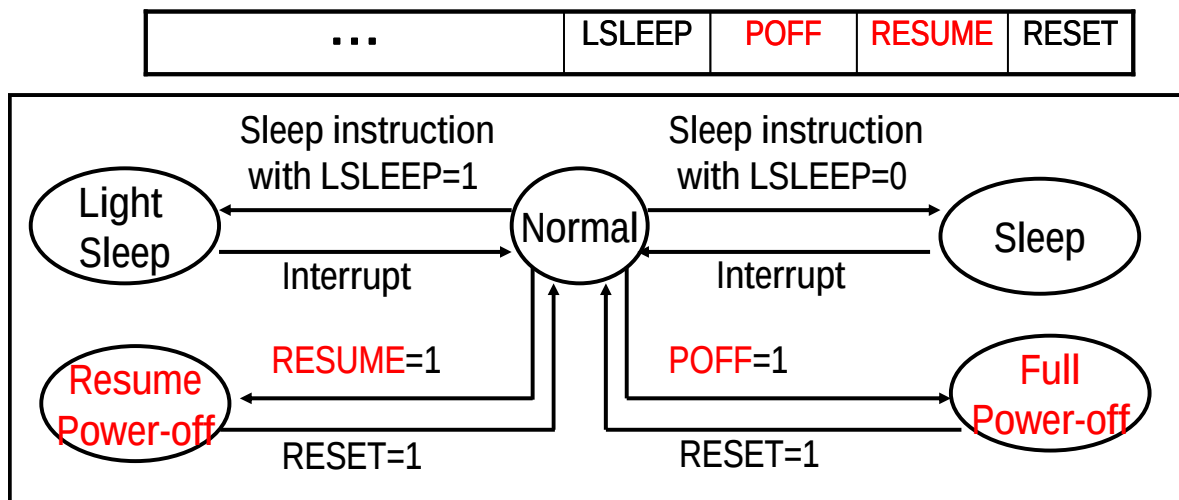


Figure 5.15 Power control register

5.4.5 Evaluation Results

Figure 5.16 shows measurement results of each power mode. The graph shows the dynamic power and leakage power data measured for 8 CPU cores at a room temperature at 1.1V. Power of normal case is measured by an idle-loop program. The total power consumption for the sleep mode amounts to 304mW and the leakage power accounts for its 70%. The RESUME mode has only 35mW leakage power of URAM block and saves 88% power. These results show that the additional power-off modes besides the clock-off modes greatly contribute to the total power reduction. Therefore, it is very important to find free CPUs or generate free periods for CPUs and control CPU power modes including power cut-off.

