PA-ZSA (Power-Aware Zero-Slack Algorithm): A Graph-Based Timing Analysis for Ultra-Low Power CMOS VLSI

Kyu-won Choi and Abhijit Chatterjee

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
{kwchoi,chat}@ece.gatech.edu

Abstract. This paper describes a slack budget distribution algorithm for ultra-low power CMOS logic circuits in a VLSI design environment. We introduce *Power-Aware Zero-Slack Algorithm* (**PA-ZSA**), which distributes the surplus time slacks into the most power-hungry modules. The **PA-ZSA** ensures that the total slack budget is near-maximal and the total power is minimal as a power-aware version of the well-known *zero-slack algorithm* (**ZSA**). Based on these time slacks, we have conducted the low-power optimization at gate level by using technology scaling technique. The experimental results show that our strategy reduces average 36% of the total (static and dynamic) power over the conventional slack budget distribution algorithms.

1 Introduction

Traditionally, power dissipation of VLSI chips is a neglected subject. In the past, the device density and operating frequency were low enough that it was not a constraining factor in the chips. As the scale of integration improves, more transistors, faster and smaller than their predecessors, are being packed into a chip. This leads to the steady growth of the operating frequency and processing capacity per chip, resulting in increased power dissipation. For state-of-the art systems, the trade-off solutions between the conflicting design criteria (i.e., delay, area, and power) should be considered [1,2]. In general, low-power optimizations without compromising performance are dependent on the time slack calculation and the surplus slack (slack budget) distribution. The time slack means the difference between the signal required time and the signal arrival time at the primary output of each module. The first use of the slack distribution approach is the popular *zero-slack algorithm* (**ZSA**) [3]. The **ZSA** is a greedy algorithm that assigns slack budgets to nets on long paths for VLSI layout design. It ensures that after the assignment, the net slack budget is maximal, which means that no more slack budget could be assigned to any of the nets without violating the path constraints. Most other slack-distribution algorithms are pruning versions of **ZSA** [4,5,6]. The **ZSA** and its off-springs are only for improving delay performance in layout design hierarchy. In this paper, we propose a low-power version of **ZSA**. The proposed **PA-**

ZSA shows that our slack budget distribution strategy ensures that the power consumption is minimal without delay performance degradation for CMOS logic circuits.

2 Background

A CMOS random logic network can be represented as a directed acyclic graph ***G=(V,E)*** where each node $v \in \boldsymbol{V}$ represents a logic gate, and a directed edge $e_{ij} \in \boldsymbol{E}$ exists if the output of gate i is an input of gate j. A primary input (PI) node has no fan-in (f_i) edges and a primary output (PO) has no fan-out (f_o) edges. A combinational circuit operates properly only within the satisfied functional and timing specification. The timing specification is given as arrival time at each primary input and required time at each primary output. The arrival time of a node is the latest time of signals to arrive at the output of any node. The arrival time, ***arr(v)***, and the required time, ***req(v)***, of a node v are recursively computed as:

$$arr(v) = \max_{i \in (1, f_i(v))} \{arr(u) + t_{u,i} + t_v\} \quad \textbf{(1)}$$

$$req(u) = \min_{j \in (1, f_o(u))} \{req(v) - t_{v,j} - t_u\} \quad (2)$$

And the slack at node ***v***, ***slack(v)***, is the difference between the require time and the arrival time at ***v***.

$$slack(v) = req(u) - arr(v) \quad (3)$$

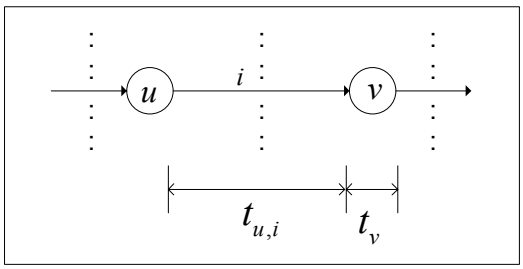

Fig. 1. Arrival Time, Required Time, and Time Slack

If an increased delay of node ***v*** is influence to the slack calculation of node ***u*** and an increased delay of node ***u*** is influence to the slack calculation of node ***v***, we define that ***v*** is slack-sensitive to ***u*** and ***u*** is also slack-sensitive to ***v***.

