Dynamic Scaling of Pipeline Depth

Power Analysis of a Floating-point Fused Multiply-Add Unit

Hunpil Kim, Yongsurk Lee
School of Electrical and Electronic Engineering
Yonsei University
Seoul, Korea
Email: yonglee {at} yonsei.ac.kr

Sukhan Lee
SoC Development Team
Samsung Techwin R&D Center
Seongnam-si, Korea

Abstract—***We are developing a multi-core processor with a core unit that can control multiple execution units capable of scaling power and performance by scaling pipeline depth. In this study, we validated the possibility of scaling the power and performance by scaling the pipeline depth. We used Synopsys' SAED_EDK90 libraries and conducted experiments using 100,000 test vectors to assess the proposed floating-point fused multiply-add (FPFMA) unit. The results of synthesis and power simulation showed that area and leakage power increased by approximately 20% and 10%, respectively.***

Keywords*- Power analysis; Scaling power; Scaling performance; Floating point FMA; Scaling pipeline*

I. INTRODUCTION

Recently, embedded processors have adopted a heterogeneous multi-core architecture to satisfy performance and power efficiency requirement. ARM's big.LITTLE architecture [1]; Nvidia's Tegra3, which use a variable symmetric multi-processor [2]; and Samsung's Exynos 5 octa-core processor [3] are examples of multicore architecture. ARM's big.LITTLE architecture consists of low-power Cortex-A7 and high-performance Cortex-A15 processor clusters. The Cortex-A7 cluster is used for power-sensitive operation, while the Cortex-A15 cluster is used for high-performance operation.

The big.LITTLE architecture operates in two modes: task migration mode is used when only one of the two clusters is operating. In other words, a task must be moved when the operating clusters are switched. The multi-processing mode is used for simultaneous operation of both clusters, which should yield high performance and power efficiency. However, the actual implementation of the multi-processing mode has not been reported because of both OS and application limits.

Similar to the big.LITTLE architecture, NVIDIA's Tegra3 consists of four Cortex-A9 processors operating at 1.5 GHz and one Cortex A9 processor operating at 0.5 GHz. The Samsung Exynos 5 has two clusters consisting of four Cortex-A15 and four Cortex-A7 processors. However, Tegra 3 cores do not operate simultaneously, while Exynos 5 supports only the task migration mode.

Although heterogeneous multicore processors have been applied to satisfy performance and power requirements, inefficiencies due to the lack of simultaneous operation remain. To address this issue, research into the composability of homogeneous multicore processors (i.e., symmetric multicore processors) is underway [4]. However, we believe that it is difficult to apply composability to multicore processor architecture because effective implementation depends on both real-time application and significant improvement of available OSs.

Govindan has focused on various aspects of multicore systems [4]. However, we believe that the implementation problem should be addressed in terms of processor core design. Hence, we are developing a processor that consists of many execution units, a core unit, a multi-clock domain, and a multi-voltage domain (multi-VDD). The execution units can be operated in multiple pipelines. The core unit can control the pipeline depth of the execution unit depending on the application or OS mode, such as low-power or high-performance modes. The multi-VDD can support dynamic voltage scaling, and the multi-clock domain can be used to scale the pipeline depth of the execution unit rather than to support dynamic frequency scaling. The overall system configuration can be similar to a general multicore processor; however, it is possible for our processor to scale the power and performance of the execution unit by controlling the pipeline depth.

This paper examines the possibility of the scalable pipeline to scale both power consumption and performance. To verify this possibility, we target a floating-point fused multiply–add (FPFMA) unit because it can be implemented in various pipeline stages and used in different applications such as communication systems, multimedia systems, and high-performance arithmetic operations.