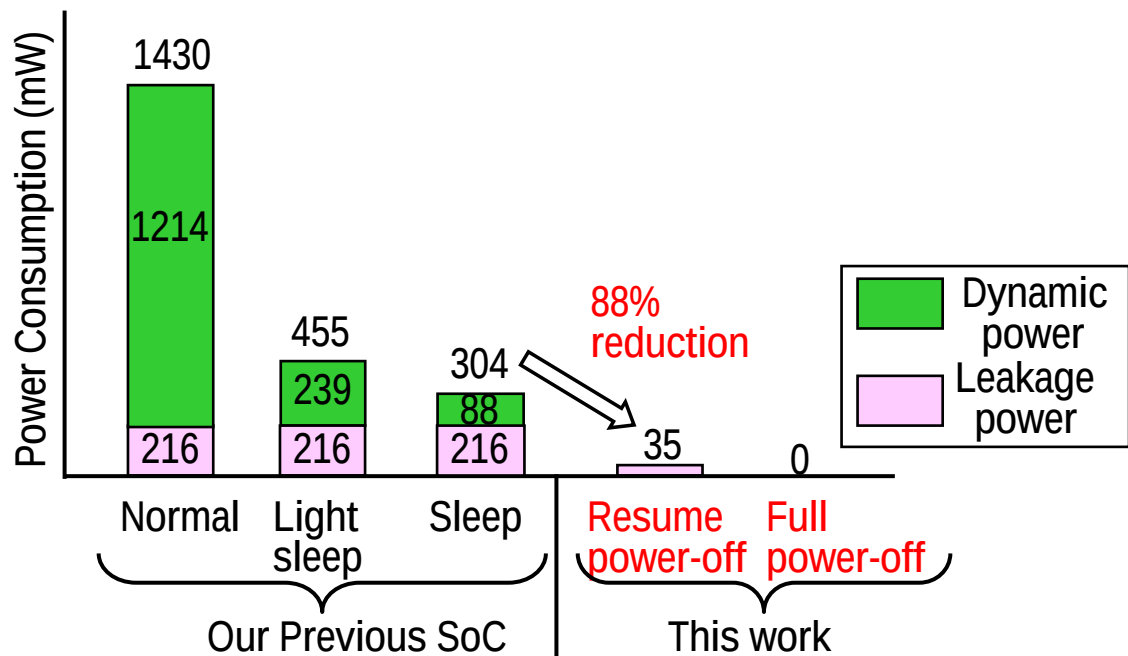


Figure 5.16 Power consumption of 8 CPU cores

Figure 5.17 shows power consumption evaluation result for real-time AAC encoding for 8 CPUs by the compiler. We have measured the power consumption with and without power control of CPU at 1.0V. The left side graph shows power consumption without power control. In this case, the average power consumption is 2.37W. The right side graph shows power consumption with power control using resume power-off mode. In this case, the average power consumption is 0.71W. Since the evaluation program is real-time AAC encoding using 8 cores, CPUs are free for the most of each period time 0.19 second. So power-down control by the compiler for such idle period during one application execution is very effective and we have achieved 70% power reduction.

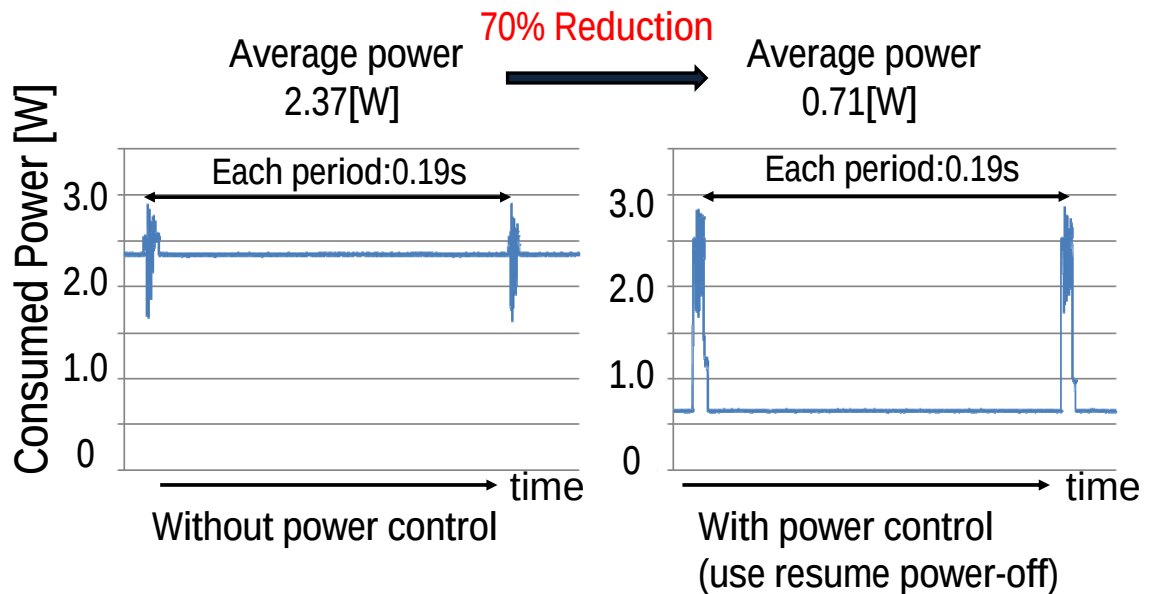


Figure 5.17 Evaluation results of 8-core SoC power consumption

5.5 Summary of Chapter 5

In order to handle leakage power amount and ratio increases in sub-100nm generation SoC, power domain separation to shutdown unused circuits is essential for any field. Required components and techniques for multi power domain have been explained.