As an aid to making timing characteristics and their easier interpretation, we introduce a use of special symbols for distributing surplus slacks. The use of these symbols is shown in Figure 2. The time duration values for each node are placed in the upper quadrant of the node. The latest arrival time, ***arr(i)***, is placed in the left hand quadrant

of node ***i*** and the earliest require time, ***req(i)***, is placed in the right hand quadrant of node ***i***. The time slack for each node are placed in the lower quadrant of the node. The slack sensitive path is identified on its edges. Fig. 2(a) shows that an example topological structure of a random logic circuit and the delay of each node. Fig. 2(b) illustrates the result of the timing analysis. In this case, there are only three slack sensitive paths to be considered, for example, G-H, I-K-J, and D-E-F.

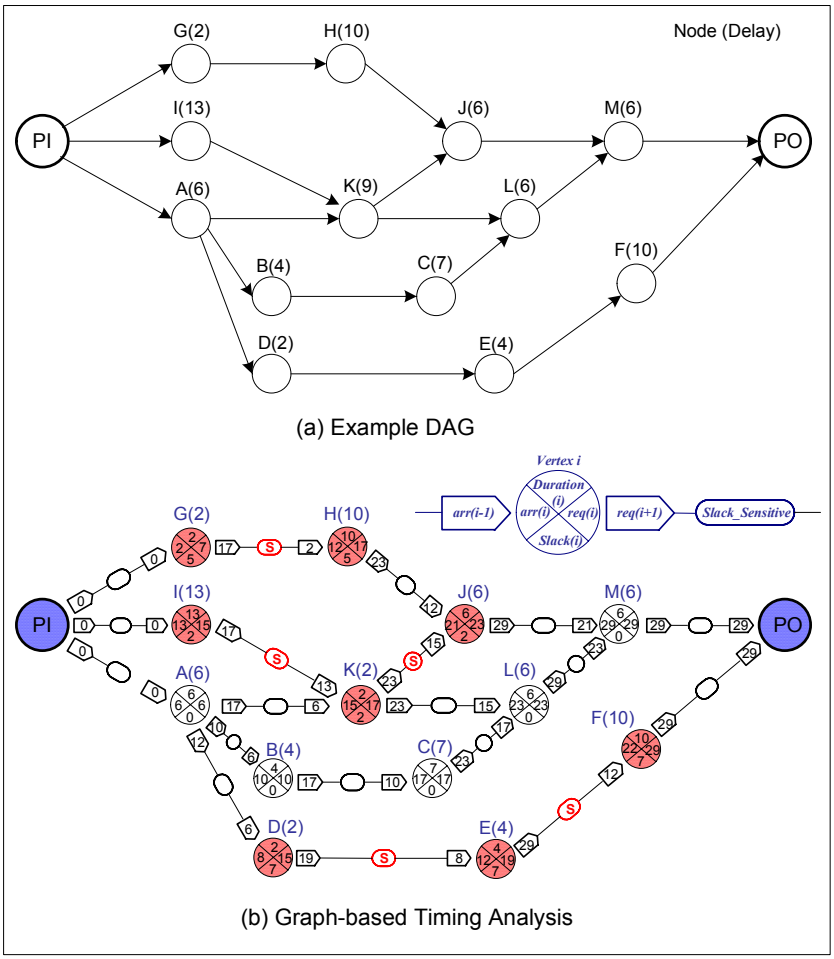

Fig. 2. Graph-Based Timing Analysis

After obtaining the timing characteristics and slack sensitive paths as shown in Fig. 2(b), the problem is how to distribute of the slack for the target performance metrics. Our goal to distribute the surplus time slacks is to make power consumption minimal without compromising the total delay and to make all slacks of the nodes zero. In the

following section, traditional slack distribution algorithms and the proposed algorithm are explained.

