Application-Driven Power Efficient ALU Design Methodology for Modern Microprocessors

Na Gong[1], Jinhui Wang[2], Ramalingam Sridhar[1]
[1]University at Buffalo, State University of New York, Buffalo, NY, USA
[2]VLSI and System Lab, Beijing University of Technology, Beijing, China
Email: {nagong,rsridhar}@buffalo.edu

Abstract

In this paper, we propose an application-driven ALU design methodology to achieve high level of power efficiency for modern microprocessors. We introduce a PN selection algorithm (*PNSA*) which enables designers to select power efficient dynamic modules for different applications, based on the detailed analysis of dynamic circuits. Experimental results on ISCAS85 and 74X-Series benchmark circuits show that the power consumption of 8-bit ALU based on this approach can be reduced by 54%-60% for different frequency levels as compared to the conventional dynamic ALU design, demonstrating the effectiveness of the proposed method on application-driven custom ALU design.

Keywords

ALU, application-driven, power, delay, dynamic circuits

1. Introduction

With the exponential growth of semiconductor technology, microprocessors are universally applied in every type of application, such as cloud computing, laptop computers, cell phones, and so on [1]-[3]. The broad application range of microprocessors results in a huge variation of power-performance requirements. As shown in Figure 1, microprocessors in leading servers are performance driven with large power budget; on the ultra-low power areas such as medical sensors, microprocessors require high power efficiency though the performance can be compromised [4][5]. As a fundamental part of microprocessors, arithmetic and logic unit (ALU) performs computing operations and it is typically on the critical path. Therefore, the operating speed of ALU determines the achievable operating frequency of the whole microprocessor. At the same time, ALU is one of the most active components in microprocessors and consumes higher share of power consumption. This situation is further exacerbated for applications with intensive computation, such as digital signal microprocessor, multimedia processor with multiple-core and multiple-ALU [6]. Hence, it is very important to achieve power efficient ALU design while meeting the performance requirement of applications.

In modern microprocessors, dynamic circuits are widely applied in ALU and other time-critical paths due to the superior speed and area characteristics as compared to static CMOS circuits [7]-[9]. For example, in IBM's recent processor - 45nm z196, the ALU design adopts N type domino circuits to minimize the latency [10]. However, since dynamic domino circuits can only perform non-inverting functions, synthesizing dynamic circuits with CAD tools is more difficult than static circuits, and designers usually have to spend a significant amount of time for iterations required to achieve power-performance goals. This adversely impacts the productivity and the quality of the design as well. To make things worse, the varying characteristics of different types of dynamic circuits (N type, P type, and Zipper) increase the design complexity of ALU design. As a result, power efficient ALU design for different applications has become a great challenge in modern processors [11].

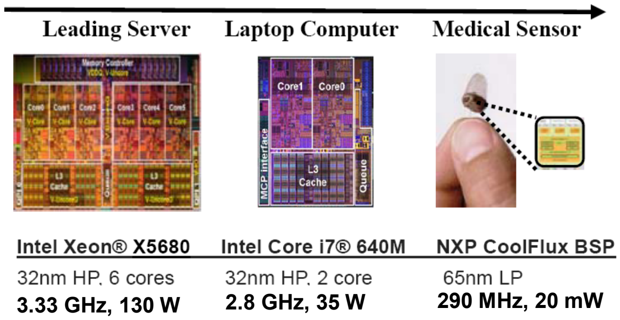

Figure 1 Modern microprocessors in different applications.

Many techniques have been developed to achieve low power ALU design. In [11], a binding algorithm based framework was proposed for low-leakage data paths. Although effective for low power ALU design, the adopted multiple threshold technique may not suitable for high performance applications. A macro-driven ALU design methodology was proposed in [12], which utilizes the design experience to generate the best possible topologies for different macros. However, it only considers the conventional N type dynamic circuits and fails to include other dynamic circuits. In addition, this methodology was implemented in in-house design environment which is not available for other designers. Another methodology for synthesis of data- path was proposed in [13], but it fails to realize equal replacement in the same chip region, thereby inducing large area penalty.

