Multiobjective Optimization of Sleep Vector for Zigzag Power-Gated Circuits in Standard Cell Elements

Seungwhun Paik
Dept. of Electrical Engineering, KAIST
Daejeon 305-701, Korea

Youngsoo Shin
Dept. of Electrical Engineering, KAIST
Daejeon 305-701, Korea

ABSTRACT

Zigzag power gating (ZPG) has been proposed to alleviate the drawback of power gating in its long wake-up delay, thereby broadening the application of power gating to suppressing active- as well as standby-leakage. However, complicated power network due to the use of nMOS and pMOS switches in zigzag fashion has limited its application to custom circuits. Heterogeneous use of power rails inevitably incurs overhead of area and wirelength during physical design. Furthermore, the use of sleep vector causes additional switching power when entering standby mode and returning back to active mode. The switching power should be minimized not to outweigh the leakage saving by employing ZPG scheme. In this paper, we propose a complete power network architecture, which allows us to use unmodified standard cell elements for implementing ZPG circuits. We formulate selecting sleep vector as a multiobjective optimization problem, minimizing transition energy and total wirelength. We solve the problem by employing multiobjective genetic-based algorithm. Experimental results show the saving of 39% in transition energy and 8% in wirelength, on average, for several benchmark circuits in 65-*nm* technology. The complete design flow starting from RTL description down to layout is proposed, and assessed with 65-*nm* technology.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Standard cells, VLSI*; B.7.2 [**Integrated Circuits**]: Design Aids—*Layout*

General Terms: Algorithms, Design

Keywords: Zigzag power gating, low power, leakage current, sleep vector, standard-cell

1. INTRODUCTION

Subthreshold leakage current has grown exponentially with every process generation, due to the scaling down of threshold voltage, and is now responsible for a high proportion of total power consumption.

Power gating [1] is one of the most effective techniques to limit subthreshold leakage. It consists of gating, or cutting off, a circuit from its power supply rails during standby mode, so that the leakage path is virtually removed. Either pMOS or nMOS can be used for the gating device, which is called a header if it is pMOS and a footer if it is nMOS. When a footer is turned off, the voltage at the virtual ground (V_{ssv}), where the footer has its drain, slowly goes up until it reaches a steady state, which is usually close to V_{dd}. This implies that all the nets internal to the circuit are charged up to V_{dd}, and many of those nets simultaneously start to discharge once the footer is turned on. This large current is a major cause of long wake-up delay in power gating, thus limiting its application to suppressing leakage when devices are in standby mode.

DAC 2008, June 8–13, 2008, Anaheim, California, USA

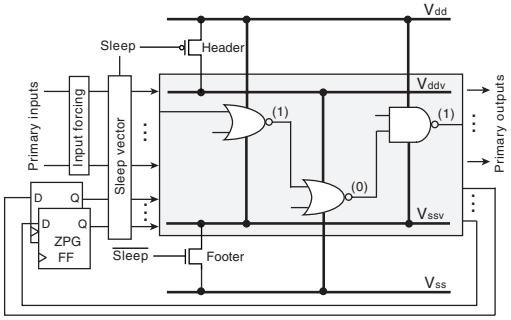

Figure 1: Block diagram of zigzag power-gated circuit.

Many circuits have been proposed to alleviate the large wake-up delay of power gated circuits [2–4]. But, the wake-up delay is still too large [2, 3] or the amount of leakage saving is not large enough [4]. Zigzag power gating (ZPG) [5–7] has been shown to achieve the best balance of leakage saving and wake-up delay. Figure 1 shows the concept of ZPG circuit. Before entering standby mode, a pre-determined input vector, called *sleep vector*, is applied to the circuit. Since the sleep vector is pre-determined, the logic states of all the nets during standby mode are known in advance. We connect the gates with logic high output to a footer, and the remaining gates are connected to a header. Once the footer and header have been turned off, their drain potentials (V_{ssv} and V_{ddv}) slowly go up and down respectively until they reach the steady-state potentials. However, in contrast to simple power gating, all the nets keep their logic states. Thus, there is no need to discharge or charge the nets to restore their logic states when returning to active mode, which is why ZPG is fast.

However, the use of both footer and header in zigzag fashion has the drawback of complicated power networks. The gates connected to a footer need V_{dd} and V_{ssv}, the gates connected to a header need V_{ddv} and V_{ss}, and the flip-flops need all four power rails (as will be explained in the next section). Therefore we cannot use standard cells in conventional power network with V_{dd} and V_{ss} rails.