3 Power-Aware Zero Slack Algorithm (PA-ZSA)

Before describing our algorithm, **ZSA** (**Z**ero **S**lack **A**lgorithm) [3] and **MISA** (**M**aximum **I**ndependence **S**et **A**lgorithm) [6] are explained briefly.

ZSA: First, ZSA is to start with nodes with minimum slack and locally perform slack assignment such that the slack over the nodes get reduced to zero. This process iterated until the slack of all nodes is zero. More specifically, at each iteration, a node having the least positive slack (s_{min}) is selected. A path on which all nodes have minimum slack (s_{min}), then assign to each node an incremental delay s_{min}/N_{min}, where N_{min} is the number of nodes on the path. For example, in Fig. 3(b), if the path M1-M2 is first found as the least positive slack, each of the two nodes is assigned an incremental delay of 3/2.

MISA: As opposed to **ZSA,** the **MISA** selects a node having the most positive slack (s_{max}). Then a maximum independence set (**MIS**) of a transitive slack-sensitive graph is extracted. An incremental dealy (S_{max}-$S_{(max-1)}$) is assigned to each node in **MIS**. This iteration is continued until all slack of the nodes are zero. In Fig. 3(b), M2 and M3 is the **MIS**, respectively, and S_{max} is 3.0 and $S_{(max-1)}$ is 0. Therefore, an incremental delay for M2 is 3.0 (S_{max}-$S_{(max-1)}$) and an incremental delay for M2 is 0.

In the example of Fig. 3(a), the incremental delay of each node by **ZSA** is (M1=1.5, M2=1.5, M3=1.5) and the incremental delay of each node by **MISA** is (M1=0, M2=3.0, M3=3.0). Therefore, the **MISA**'s total slack budget (0.0+3.0+3.0=6.0) is greater than ZSA's (1.5+1.5+1.5=4.5) by 1.5. If the fan-out number is large enough, the maximum total budget by **MISA** is as large as twice the maximum total budget by **ZSA** as shown in [6]. This shows that **MISA** provides significant improvement over **ZSA** when there exist a large number of fan-outs. However, these algorithms are guaranteed that the delay performance improvement in placement, especially. If the performance metric is the power not the delay, maximum total slack budget does not guarantee low power. For example, in the Fig. 3(a), node M1 has more switching activity than M2 or M3, so, for low power, M1 must have more slack budget than M1 or M2 because power consumption can be reduced only when more surplus slack goes to more power hungry node which has more switching activity and/or capacitance. Therefore, **MISA** or **ZSA** does not ensure that their slack distribution algorithms are optimal in terms of power.

PA-ZSA: We propose a power-aware version of **ZSA** in this paper. As a pre-procedure, we perform an activity generation for each node. *Monte Carlo simulation* is performed for activity profiling of each module/sub-module as described in [2]. This approach consists of applying randomly generated input patterns at the primary inputs of the circuit and monitoring the switching activity per time interval ***T*** using a simulator. Under the assumption that the switching activity of a circuit module over any period ***T*** has a normal distribution, and for a desired percentage error in the activity estimate and a given confidence level, the number of required simulation vectors is

estimated. The simulation based approach is accurate and capable of handling various device models, different circuit design styles, single and multi-phase clocking methodologies, tristate drives, etc. As shown in Fig. 3(c), after timing analysis, the optimal set of the slack distribution is chosen in proportion to the energy consumption of each node. By the following *lemma3.1*, this algorithm guarantees that the power is minimal.