In this paper, we present a new power efficient application-driven ALU design methodology. We proposed a PN selection algorithm (*PNSA*) and therefore the dynamic circuits based ALU modules can be selected effectively to achieve power-performance optimization. Based on final selections, designers can complete the routing work for custom ALU manually or by CAD tools. Compared to existing work, our proposed methodology is different in the following ways: (1) it does not only include conventional N type but also other dynamic circuits and therefore it has greater potential for power reduction; (2) it achieves convenient port-to-port replacement with similar layout area; (3) it can be extended easily to cover a wide range of circuit logic, such as static CMOS, pass-gate, transmission-gate, and tri-state gate.

978-1-4673-4953-6/13/$31.00 ©2013 IEEE

2. Arithmetic and Logic Unit (ALU)

In modern microprocessors, ALU is typically partitioned into function modules and control blocks. A typical 8-bit ALU comprising three modules is shown in Figure 2. It behaves in the following manner: the incoming numbers are added or subtracted in the first module (arithmetic unit), the logic operations are performed in the second module (logic unit), and the bits are shifted in the third module (bit shift unit), respectively. The output selection of the ALU is implemented by a multiplexer (MUX), based on the instruction set called control codes or "concode" [14].

In ALU, the function modules basically perform Boolean (such as AND, OR, XOR) or arithmetic (such as ADD, SHIFT) operations, and performs similar operations on different bits of the same bus, so they are highly regular, consuming considerable power. Such regularity is very helpful in ALU design [12]. Also, the delay time of an ALU is determined by its critical path. In the 8-bit ALU shown in Figure 2, the adder (arithmetic unit) is usually slower than other modules, especially when it is designed as a ripple carry style, and hence the adder is on the critical path of this ALU.

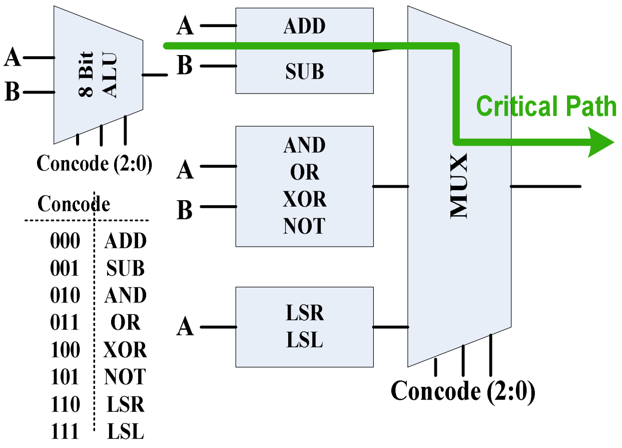

Figure 2 A typical 8 bit ALU Design.

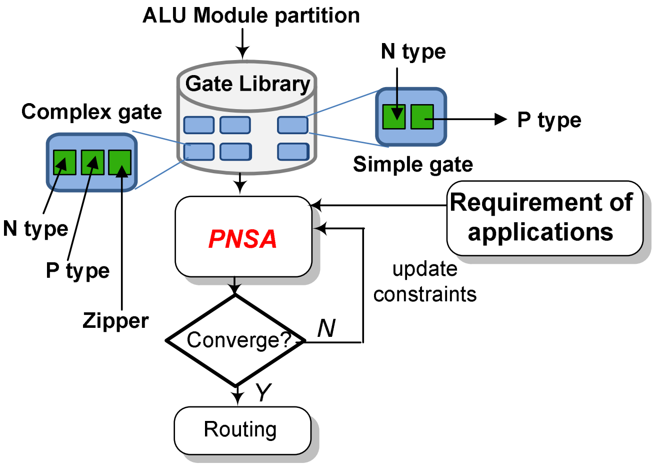

Figure 3 The proposed ALU design methodology.