Another drawback of ZPG scheme is caused by the use of sleep vector. When the sleep vector is applied as a circuit enters standby mode and when it is removed as a circuit returns back to active mode, additional switching power is consumed. This switching power should be minimized not to outweigh the leakage saving achieved by employing ZPG scheme. In addition, the sleep vector determines whether each gate is connected to a footer or to a header.

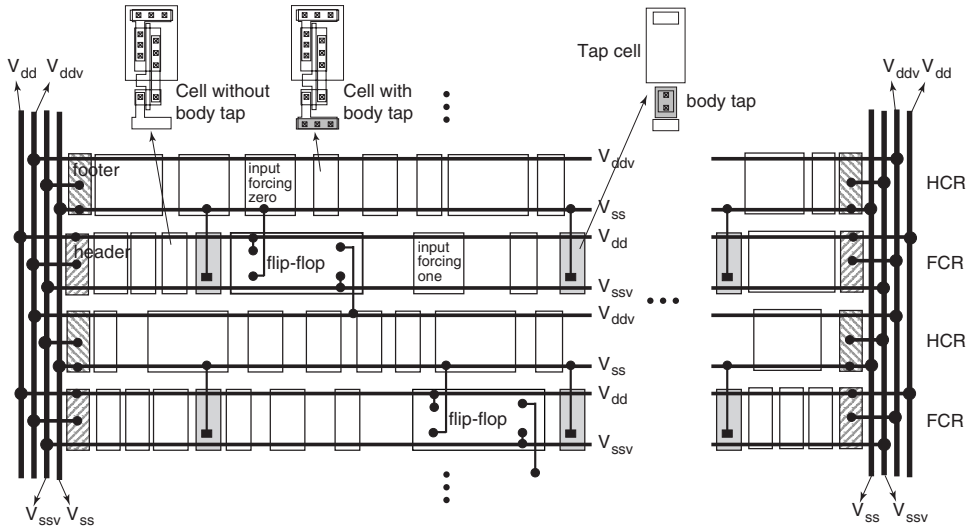

Figure 2: Power network for ZPG circuits based on standard-cell elements.

Since the gates connected to different types of current switches cannot be placed in the same circuit row, careless use of sleep vector can cause large overhead in terms of area and wirelength during physical design stage.

In this paper, we address a question of how to design ZPG circuits using conventional standard cell elements, and a question of how to determine sleep vector such that we have less extra switching power and less overhead of area and wirelength during physical design. Our contribution can be summarized as follows:

- We propose a (complete) power network architecture, which allows us to use unmodified standard cell elements for implementing ZPG circuits (Section 2). The practical implementation of ZPG flip-flops is proposed.
- We formulate selecting sleep vector as a multiobjective optimization problem, minimizing both transition energy and total wirelength. We solve the problem by employing multiobjective genetic-based algorithm (Section 3).
- The complete design flow starting from RTL description down to layout using commercial CAD tools is proposed, and assessed with 65-*nm* commercial technology (Section 4).

2. CELL-BASED DESIGN OF ZPG CIRCUITS

Figure 2 shows a proposed power network architecture for ZPG circuits based on standard-cell elements. There are two types of circuit rows: HCR (header-connected circuit row) and FCR (footer-connected circuit row). The gates connected to a header are placed in HCR as its name implies. Thus, HCR has power rails of V_{ddv} and V_{ss}. Similarly, the gates connected to a footer are placed in FCR which has V_{dd} and V_{ssv} for its power rails. This arrangement allows us to use unmodified standard cells, which is a major advantage of the proposed power network.

The bodies of the conventional standard cells are tied to their GND terminals. Thus, the bodies of HCR gates are biased to V_{ss}, while FCR gates have their bodies tied to V_{ssv}. To avoid the electrical short between V_{ss} and V_{ssv}, the body contacts of FCR cells are removed after placement, implying that the bodies of all the cells are now biased to V_{ss}. To compensate for the removed body contacts, extra cells, called body tap cells, are placed in regular distance in FCRs, as shown in Figure 2.

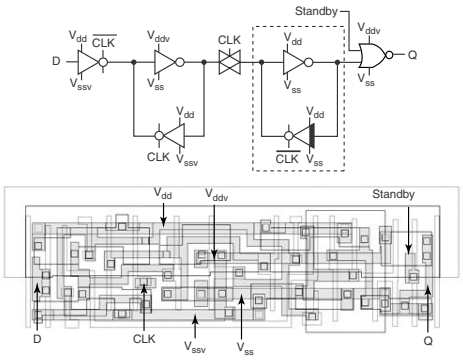

Figure 3: ZPG D flip-flop with logic low for both the corresponding bit value of the sleep vector and D-input.