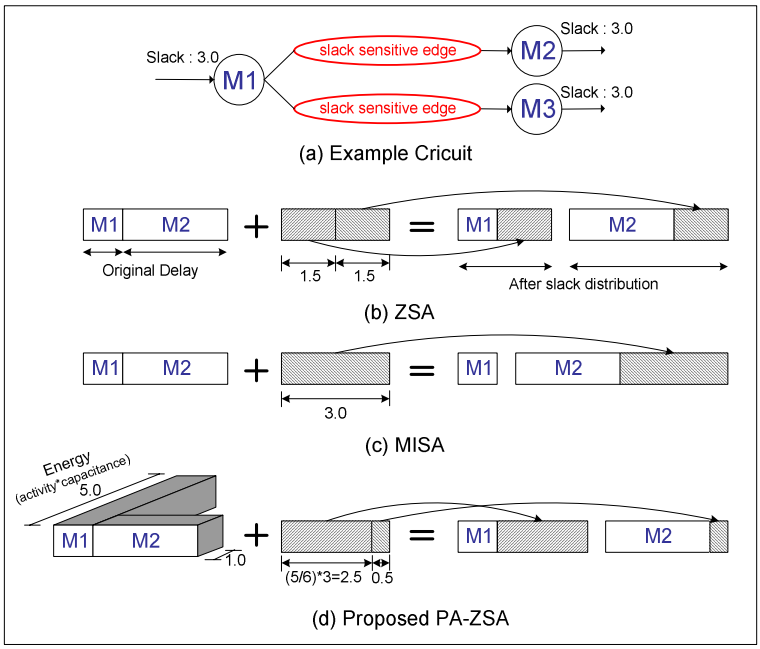

Fig. 3. Slack Distributions Overview

Lemma 3.1: EDR (Energy-Delay Ratio) Paradigm: When the ratios of the energy and the delay for each module are same, the total energy consumption is minimal. In other words, the delay for each module should be proportional to the energy of that module for minimum power consumption. Therefore the surplus time slacks should be assigned onto each module according to the cost function of $\frac{E_1}{d_1} = \frac{E_2}{d_2} = \ldots = \frac{E_n}{d_n}$ (1,…, n: module number) for low power.

Proof: Let's assume the energy consumptions of the module1 (M1) and module2 (M2) in Fig. 4 are $E_1 = K_1 \ V_{dd1}{}^2$ and $E_2 = K_2 \ V_{dd2}{}^2$ respectively. Here, the K_1 and K_2 are proportional to the switching activities and load capacitances of the Modules. The delay of the M1 can be assumed by $d_1 = L_1 \ / \ V_{dd1}$ and the delay of the M2 can be assumed by $d_2 = L_2 \ / \ V_{dd2}$, where the L_1 and L_2 are dependant on the threshold voltage of each module. And the total delay is $T = d_1 + d_2$. We want to minimize the total power P_{Total}:

$$P_{Total} = \frac{K_1 \ V_{dd1}{}^2 + K_2 \ V_{dd2}{}^2}{T}$$

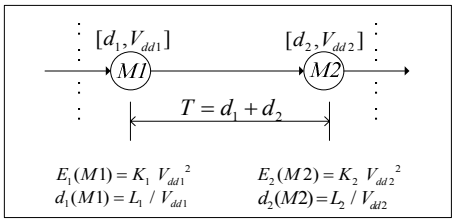

Fig. 4. Two Nodes Example

Now, $d_2 = T - d_1 = (T - \frac{L_1}{V_{dd1}}) = \left(\frac{TV_{dd1} - L_1}{V_{dd1}} \right)$ and after transposing, we have $V_{dd1} = \left(\frac{L_1}{T - d_2} \right)$ and $V_{dd2} = \left(\frac{L_2}{d_2} \right)$. Let's substitute for V_{dd1} and V_{dd2}, then the total power P_{Total} is

$$P_{Total} = \frac{K_1 \left(\frac{L_1}{T - d_2} \right)^2 + K_2 \left(\frac{L_2}{d_2} \right)^2}{T}$$

$$\frac{\partial P_{Total}}{\partial d_2} = \left(\frac{K_1 L_1{}^2}{T} \times (-2) \times \frac{1}{(T - d_2)^3} \times (-1) \right) - \left(\frac{2K_2}{T} \cdot \frac{L_2{}^2}{d_2{}^3} \right)$$

At minimum power, $\frac{\partial P_{Total}}{\partial d_2} = 0$, therefore,

$$\left(\frac{K_1 L_1{}^2}{T} \times (-2) \times \frac{1}{(T - d_2)^3} \times (-1) \right) - \left(\frac{2K_2}{T} \cdot \frac{L_2{}^2}{d_2{}^3} \right) = 0$$

$$\frac{K_1 L_1{}^2}{K_2 L_2{}^2} = \frac{(T - d_2)^3}{d_2{}^3} \text{ or } \frac{(T - d_2)}{d_2} = \left(\frac{K_1 L_1{}^2}{K_2 L_2{}^2} \right)^{1/3}$$