We have developed a multi-power domain SoC which is a third generation single-chip baseband and application processor with media engines for 3G cellular phone system at 65nm LP process. Our single chip SoC contains 20 power domains and has applied hierarchical power domain management for overhead reduction such as complicated control logic and isolation cells. We also have applied core standby CPU technique as described in section 3.3 in this SoC as well.

We have developed an 8-processor core SoC for embedded applications. We have implemented 16 split power supply domains for each of the 8 processor cores and 8 RAMs so that the power supply of unused CPU cores can be cut off and RAM resume mode can be selected for fast restart on wake up. Our chip evaluation results show that each CPU's power mode control achieves 70% power reduction for real-time AAC encoding executed by 8 CPUs.

CHAPTER 6: Conclusions

6.1 Summary of the Thesis

This thesis has explored how power consumption, both dynamic power and static power, can be reduced in functional-design level of microprocessors and SoCs. In the functional-design level, which corresponds to the micro-architecture level in case of microprocessor design, how to achieve the specification is the main points. In this design-level, any individual technique can contribute to a large power consumption reduction, which increases the value of microprocessors and SoCs that adopt the low-power functional design technique. In this thesis, low-power techniques both for dynamic power reduction and static power reduction for microprocessors and SoCs have been researched and discussed in Chapters 2-5.

In Chapter2, low-power functional design techniques for arithmetic units are discussed. We have proposed a novel low power multiplication algorithm for reducing switching activity through operand decomposition. The power savings with the proposed method are 13% and 10% respectively for the 64-bit array and tree multipliers. Our algorithm is effective for both array and tree multipliers of 32 bits or larger widths. The proposed algorithm can be practically applied to many digital systems, where minimizing power consumption is important. For the calculation of square root, we have proposed a new converging algorithm. The iterative formula of our algorithm consists of one multiply-accumulate operation and can be executed in one cycle on a multiply-add unit. When we calculate double-precision square roots in 6 cycles, our method requires only

1/8 the size of look-up tables compared to the multiplicative Newton-Raphson method, and thus achieves the same ratio of leakage power of the lookup tables. We have also proposed an efficient initial approximation method with an accelerated higher order converging algorithm for division. When we calculate double precision division at 4 clock cycles, our method requires only 1/6 the size of the lookup table compared to the linear approximation method, and thus achieves the same ratio of leakage power reduction of the lookup table. Since the proposed square rooting and division schemes are all based on a high-speed multiplier, the low power multiplier proposed in section 2.2 can also be applied and the power reduction effects can all be achieved at the same time.

Chapter 3 discussed low-power functional design for CPU-cores using low leakage onchip-SRAM. We have introduced high-performance and low-power CPU using low-leakage SRAM with resume-standby support and a new CPU power saving mode called core-standby mode. In the core-standby mode, CPU logic is powered-off while Cache and internal-SRAM keeps powered to resume level at which the data can be retained. Evaluation results show that the transition time and recovery time to and from the core standby mode is 5 μ s and 7 μ s including resource push and pop time and power switch transition time. With only this slight overhead for transition and recovery time, we can achieve high CPU core frequency by using lower-V_t transistors. The measured leakage power on silicon at room temperature is only 0.02mA, which is a 98% reduction compared with the conventional sleep-mode of about 1.0mA. These results show that the core standby mode cuts almost all leakage current.

In Chapter 4, clock-gating techniques for dynamic power consumption of SoCs have been discussed. We show that the conventional clock-gating techniques such as static clock-gating by setting clock control registers and dynamic clock control by the hardware applied by logic synthesis tools are not sufficient for the current large SoCs. We have proposed two clock-gating techniques that can be designed at functional design level. One is dynamic clock control for large IP, with dynamic module-stop scheme at sub-block level. The other is dynamic clock control for bus-related modules, with dynamic module-stop by new bus transaction request detection. Our evaluation results show that the dynamic module-stop techniques reduce current more than 35mA out of 149mA and 109mA for video telephony and picture playback applications, respectively. Dynamic module-stop techniques can be applied to almost all modules that work intermittently and have a certain amount of idle state where clock supply is not needed. We have applied a more sophisticated dynamic power reduction of clock-gating at more system control level called partial clock activation method, where clock systems including PLLs are separated and only activated when necessary. Active power in common area is reduced to 33%. 40% of SoC power reduction has been achieved for music playback.