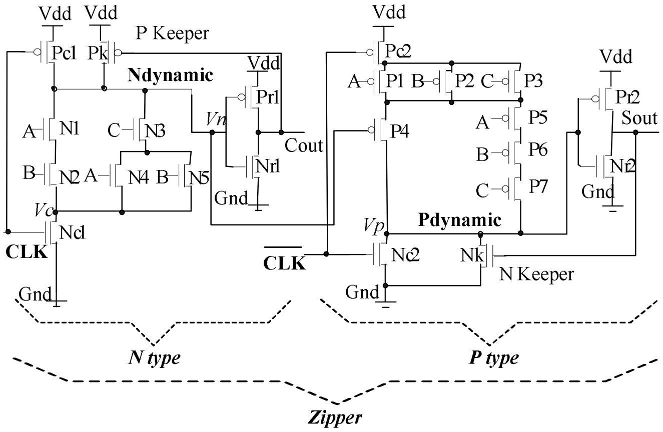

Figure 4 Zipper dynamic full adder

3. Proposed application-driven power efficient ALU Methodology

3.1. Gates library

Since the regular modules of ALU (arithmetic unit, logic unit, and bit shift unit) are designed by the basic gates, we build the gate library as the first step of ALU implementation, as shown in Figure 3. For dynamic circuits, there are three categories: N type, P type, and Zipper dynamic circuits. A detailed analysis of these dynamic circuits is required to implement a power efficient ALU design.

The conventional N type dynamic circuits adopt high speed NMOS transistors to realize high performance. Alternatively, P type dynamic circuits use slow PMOS transistors in the evaluation path and therefore their speed is lower, but their power efficiency is higher than conventional N type circuits. This is because, under inversion bias, the gate leakage current of PMOS transistors is an order of magnitude lower than that of NMOS transistors due to the higher barrier height for the hole tunneling [15]-[17]. The Zipper dynamic circuits were proposed to achieve power-delay tradeoff between N type and P type dynamic circuits [18], [19].

Figure 4 shows a Zipper dynamic full adder, which consists of both N type and P type dynamic circuits. The operating principle of these dynamic circuits is as follows. In the precharge phase of N type dynamic circuits, the clock (CLK) is set low and the dynamic node ($N_{dynamic}$) is charged to V_{dd}. When CLK is set high, the evaluation phase begins and Pc1 is cut off. Provided that the necessary input combination is applied, $N_{dynamic}$ is discharged to ground. Otherwise, the high state of $N_{dynamic}$ will be preserved until the following precharge phase. Different from N type, the P type dynamic circuits enter the predischarge phase when the clock signal is high. After the clock becomes low, $P_{dynamic}$ is charged to V_{dd} depending on the input combination. Otherwise, the low state of $P_{dynamic}$ will be kept until the next predischarge phase [19]-[21]. If we connect the output of N type to the input of P type dynamic circuits, the Zipper dynamic circuits are formed. Accordingly, Zipper dynamic circuits realize a trade-off between power efficiency and speed of N type and P type circuits.

Based on three dynamic circuits, we design the basic gate library to implement a typical ALU shown in Figure 2. The gate library includes multiple simple gates: AND, OR, XOR, SHIFT, INVERTER, and one complex gate: Full ADDER. As shown in Figure 3, the simple gates are designed with N type and P type dynamic circuits and the ADDER is designed based on N type, P type, and Zipper dynamic circuits. Note that, in order to conveniently exchange gates and modules, different designs of each gate have the similar layout area.