II. TARGET DESIGN, VERIFICATION, AND SYNTHESIS

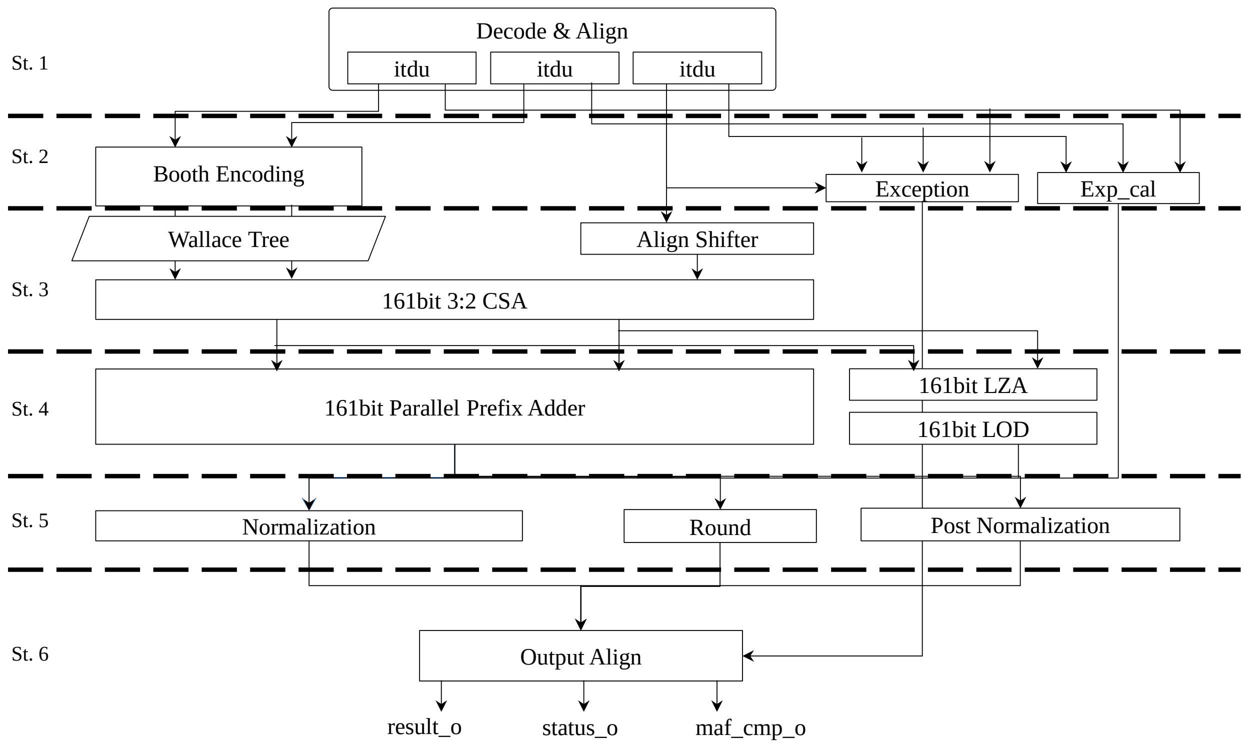

Figure 1. Floating-point FMA block diagram

A floating-point FMA receives three floating-point input data and performs the A × B + C operation. Addition is performed when B is set to 1, and multiplication is performed when C is 0. In general, the most critical problem associated with this operation is that additional rounding and normalization are required for addition after the initial rounding and normalization steps. We employ a 161-bit alignment shifter in our FMA module to solve this problem. Figure 2 shows a block diagram of our floating-point FMA module.

This study aims to evaluate how power consumption changes under the assumption that a user or the OS can scale the pipeline depth depending on the operating environment. Therefore, as shown in Figure 1, we added a select signal (sel_pipe); it can switch on/off the pipeline registers and select the signals that can bypass a pipeline register. This allows the user or OS to scale the pipeline depth of a six-stage floating-point FMA. The user or OS are then capable of using the six-stage pipeline when running applications that require high performance and using the one- or two-stage pipelines for background operations or applications that support low-power operations.