Chapter 5 discussed static power reduction techniques for SoCs. We have developed a multi-power domain SoC which is a third generation single-chip baseband and application processor with media engines for a 3G cellular phone system at 65nm LP process. Our single chip SoC contains 20 power domains and has applied hierarchical power domain management for overhead reduction such as complicated control logic. We

have also described an 8-processor core SoC for embedded applications. We have implemented 16 split power supply domains for each of the 8 processor cores and 8 RAMs so that the power supply of unused CPU cores can be cut off and RAM resume mode can be selected for fast restart on wake up. Our evaluation results show that each CPU's power mode control achieves 70% power reduction for real-time AAC encoding executed by 8 CPUs.

6.2 Perspective

All the low power techniques proposed in this thesis are at functional-design level and many of the techniques have been applied to real chips in the market. The power reduction effects are different one by one but the techniques surely have produced additional value for the chip. Some of the techniques have given a good power reduction ratio with a slight trade-off in performance. Some of the techniques have given a substantial power reduction with a slight area overhead. All of the techniques cannot be achieved automatically by EDA tools. This research theme can only be handled by human designers.

Today, for most SoC designs, the power budget is the most important design goals of the project. Exceeding the power budget can be fatal to a project, whether it means moving from a cheap plastic package to an expensive ceramic one, causing an unacceptably poor reliability due to excessive power density, or failing to meet the required battery life. These problems are all expected to get worse as we move to the next technology nodes. This gives a significant effect on how chips are designed. The power density of the highest performing chips has grown to the point where it is no longer possible to increase clock speed as technology shrinks. As a result, designers are designing multi-processor chips instead of chips with a single, ultra-high speed processor.

As such paradigm shifts for architectures occur, new functional-level low power techniques are always desired in value-added chip development both from chip providers

and markets. I believe this research theme continues to be an important position for the engineers and researchers in this field.

ACKNOWLEDGEMENTS

My gratitude is especially extended to Professor Hideo Maejima of Tokyo Institute of Technology, Interdisciplinary Graduate School of Science and Engineering, Department of Information Processing, for his giving me an opportunity to do research at his institute. His continuous suggestions and encouragements allowed me to complete my thesis successfully. I would like to acknowledge Professor Yoshinori Hatori, Professor Takao Kobayashi, Associate Professor Minoru Kurosawa, and Associate Professor Nobuhiko Sugino, all of Tokyo Institute of Technology, Interdisciplinary Graduate School of Science and Engineering, Department of Information Processing, for their fruitful discussions and suggestions.

My greatest appreciation goes to Professor Naofumi Takagi of Kyoto University. He led and supported my research from 1992 when I started this research and received my B.S. and M.S. degrees in computer engineering at Kyoto University. Through his enthusiastic leadership, I gained a good attitude and approach for research.

I would like to express my sincere gratitude to Professor Kurt Keutzer and Professor Borivoje Nikolic of University of California at Berkley for their warmhearted encouragements during my one-year research as an industrial fellow of University of California at Berkeley. I also thank Dr. David Chinnery for his valuable comments about my research at UC Berkeley.

I would like to thank all of the co-authors of the papers I published. I would also like to especially thank Dr. Toshihiro Hattori of Renesas Electronics for giving me an

opportunity to work with mobile SoCs and for giving me chances to do many presentations in international conferences. I also thank my group and the related people in Renesas Electronics.

Finally I would like to extend my deepest thanks to my wife Rieko Ito and our twin daughters Aoi and Mai for their strong supports, deep understanding and affectionate encouragements.

REFERENCES

Chapter 1

- [1] Fumio Arakawa, Osamu Nishii, Kunio Uchiyama, Norio Nakagawa, "SH4 RISC Multimedia Microprocessor," *IEEE Micro*, vol. 18, no. 2, pp. 26-34, Mar./Apr. 1998
- [2] M. Ito, et al., "A 65nm Single-Chip Application and Dual-Mode Baseband Processor With Partial Clock Activation and IP-MMU", *IEEE Journal of Solid-State Circuits*, Vol.44, No. 1, pp. 83-89, Jan. 2009
- [3] J. M. Rabaey, A. Chandrakasan and B. Nikolic, *Digital Integrated Circuits (Second Edition)*, Prentice Hall, 2003.
- [4] A. P. Chandrakasan, S. Sheng and R. W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, Vol. 24, 1992, pp.487-493.