In the gate library, the delay and power of gates strongly depend on the manufacturing technology and supply voltage. With a specific technology and supply voltage, the delay (*d*) and power (*e*) relationship of different designs of each gate can be expressed as:

$$\begin{cases} d(Ntype) < d(Zipper) < d(Ptype) \\ e(Ptype) < e(Zipper) < e(Ntype) \end{cases} \qquad (1)$$

3.2. PN selection algorithm (*PNSA*)

An important concern to implement a power efficient ALU is the selection of appropriate gates from the library. Accordingly, we introduce a PN selection algorithm (*PNSA*), based on multidimensional multiple-choice 0-1 Knapsack problem (*MMKP*) [11].

As we discussed in Section 2, the delay time of critical path determines the performance of an ALU. Therefore, we need to select gates in critical path to meet the performance requirement; the gates in uncritical paths adopt low power design to enhance the power efficiency.

The proposed *PNSA* is as follows. We assume there are *n* gates in critical path of an ALU. In the gate library, each gate has two or three types of designs depending on its feature (simple gate or complex gate). The delay and the power of each type of design can be expressed as $d_{ij}(P, V)$ and e_{ij} *(P, V) (i=1, 2,..., n, j=1, 2, or 1, 2, 3),* respectively. Obviously $d_{ij}(P, V) > 0$ *and* e_{ij} *(P, V)* ≥ 0. We consider an application with the delay constraint *D (D>0)*. Now we need to find such *n* gates (each gate is selected from two (or three) designs), so we need to find a 0/1 matrix $|x_{ij}|$, x_{ij} □ {0,1} to make $\sum\sum d_{ij}(P, V)\, x_{ij} \leq D$ and the minimum value of $\sum\sum e_{ij}(P, V)x_{ij}$. Here *P* and *V* represent manufacturing technology and supply voltage, respectively, which influence delay and power directly. Accordingly, *PNSA* can be formulated as a multiple-choice 0-1 Knapsack problem:

$$min \sum_{i=1}^{n}\sum_{j=1}^{3} e_{ij}(P,V)\,x_{ij}$$

$$Delay \begin{cases} \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{3} d_{ij}(P,V)x_{ij} \leq D \\ x_{ij} \in \{0,1\}, i \in [1,n], j \in [1,3] \end{cases} \qquad (2)$$

Similar to *MMKP*, *PNSA* can be solved using dynamic programming approach [22], branch and bound approach [23], or the recent GPU-based approach [24].

TABLE I: BENCHMARK CIRCUITS IN OUR EXPERIMENT

Circuit	Function	No. of Gate	No. of Input	No. of Output
C880	**8 bit ALU**	**383**	**60**	**26**
C2670	**12 bit ALU**	**1193**	**233**	**140**
C3540	**9 bit ALU**	**1669**	**50**	**22**
74181	**4 bit ALU**	**61**	**14**	**8**

TABLE II: SELECTED DESIGN OF MODULES IN CRITICAL PATH OF 8-BIT ALU FOR DIFFERENT APPLICATIONS

Frequency	Gates in Critical Path of 8-bit ALU
100 MHz	**Full adder: 8 P type**
	MUX: P type
200 MHz	**Full adder: 8 P type**
	MUX: P type
300 MHz	**Full Adder: 2 P type, 4 Zipper, 2 N type**
	MUX: N type
500 MHz	**Full Adder: 1 P type, 7 N type**
	MUX: N type

PNSA can be extended to include other design considerations, such as circuit topologies, sizing, process variation and voltage variation.

4. Experimental results

We applied the proposed *PNSA*-based application-driven ALU methodology to a set of benchmarks taken from the ISCAS85 and 74X-Series suites, as listed in Table I. All designs of gate library are implemented based on 0.35 µm GLOBAL FOUNDRIES technology with 3.3 V supply voltage, which characterizes the delay $d_{ij}(P, V)$ and power e_{ij} *(P, V)*) of each design. The temperature (*T*) is 25 °C. Also, we implemented all the adders in ripple carry style and thus the critical path goes through the arithmetic unit of ALU. In our experiment, we consider different applications with performance requirements ranging from 100MHz to 1GHz. Accordingly, ALU designs with different bits are obtained base on the proposed methodology.