Let

$$\alpha = \left(\frac{K_1 L_1^{\,2}}{K_2 L_2^{\,2}} \right)^{1/3} \tag{4}$$

, then $\frac{T}{d_2} = \alpha + 1$. So, when P_{Total} is minimum,

$$d_2 = \frac{T}{\alpha + 1} \tag{5}$$

$$d_1 = T - d_2 = \frac{\alpha T}{\alpha + 1} \tag{6}$$

At the beginning, we recall $E_1 = K_1 \; V_{dd1}{}^2$ and $E_2 = K_2 \; V_{dd2}{}^2$, and then $\frac{E_1}{d_1} = \frac{K_1 V_{dd1}{}^2}{d_1} = \frac{K_1 L_1^{\,2}}{d_1^{\,3}}$. Substituting for $d_1^{\,3}$ with optimal d_1 from Eqn.(6),

$$\frac{E_1}{d_1} = \frac{K_1 L_1^{\,2}}{\alpha^3 T^3} (\alpha + 1)^3 \tag{7}$$

$\frac{E_2}{d_2} = \frac{K_2 V_{dd2}{}^2}{d_2} = \frac{K_2 L_2^{\,2}}{d_2^{\,3}}$ and Substituting for $d_2^{\,3}$ with optimal d_2 from Eqn.(5),
$\frac{E_2}{d_2} = \frac{K_2 L_2^{\,2}}{T^3} (\alpha + 1)^3$. But from Eqn.(4), we know that $K_2 L_2{}^2 = \frac{K_1 L_1^{\,2}}{\alpha^3}$, so

$$\frac{E_2}{d_2} = \frac{K_1 L_1^{\,2}}{\alpha^3 T^3} (\alpha + 1)^3 \tag{8}$$

The Eqn.(7) and Eqn.(8) are exactly same. Hence, for total minimum power,

$$\frac{E_2}{d_2} = \frac{E_1}{d_1}$$

□

Fig. 5 shows the algorithm of the **PA-ZSA**. At the initialization step, a *Monte Carlo simulation* is conducted for the activity profiling of each module/sub-module as described in [2]. In the phase I, topologic structure is identified by using breath first based search. Then, slack times are calculated by Eqn. (1-3) from the previous section 2. Finally, at the phase III, surplus time slacks are distributed iteratively according to the EDR Paradigm (*lemma 3.1*).

```
Procedure Power-Aware Zero-Slack Algorithm (PA-ZSA)

Input: directed acyclic graph G = (V,E)
Output: power-aware time slack distribution vector TSD(v)
Begin
  Phase 0: Initialization
 Initialize the class variables of all nodes V in G ;
 (delay, activity, early start time, early finish time, late start time,
 late finish time, slack, level=∞)
  Phase I: Labeling {Identify topological order: breath first search}
 Put a start node to FIFO Queue q;
 Initialize the level of the start node = 0;
  While (q Is not 0)
  {
       Obtain a reference to the first element(x) in q;
       Remove node x from q;
         For each fan-out node y of node x
         {
           If level of y = ∞ then
           {
              If number of fan-in node of y =1 then
                 level of y = level of x +1;
              else if number of fan-in node of y >1 then
                 level of y =MAX( levels of fan-in nodes of y) +1;
              Add node y into q;
           }
       }
  }
  Phase II: Timing Analysis and Slack Calculation
  Sort all nodes according to the topological levels;
  For each node of the sorted V
 {
    Assign early start/finish times and late start/finish times;
   Compute time slacks;
  }
   Phase III: Distributing Time Slacks
  Find slack sensitive graph Gsub ∈ G;
  Find maximum activity set Amax from Gsub ;
   Initialize  TSD(Vsub)= 0 ;
  While (Amax  is not empty )
  {
     Assign time slack of Amax by EDR cost function;
     Repeat Phase II
     Update  TSD(V)
  }
   Construct  TSD(V)
End
```