Chapter 2

- [5] A. D. Booth, "A Signed Binary Multiplication Technique," *Quarterly J. Mechanical Applications in Math.*, vol. 4, part 2, 1951, pp.236-240.
- [6] C. V. Ramamoorthy, James R. Goodman and K.H.Kim, "Some properties of iterative square rooting methods using high-speed multiplication," *IEEE Trans. on Computers* C-21 (Aug. 1972), 837-847
- [7] D. Ferrari, "A division method using a parallel multiplier," *IEEE Trans. on Computers* EG16 (Apr. 1967) ,224226
- [8] D. Goldberg, "Computer Arithmetic", in D. A. Patterson and J. L. Hennessey, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, 1990.

- [9] M. J. Flynn and S. F. Oberman, *Advanced Computer Arithmetic Design*, Wiley-Interscience, 2001.
- [10] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers*, Vol. EC-13, 1964, pp.14-17.
- [11] V. G. Moshnyaga and K. Tamaru, "A Comparative Study of Switching Activity Reduction Techniques for Design of Low-Power Multipliers," *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1995, pp.1560-1563.
- [12] T. Ahn and K. Choi, "Dynamic operand interchange for low power," *Electronics Letters*, Vol.33, No.25, 1997, pp.2118-2220.
- [13] P.-M. Seidel, "Dynamic Operand Modification for Reduced Power Multiplication," *Proceedings of the 36th Asilomar Conference on Signals, Systems and Computers*, 2002, pp.52-56.
- [14] A. A. Fayed and M. A. Bayoumi, "A Novel Architecture for Low-Power Design of Parallel Multipliers," *Proceedings of the IEEE Computer Society Workshop on VLSI*, 2001, pp.149-154.
- [15] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Transactions on Computers*, Col. C22, No.8, 1973, pp.786-793.
- [16] D. L. Fowler and J. E. Smith, "An accurate high speed implementation of division by reciprocal approximation," *Proc. of 9th Symp. on Computer Arithmetic* (Sep.1989), 60-67

- [17] D. DasSarma and D. W. Matula, "Measuring the accuracy of ROM reciprocal tables," *IEEE Trans. on Computers*, Vol.43, No.8(Aug. 1994), 932-940
- [18] N. Anderson, "Minimum relative error approximations for $1/t$," *Numerische Mathematic* 54 (1988),
- [19] M. Schlute and E. Swartzlander, "Exact rounding of certain elementary functions," *Proc. of 11th Symp. on Computer Arithmetic* (Jun. 1993), 138-145
- [20] M. Ito, N. Takagi, and S. Yajima, "Efficient Initial Approximation Methods for Division and Square Using a Multiply-Add Unit," *IPSJ SIG Notes* 95 HPC-55-10 (Mar. 1995), 73-80 117-124

Chapter 3

- [21] T. Kamei, et al., "A Resume-Standby Application Processor for 3G Cellular Phones," *ISSCC Dig. Tech. Papers*, pp. 336-337, 2004
- [22] M. Ito, et al., "A 390MHz Single-Chip Application and Dual-Mode Baseband Processor in 90nm Triple-Vth CMOS," *IEEE ISSCC Dig. Tech. Papers*, Feb. 2007, pp.274-275

Chapter 4

- [23] M. Ito, et al., "SH-MobileG1: A Single-Chip Application and Dual-mode Baseband Processor", *Proceedings of Hot chips 18*, Aug. 2006

[24] T. Hattori, et al., “A Power Management Scheme Controlling 20 Power Domains for a Single-Chip Mobile Processor,” *IEEE ISSCC Dig. Tech. Papers*, Feb. 2006, pp.542-543

Chapter 5

[25] Y. Kanno et al., “Hierarchical Power Distribution With Power-Tree in Dozens Power Domains for 90-nm Low-Power Multi-CPU SoCs,” *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 74-83, Jan. 2007.