Figure 5 shows the normalized power consumption of ALU designs based on the proposed methodology. As shown, for ALU designs with different data-path width, 4-bit ALU has the shortest critical path and therefore its operation frequency can reach 1 GHz. Alternatively, 8-bit and 9-bit ALU cannot run at 800 MHz, and the maximum work frequency of 12-bit ALU is lower than 500 MHz. As also shown in Figure 5, generally the power consumption of ALU designs grows gradually with the increasing performance requirement of applications. An interesting observation is that for 4-bit ALU, the designs for 100-300 MHz result in the same power consumption. This is because the gates using P type dynamic circuits achieve highest power efficiency while satisfying 300 MHz performance requirement. Therefore, the ALU designs for 100-300 MHz applications all adopt P type dynamic circuits. Similarly, for 8-bit and 9-bit ALU, the designs have the same power efficiency when work frequencies are 100 MHz and 200 MHz. In addition, we can see from Figure 5 that the power increase with work frequency of 4-bit ALU is larger than other ALU designs. This is due to the fact that a smaller number of gates are used in 4-bit ALU, and the power consumption of the gates in its critical path accounts for higher ratio in total power consumption.

In addition, we compare the power efficiency of the proposed methodology to the conventional ALU design. Using 8-bit ALU as an example, based on *PNSA*, the selected designs of different modules in critical path targeting different frequency levels are listed in Table II. We can see that with the increasing of performance requirement of applications, more N type dynamic circuits are adopted in critical path to achieve the required frequency level. Figure 6 shows the power comparison of 8-bit ALU designs based on the proposed methodology and the conventional N type ALU for different applications. It is shown that, power savings higher than 54% can be achieved by the ALU based on the proposed methodology, demonstrating the effectiveness as well as the practical usability of the proposed methodology.

Finally, we implement the custom layout design of 8-bit ALU for 500 MHz applications, as shown in Figure 7. We can see that, the ALU design based on the proposed methodology (ALU_proposed) have similar layout area to the conventional ALU (ALU_conv). Also, the two ALU designs have the same positions for input/output pins, thereby achieving convenient port-to-port exchange in a given microprocessor or an IP without any adaptability work.

5. Conclusion

In this paper, we have presented a new, application-driven power efficient ALU methodology for modern microprocessors. Based on the proposed PN selection algorithm (*PNSA*), our methodology allows a trade-off between power efficiency and operation frequency and therefore it helps designers to achieve maximum power efficiency under specific timing constraints of different applications. The proposed methodology has been validated on selected ISCAS85 and 74X-Series benchmark circuits based on 0.35 μm GLOBAL FOUNDRIES technology. The experiment results show that the proposed methodology exploits high power efficiency for different applications and therefore it is suitable for application-driven custom ALU design. More importantly, the methodology presented in this paper can be extended to cover static CMOS, pass-gate, transmission-gate, and tri-state gate, and other circuit logic.

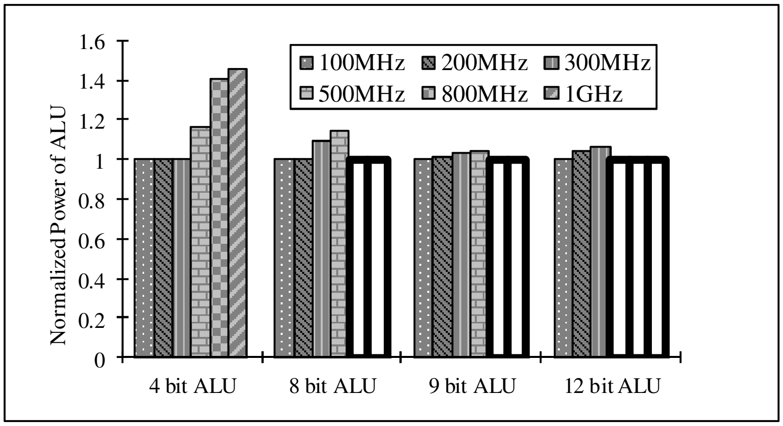

Figure 5 Normalized power consumption of ALU designs base on the proposed methodology.