Fig. 5. PA-ZSA Algorithm

4 Experimental Results

Fig. 6 shows overall optimization methodology procedure.
We used an ARM core verilog description for the target system and VCS (synopsys) and design analyzer (synopsys) are used for the functional verification and logic synthesis with 0.25 micron TSMC library. A few arithmetic units (gate-level net lists) are used for the benchmark circuits after the logic synthesis. Then the **PA-ZSA** is performed for delay assignment of each module. After the maximum delays have been assigned to each module/gate in the circuit, we optimize each gate individually for

minimum power. The strategy is to find iteratively, using binary search, the optimal combination of Vdd, Vth, and W for each gate that meets the maximum delay condition while achieving minimum power dissipation [7].

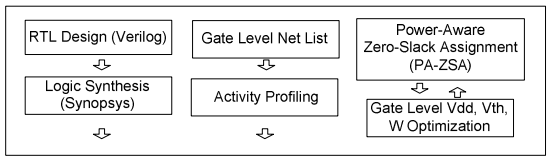

Fig. 6. Optimization Procedure

We developed the optimization simulators and the interface programs with C/C++/STL on Ultra-80 Unix machine. Table 1 demonstrates the impact of the **PA-ZSA** based optimization in terms of power and area. We can obtain over an order of magnitude of total (dynamic and static) power savings. Table 2 shows the effectiveness of the proposed slack budget distribution scheme over traditional algorithms with the same gate level optimization. Approximately 36%-39% more power reduction over conventional slack budget distribution algorithms is achieved.

Table 1. Rsults of PA-ZSA Based Optimization

(a) Before Optimization (Fixed Vdd:3.3v, Vth:0.7v)

System Module	Gates/ Depth	Delay (ns)	Input Activity	ϖ, σ_ω	Power Dissipation			
					Static	Dynamic	Short-ckt	Total
4 - Full Adder	106/48	3.36	0.5	17.9, 24.5	2.09x10E-20	4.37x10E-11	2.15x10E-12	4.59x10E-11
			0.05	17.9, 24.5	2.09x10E-20	4.33x10E-12	2.13x10E-13	4.54x10E-12
16- Full Adder	1030/ 93	6.98	0.5	7.4, 5.6	8.60x10E-20	9.80x10E-11	8.99x10E-11	1.87x10E-10
			0.05	7.4, 5.6	8.60x10E-20	9.56x10E-11	7.51x10E-12	1.03x10E-11
16 - Look ahead (1)	1838/ 81	7.0	0.5	5.9, 6.2	1.48x10E-19	7.65x10E-10	9.33x10E-11	8.58x10E-10
			0.05	5.9, 6.2	1.48x10E-19	1.39x10E-10	9.29x10E-12	1.48x10E-10
16 - Look ahead (2)	1871/ 75	6.3	0.5	7.2, 4.9	1.88x10E-19	8.15x10E-10	9.63x10E-11	9.11x10E-10
			0.05	7.2, 4.9	1.88x10E-19	5.39x10E-10	9.59x10E-12	5.49x10E-10
16 - Look ahead (3)	1928/ 69	5.9	0.5	7.9, 5.3	1.63x10E-19	8.59x10E-10	9.91x10E-11	9.58x10E-10
			0.05	7.9, 5.3	1.63x10E-19	5.50x10E-10	9.90x10E-12	5.59x10E-10

(b) After Optimization (Vdd:0.6-3.3v, Vth:0.1-0.7v)