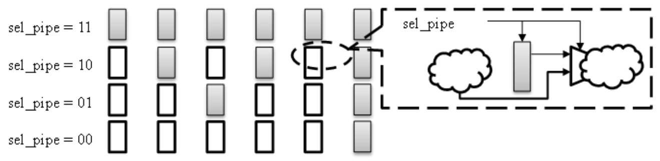

Figure 2. Pipeline control for a scalable 6-stage pipeline

We verified our design using 10,000 input test vectors sampled by dividing the normalization/denormalization and Not a Number (NaN) [5]. We used the DesignWare IP block for comparison with the RTL simulation results.

We synthesized our module using Synopsys' Design Compiler and the 90 nm digital standard cell library (SAED_EDK90) [6]. We conducted syntheses of three operating conditions: a single clock cycle, a non-scalable six-stage pipeline as a control group, and a scalable six-stage pipeline with an additional scaling circuit. In addition, our module was synthesized using three SAED_EDK90 libraries, which are the commonly used fast/fast combinations for power measurement, such as temperature characteristics (−40°C, 25°C, and 125°C) and voltage characteristics (1.32 V).

For accurate comparison and analysis, we used a "grouping" skill. When the "grouping" skill is used, cells (instances) in the design can be grouped into a new sub-design, which operate independently of other grouped participants [7]. Without "grouping," Design Compiler would have to optimize the design after releasing all instantiated cells.

TABLE I shows the synthesis results of the operating conditions using the three libraries. The pipeline structure is larger than the non-pipeline structure by an average of 35%, and the scalable pipeline is approximately 20% larger than the pipeline structure. Because the non-pipeline structure is synthesized without the "grouping" skill, the area is optimized. On the other hand, the scalable pipeline has an overhead of approximately 20% due to the added circuits for pipeline control and the bypass circuit shown in Figure 1.

TABLE I Synthesis results for the 3 structure

Library [Gates / ns]		min	min_nt	min_ht
non-pipeline	Area	42.7k	42.7k	42.9k
	Timing	19.40	29.47	53.72
	Size ratio	0.72	0.70	0.61
6stage pipeline	Area	59.0k	60.7k	70.0k
	Timing	1.92	1.92	2.07
	Size ratio	1	1	1
scalable 6stage pipeline	Area	69.3k	70.9k	88.4k
	Timing	1.94	1.93	2.37
	Size ratio	1.17	1.17	1.26

TABLE II shows the synthesis results for each pipeline stage in the scalable pipeline. The fourth stage is a critical path due to the 161-bit adder, while the second stage has the largest area because the Wallace tree has many signals for both partial products and bypass signals for scalable pipeline stages.

TABLE II Synthesis result for each pipeline stage in scalable pipeline structure

Library [Gates / ns]			min	min_nt	min_ht
Scalable pipeline stage	6	Area	1355	1449	1551
		Timing	1.93	1.90	1.89
	5	Area	3677	3796	4639
		Timing	1.77	1.92	1.84
	4	Area	12275	12300	17316
		Timing	1.93	1.93	2.37
	3	Area	15690	15725	21038
		Timing	1.29	1.88	1.90
	2	Area	32296	33527	39157
		Timing	1.94	1.91	1.85
	1	Area	4022	4073	4719
		Timing	1.68	1.93	1.85

III. Power Analysis

A. Full coverage power analysis

The most accurate power analysis is obtained by testing an actual manufactured chip. However, this requires significant time and effort, and it does not make sense to manufacture a chip solely for the purpose of power analysis. For our purposes, we believe an overview of the data is sufficient because the scope of our study is limited to power consumption changes when scaling the pipeline depth. Thus, we analyzed power using the commonly employed method presented in [8, 9]. In addition, we analyzed three types of known power factors.

$$P_{total} = P_{static} + P_{dynamic}$$

$$P_{static} = V_{dd} \times I_{leakage}$$

$$P_{dynamic} = P_{switching} + P_{internal}$$

$$P_{switching} = \frac{1}{2} \times C_{load} \times V_{dd}^2 \times t$$

$$P_{internal} = V_{dd} \times I_{sc} + \frac{1}{2} \times C_{int} \times V_{dd} \times t$$

Figure 3 shows a full-coverage power analysis, which is the average power consumed by all gates per unit time. It can be seen that the full-coverage power of the nonpipeline structure is higher than the pipeline and the sc_pipeline structures for the overall library. The nonpipeline structure has high power density by performing the same operations on a smaller area for a specific period of time, because synthesis is optimized under the constraints of both maximum area and minimum timing. On the other hand, the full-coverage power of the sc_pipeline structure is smaller than the pipeline structure because the former has lower power density.