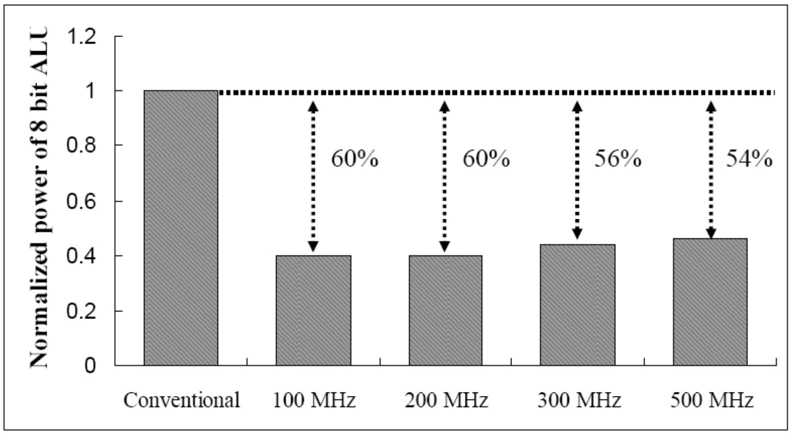

Figure 6 Power savings of the proposed 8-bit ALU as compared to conventional ALU for different frequency levels.

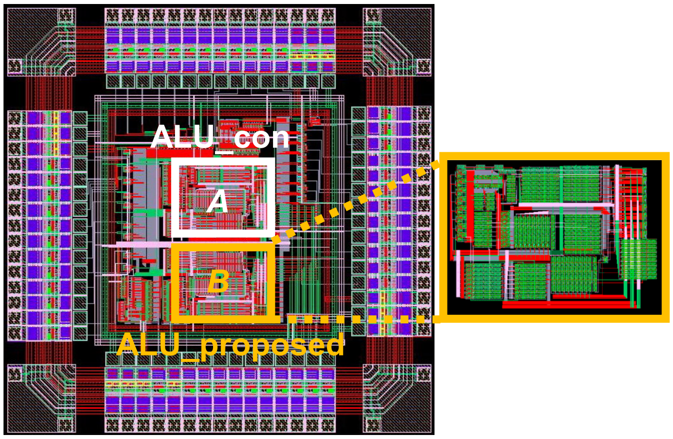

Figure 7 Layout Design of 8-bit conventional ALU (ALU_conv) and new ALU design (ALU_proposed) for 500 MHz applications.

6. Acknowledgment

Jinhui Wang's work was supported in part by the National Natural Science Foundation of China (No.61204040), Beijing Municipal Natural Science Foundation (No.4123092), Ph.D. Programs Foundation of Ministry of Education of China (No.20121103120018).

7. References

[1] R. Riedlinger *et al.* "A 32 nm, 3.1 Billion Transistors, 12 Wide Issue Itanium® Microprocessor for Mission-Critical Servers." IEEE Journal of Solid-State Circuits, IEEE, 2012, 47(1), pp. 177-193.

[2] J. L.Shin *et al.* "A 40 nm 16-Core 128-Thread SPARC SoC Microprocessor." IEEE Journal of Solid-State Circuits, 2011, 46(1), pp. 131-144.

[3] D. F. Wendel *et al.* "POWER7™, a Highly Parallel, Scalable Multi-Core High End Server Microprocessor." IEEE Journal of Solid-State Circuits, IEEE, 2011, 46(1), pp. 145-161.