[26] M. Ito, et al., “An 8640 MIPS SoC with Independent Power-Off Control of 8 CPUs and 8 RAMs by An Automatic Parallelizing Compiler”, *International Solid-State Circuits Conference Digest of Technical Papers*, pp.90-91, Feb.2008

[27] Y. Yoshida, et al., “A 4320MIPS Four-Processor Core SMP/AMP with Individually Managed Clock Frequency for Low Power Consumption”, *ISSCC Dig. Tech. Papers*, pp.100-101, 2007.

RELATED PAPERS AND PRESENTATIONS

This thesis is based partially on the following publications.

Papers

- 1) **M. Ito**, K. Nitta, K. Ohno, M. Saigusa, M. Nishida, S. Yoshioka, T. Irita, T. Koike, T. Kamei, T. Komuro, T. Hattori, Y. Arai, Y. Kodama, “A 65nm Single-Chip Application and Dual-Mode Baseband Processor With Partial Clock Activation and IP-MMU”, *IEEE Journal of Solid-State Circuits*, Vol.44, No. 1, pp. 83-89, Jan. 2009
- 2) **M. Ito**, N. Takagi, S. Yajima, “Efficient initial approximation for multiplicative division and square root by a multiplication with operand modification”, *IEEE Transactions on Computers*, Volume 46, Issue 4, pp. 495-498, Apr. 1997
- 3) **M. Ito**, N. Takagi, S. Yajima, “Square rooting by iterative multiply-additions”, *Information Processing Letters* 60, pp. 267-269, 1996

International Conferences

- 1) **M. Ito**, T. Hattori, Y. Yoshida, K. Hayase, T. Hayashi, O. Nishii, Y. Yasu, A. Hasegawa, M. Takada, H. Mizuno, K. Uchiyama, T. Odaka, J. Shirako, M. Mase, K. Kimura, H. Kasahara, “An 8640 MIPS SoC with Independent Power-Off Control of 8 CPUs and 8 RAMs by An Automatic Parallelizing Compiler”, *International Solid-State Circuits Conference Digest of Technical Papers*, pp.90-91, Feb.2008
- 2) **M. Ito**, T. Hattori, T. Irita, K. Tatezawa, F. Tanaka, K. Hirose, S. Yoshioka, K. Ohno, R. Tsuchihashi, M. Sakata, M. Yamamoto, Y. Aral, “A 390MHz Single-Chip Application and Dual-Mode Baseband Processor in 90nm Triple-Vt CMOS”,

International Solid-State Circuits Conference Digest of Technical Papers, pp.274-275, Feb.2007

3) **M. Ito**, T. Irita, E. Yamamoto, K. Nishiyama, T. Koike, Y. Tsuchihashi, H. Asano, H. Yagi, S. Tamaki, K. Tatezawa, T. Hattori, S. Yoshioka, K. Ohno, "SH-MobileG1: A Single-Chip Application and Dual-mode Baseband Processor", *Proceedings of Hot chips 18*, Aug. 2006

4) **M. Ito**, D. Chinnery, K. Keutzer, "Low power multiplication algorithm for switching activity reduction through operand decomposition", *Proceedings of 21st International Conference on Computer Design*, pp.21-26, Oct. 2003

5) **M. Ito**, N. Takagi, S. Yajima, "Efficient initial approximation and fast converging methods for division and square root", *Proceedings of the 12th Symposium on Computer Arithmetic*, pp.2-9, Jul. 1995

Technical Reports

1) **M. Ito**, N. Takagi, S. Yajima, "Efficient Hardware Initial Approximations for Multiplicative Division and Square Root", *Technical report of IEICE COMP95-16*, 1995.

2) **M. Ito**, N. Takagi, S. Yajima, "Efficient Initial Approximation Methods for Division and Square Root Using a Multiply-Add Unit", *Technical report HPC 55-10*, Information Processing Soc. of Japan, Mar.1995

3) **M. Ito**, N. Takagi, S. Yajima, "A High-Speed Square-Rooting Algorithm Using Continued Fractions", *Technical report of IEICE COMP94-35*, 1994.