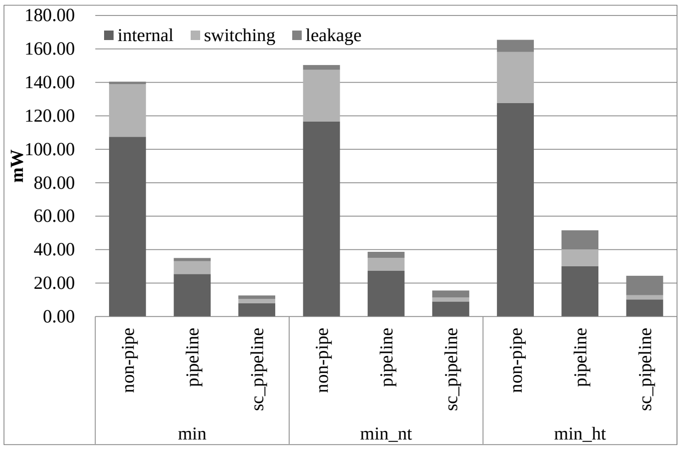

Figure 3. Full-coverage power analysis

B. Average power analysis for scaling pipleline depth

We believe that full-coverage power analysis is insufficient to fully explain the power characteristics of a circuit because there is no application that can drive all circuits simultaneously. Therefore, we performed an additional simulation to analyze the power consumption in actually driven circuits when driving 100,000 test vectors and scaling the pipeline depth. The scaled six-stage pipeline and the pipeline structure were operated at 500 MHz clock frequency, while the scaled three-, two-, and one-stage pipelines were operated at 250, 125, and 83.3 MHz, respectively. Figure 4 shows a graph of the average power analysis for each pipeline stage of the scalable pipeline.

Figure 4 reveals some significant implications. First, the average powers of scalable pipelines are higher than those shown in Figure 3. This is due to an increase of dynamic power, which is due to the input test vectors. Second, the

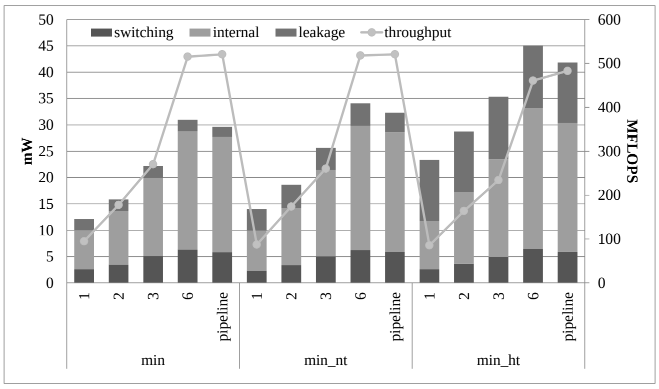

Figure 4. Average power analysis for scaling pipeline depth

leakage power of the “min_ht” library is higher than those of the others due to high temperature characteristics. Third, the power of the scaled six-stage pipeline is approximately 10% higher than that of the pipeline structure.