System Module	Gates/ Depth	Delay (ns)	Input Activity	Vdd, Vth	ϖ, σ_ω	Total Power	Power Reduction	Area Reduction
4 - Full Adder	106/48	3.01	0.5	0.65, 0.1	6.61, 8.41	6.52x10E-13	70.4x	63.2%
			0.05	0.625, 0.1	6.61, 8.41	7.13x10E-14	63.7x	63.2%
16- Full Adder	1030/ 93	7.14	0.5	0.625, 0.1	5.98, 5.03	8.70x10E-13	21.5x	19.1%
			0.05	0.6, 0.1	5.02, 4.03	6.30x10E-13	16.3x	32.1%
16 - Look ahead (1)	1838/ 81	7.0	0.5	0.65, 0.1	5.21, 5.70	1.80x10E-12	47.6x	11.7%
			0.05	0.625, 0.12	3.16, 2.05	3.78x10E-12	39.1x	46.4%
16 - Look ahead (2)	1871/ 75	6.3	0.5	0.625, 0.1	7.1, 6.90	5.98x10E-11	15.2x	1.3%
			0.05	0.625, 0.1	6.16, 6.05	3.08x10E-11	17.8x	14.4%
16 - Look ahead (3)	1928/ 69	5.9	0.5	0.625, 0.1	7.81, 5.01	6.19x10E-11	14.7x	1.1%
			0.05	0.625, 0.1	3.16, 2.05	7.78x10E-11	7.0x	60.0%

Table 2. Comparison with other **ZSA**s

(a) ZSA Based Optimization (Vdd:0.6-3.3v, Vth:0.1-0.7v)

System Module	Delay (ns)	Gates/ Depth	Input Activity	Vdd, Vth	ϖ, σ_ω	Total Power
64 - ALU	20.07	3417/ 226	0.5	0.625, 0.1	5.89, 4.94	1.56x10E-08
			0.05	0.625, 0.1	4.81, 3.88	1.07x10E-09

(b) MISA Based Optimization (Vdd:0.6-3.3v, Vth:0.1-0.7v)

System Module	Delay (ns)	Gates/ Depth	Input Activity	Vdd, Vth	ϖ, σ_ω	Total Power
64 - ALU	20.07	3417/ 226	0.5	0.625, 0.1	6.19, 5.04	1.98x10E-08
			0.05	0.625, 0.1	4.98, 4.74	1.82x10E-09

(c) PA-ZSA Based Optimization (Vdd:0.6-3.3v, Vth:0.1-0.7v)

System Module	Delay (ns)	Gates/ Depth	Input Activity	Vdd, Vth	ϖ, σ_ω	Total Power
64 - ALU	20.07	3417/ 226	0.5	0.625, 0.1	6.09, 9.04	9.91x10E-09
			0.05	0.625, 0.1	4.91, 6.14	6.47x10E-10

*Note: Scheme (c) is better than scheme (a) around 36% - 39% in total power reduction

5 Conclusion

We have presented a new approach to the problem of slack budget distribution for low power. The idea behind the approach has been embodied in an efficient algorithm called the power-aware zero slack algorithm. The algorithm guarantees that the distributed slack budget is optimal in terms of power and the total slack budget is near maximal. Our experiments demonstrate that the proposed slack distribution algorithm achieves results that provide significant improvement over previous approaches in power savings.

References

1. A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 473-484, April 1992.
2. J.M. Rabaey and M. Pedram, *Low Power Design Methodologies*, Kluwer Academic Publishers, 1996, pp 21-64,130-160.
3. R. Nair, C.L. Berman, P.S. hauge, and E.J. Yoffe, "Generation of performance constraints for layout," *IEEE Transactions on Computer-Aided Design*, pp.860-874, Aug. 1989.
4. T. Gao, P.M. Vaidya, and C.L. Liu,"A new performance driven placement algorithm," *Proc. of ICCAD*, pp. 44-47, 1991.
5. H. Youssef and E. Shragowitz, "Timing constraints for correct performance," *Proc. of ICCAD*, pp. 24-27, 1990.
6. C. Chen, X. Yang, and M. Sarrafzadeh, "Potential slack: an effective metric of combinational circuit performance," *Proc. of ICCAD*, pp. 198-201, 2000.
7. P. Pant, V. De, and A. Chatterjee, "Simultaneous power Supply, threshold voltage, and transistor size optimization for low-power operation of CMOS circuits," *IEEE Trans. On VLSI Systems*, vol. 6, no. 4, pp. 538-545, December 1998.