[4] http://www.intel.com/

[5] http://www.nxp.com/

[6] G. Burda, Y. Kolla, J. Dieffenderfer, and F. Hamdan "A 45nm CMOS 13-Port 64-Word 41b Fully Associative Content-Addressable Register File." 2010 IEEE International Solid-state Circuits Conference (ISSCC2010), pp. 286-287.

[7] H. McIntyre *et al.* "Design of the Two-Core x86-64 AMD "Bulldozer" Module in 32 nm SOI CMOS." IEEE Journal of Solid-State Circuits, 2012, 47 (1), pp. 164-176.

[8] M. Golden, S. Arekapudi, and J. Vinh. "40-Entry Unified Out-of-Order Scheduler and Integer Execution Unit for the AMD Bulldozer x86-64 Core." 2011 IEEE International Solid-state Circuits Conference (ISSCC2011), pp. 80-82.

[9] W. Hu *et al.* "Godson-3B: A 1GHz 40W 8-Core 128GFLOPS Microprocessor in 65nm CMOS." 2011 IEEE International Solid-state Circuits Conference (ISSCC2011), pp. 76-78.

[10] http://www.ibm.com/

[11] C. Gopalakrishnan and S. Katkoori. "Knapbind: an area efficient binding algorithm for low-leakage data paths." 21st International Conference on Computer Design, 2003, pp. 13-15.

[12] M. Nemani and V. Tiwari. "Macro-Driven Circuit Design Methodology for High-Performance Data paths." Design Automation Conference 2003, pp. 661-666.

[13] A. Chowdhary and R. K. Gupta. "A Methodology for Synthesis of Data Path Circuits." IEEE Design & Test of Computers, pp. 90-100.

[14] K. Myny *et al.* "An 8-Bit, 40-Instructions-Per-Second Organic Microprocessor on Plastic Foil." IEEE Journal of Solid-State Circuits, IEEE, 2012, 47(1), pp. 284-291.

[15] G. Yang, Z. Wang, and S. M. Kang. "Gate leakage tolerant circuits in deep sub-100 nm CMOS technologies." Smart Materials and Structures, 2006, 15(1), pp. 21-28.

[16] Y. C. Yeo *et al.* "Direct tunneling gate leakage current in transistors with ultrathin silicon nitride gate dielectric." IEEE Electron Device Letter, 2000, 21, pp. 540-542.

[17] A. Alvandpour, R. K. Krishnamurthy, K. Soumyanath, and S. Y. Borkar "A sub-130-nm conditional keeper technique." IEEE Journal Solid-State Circuits, 2002, 37(5), pp. 633-638.

[18] J. Wang *et al.* "Low Power and High Performance Dynamic CMOS XOR/XNOR Gate Design." Microelectronic Engineering, 2011, 88(8), pp. 2781-2784.

[19] J. Wang *et al.* "Using Charge Self-compensation Domino Full-adder with Multiple Supply and Dual Threshold Voltage in 45nm." 10th International Conference on Ulimate Integration of Silicon (ULIS2009), pp. 225-228.

[20] N. Gong, J. Wang, S. Jiang, R. Sridhar. "Clock-biased local bit line for high performance register files." Electronics Letters, 2012, 48(18), pp. 1104-1105.

[21] N. Gong, B. Guo, J. Lou, and J. Wang. "Analysis and Optimization of Leakage Current Characteristics in Sub-65nm Dual Vt Footed Domino Circuits." Microelectronics Journal, 2008, 39(9), pp. 1149-1155.

[22] H. Kellerer, U. Pferschy, and D. Pisinger. "Knapsack problems". Springer, 2004.

[23] M. E. Dyer and N. KayalJ. Walker. "A Branch and Bound Algorithm for Solving the Multiple-Choice Knapsack Problem." Journal of Computational and Applied Mathematics. 1984, 11(2), pp. 231-249.

[24] B. Suri, U. D. Bordoloi, and P. Eles. "A scalable GPU-based approach to accelerate the multiple-choice knapsack problem." DATE 2012, pp. 1126-1129.