We believe that the difference of power between the scaled six-stage pipeline and the pipeline structure does not significantly affect the overhead of pipeline scaling. This is because power savings is significantly larger by scaling the pipeline depth when considering the percentage of time for operating a high-performance mode against total time. Fourth, the dynamic power (internal + switching power) is rationally increased in deep pipelines and high-frequency operation. This indicates that deep pipelines are not required to be in low-power mode. Finally, as can be seen in Figure 4, power and performance is increased almost linearly, which indicates that

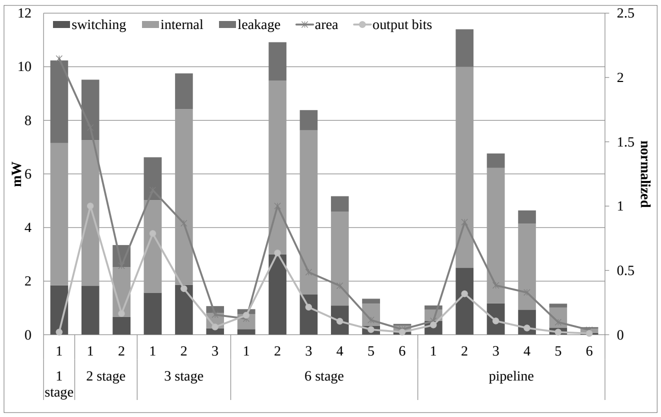

Figure 5. Average power analysis for scaling pipeline depth

they can be scaled by scaling the pipeline depth.

C. *Correlation among the performance, power, area, and ouput bits*

We reported the power dissipated for a circuit added to the scaled pipeline stages and determined the effect on power consumption in each pipeline stage. The added circuit consisted of a pipeline control and bypass signals so that the output bits of the current stage would affect the dynamic power of the next pipeline stage. Figure 5 shows the correlation among the area, the number of output bits, and power analysis details for each pipeline stage from the results of the “min” library shown in Figure 4.

From TABLE II and Figure 5, it can be seen that the second pipeline stage is the largest and consumed the most power in the scalable six-stage pipeline. However, the third pipeline stage would be affected the most by increasing the dynamic power through the output bits of the second pipeline stage from the “3 stage” and “6 stage.” Therefore, we are planning additional research to develop a technique that can scale pipeline stages while optimizing power and area.

IV. CONCLUSION

We proposed a structure that can scale performance and power consumption of a multicore processor and analyzed power consumption by applying it to a floating-point FMA unit. We synthesized the proposed structure with the three fast/fast libraries from Synopsys’ SAED_EDK90 and conducted experiments using 100,000 test vectors. It was observed that the circuit for scaling the pipeline depth can extend the area by approximately 20% and a leakage power by approximately 10%. However, we found that scalable pipelines could scale power consumption and performance scalably. In future, we will conduct additional research into the optimization of the design for scaling pipeline stages that considers power, area, and critical path.

REFERENCES

[1] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin, “Hierarchical power management for asymmetric multi-core in dark silicon era,” in *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pp. 1-9, 2013.

[2] N. Rajovic, A. Rico, J. Vipond, I. Gelado, N. Puzovic, and A. Ramirez, “Experiences with mobile processors for energy efficient HPC,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, pp. 464-468, 2013.

[3] Y. Se-Hyun, L. Seogjun, L. Jae Young, C. Jeonglae, L. Hoi-Jin, C. Dongsik*, et al.*, “A 32nm high-k metal gate application processor with GHz multi-core CPU,” in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC),* pp. 214-216, 2012.

[4] M. Govindan, B. Robatmili, D. Li, B. Maher, A. Smith, S. Keckler, *et al.*, “Scaling Power and Performance via Processor Composability,” *Computers, IEEE Transactions on,* vol. 1, pp. 1-14, 2013.

[5] “IEEE Standard for Floating-Point Arithmetic,” in *IEEE Computer Society*, ed: IEEE, 2008.

[6] “Digital Standard Cell Library,” Synopsys, Ed., 1.11 ed: Synopsys ARMENIA Educational Department, 2011.

[7] Synopsys, “Design Compiler User Guide,” vol. Version F-2011.09-SP2, ed: Synopsys, 2011.

[8] “PrimeTime PX Methodology for Power Analysis,” ed: Syniosys Proprietary, 2006.

[9] K. K. Duane E. Galbi, “Measuring Active Power Using PT PX a User Perspective,” in *SNUG Bostone*, 2010.