As shown in Figure 1, a part of a sleep vector should be provided by flip-flops, while the remaining part corresponding to primary inputs is provided by extra circuitry[1]. Figure 3 shows one of ZPG D flip-flops[2], which includes a NOR gate at its output to provide a logic low to the sleep vector. The NAND gate, if used in place of NOR gate, will provide a logic high. Once the sleep vector is applied, all the internal nets including flip-flop input are determined. The flip-flop in Figure 3 can be used when D-input is logic low in standby mode. Note that an inverter and two tri-state inverters in the master latch are connected either to footer or to header. The latch in slave portion of the flip-flop (enclosed in dashed box) should be isolated from the rest of the flip-flop during standby mode to preserve the current state, thus it is directly connected to V_{dd} and V_{ss}. The state is restored once the circuit returns to active mode. Note that the tri-state inverter, which is out of clock-to-Q path, utilizes high V_t to reduce subthreshold leakage.

As opposed to combinational gates, we need all four power rails in the flip-flop, which are provided through signal ports instead of

[1]The box denoted as sleep vector is a conceptual one.

[2]There are four of them, depending on the corresponding bit value of the sleep vector bit it provides, and the D-input in standby mode.

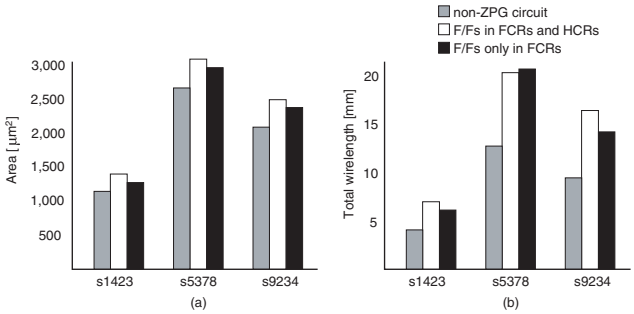

Figure 4: (a) Area and (b) wirelength of three implementation styles: non-ZPG circuits, ZPG-circuits where flip-flops are allowed in FCRs and HCRs, and ZPG-circuits when flip-flops are restricted to FCRs alone. For each placement, we enforce that 86% of placement region is occupied by the cells, which represents a tight placement.

VDD and GND terminals (see Figure 2 and Figure 3). This arrangement allows us to place flip-flops anywhere in the placement region. However, if it is placed in HCR, it incurs the area overhead (about 32% of ZPG flip-flop area) due to n-well isolation. This is because HCR gates have their n-well tied to V_{ddv}; the n-well of the flip-flop should be at V_{dd} because of slave latch, which is responsible for state retention (see Figure 3). This fact has important implication for physical design optimization, as will be discussed in 3.1.

Input forcing circuit [6, 7] is a circuit that is located at each primary input to provide the corresponding bit value of the sleep vector (see Figure 1). The one that provides logic high, a block labeled input forcing one in Figure 2, is placed in FCR and the one that provides logic low, a block labeled input forcing zero in Figure 2, is placed in HCR due to the availability of V_{dd} and V_{ss}, respectively. Footer cells and header cells are placed in HCRs and FCRs, respectively, as shown in Figure 2, as they require access to V_{dd} and V_{ss}. Although current switches can be placed anywhere within corresponding circuit rows, the boundary is preferred to minimize current path from virtual power rails (V_{ddv} or V_{ssv}) to real power rails (V_{dd} or V_{ss}), which helps minimize IR drop on power rails.

3. SLEEP VECTOR OPTIMIZATION

3.1 Choosing a Sleep Vector to Minimize Wirelength

Our ZPG flip-flops can be placed anywhere in the placement region, which is usually preferred in favor of clock design. On the other hand, the flip-flops placed in HCRs incur area overhead due to n-well isolation as we discussed in the previous section. We tested several ISCAS benchmark circuits in commercial 65-*nm* technology. Figure 4(a) compares the area of two implementation styles of each circuit (white and black bars). The increase from non-ZPG to ZPG is inevitable, since there are more circuitry in ZPG implementation, and placer does not have full flexibility in placing cells due to the presence of FCRs and HCRs. When we compare two styles of ZPG implementations, it is readily seen that restricting flip-flops in FCRs is preferred. Note that flip-flops are not totally restricted in their placement, as they can be freely located in 77% of the placement region on average of three circuits. This is because we re-computed the area of the cells that will be placed in FCRs and HCRs, assuming that flip-flops will be placed in FCRs, and then determined the number of FCRs and HCRs we need in each design.

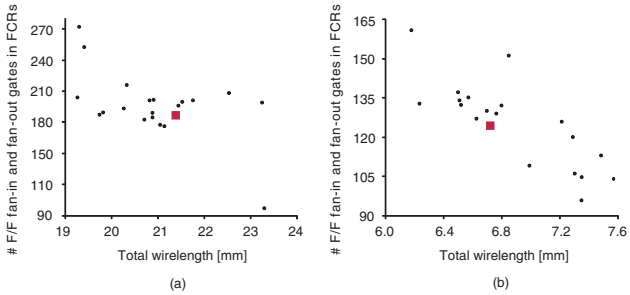

Figure 5: Total wirelength versus the number of flip-flops' fan-in and fan-out gates placed in FCRs: (a) s5378 and (b) s1423.

We also compared the total wirelength (after detailed routing) of two implementation styles, which is shown in Figure 4(b). The wirelength tends to decrease in most benchmarks (including other circuits not shown in Figure 4) when the placement of flip-flops is restricted to FCRs (black bar), which is also preferred in terms of area. This can be understood from the fact that major portion of placement region is now assigned to FCRs, thereby providing more flexibility to placer, which then translates into less wires in the designs.

In order to see the possibility of further reducing wirelength while we keep the flip-flops in FCRs, we took s5378 as an example. The total wirelength of non-ZPG implementation was about 12.6 mm; that of ZPG implementation was about 21.4 mm. In the increase of 8.8 mm, about 16.1% were due to sleep signal (which is routed to footers, headers, flip-flops, and input forcing circuits) and about 6.4% were caused by input forcing circuits, both of which are inevitable. The remaining 77.5% mainly stemmed from restricted placement (thus, hard to categorize), but it turned out that the wires between flip-flops, which are now in FCRs, and the gates in HCRs, which are either their fan-ins or fan-outs, take a large proportion of it. Since the type of circuit row assigned to each gate is determined by sleep vector, this implies that 77.5% of extra wires for ZPG implementation of s5378 can be controlled to some extent. Figure 5(a) shows the result. The initial sleep vector, which was randomly generated, corresponds to the data point denoted as a box. The other data point correspond to different sleep vectors. It is clearly seen that the sleep vectors that lead to more number of flip-flops' fan-in and fan-out gates placed in FCRs tend to yield less wirelength, which is also confirmed in other circuit, s1423, as shown in Figure 5(b). In summary, we place flip-flops in FCRs in favor of area. We want to select sleep vector such that as many fan-in and fan-out gates of flip-flops as possible are placed in FCRs in favor of wirelength.

3.2 Choosing a Sleep Vector to Minimize Transition Energy

When the sleep vector is applied as a ZPG circuit enters standby mode, additional switching power is consumed. Similarly, switching power is consumed again, when the sleep vector is removed as the circuit returns back to active mode. The transition energy caused by these switching power should be minimized not to outweigh the leakage saving achieved by employing ZPG scheme. The minimum idle time that yields energy saving from ZPG scheme, T_{idle}, can be readily expressed by [8]:

$$T_{idle} = \frac{E_{tr\_2\_sb} + E_{tr\_2\_at} - P_{sb\_zpg}(2T_{tr})}{P_{idle} - P_{sb\_zpg}} \approx \frac{2E_{tr}}{P_{idle}}, \qquad (1)$$

where $E_{tr\_2\_sb}$ and $E_{tr\_2\_at}$ are transition energy to enter standby mode and to return back to active mode, respectively. They are assumed to be equal in magnitude, thus are denoted as E_{tr}. The

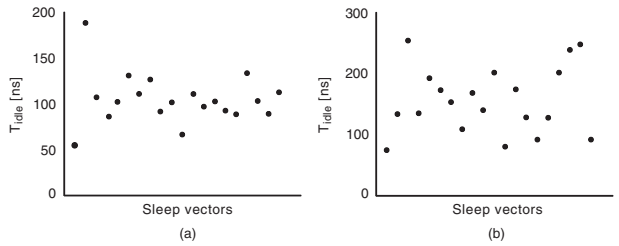

Figure 6: Minimum idle time T_{idle} for various sleep vectors: (a) s1423 and (b) s820.

power consumption of non-ZPG circuit is denoted as P_{idle}; that of ZPG circuit is denoted as P_{sb_zpg}, which is usually negligible compared to P_{idle}. The transition time (from active to standby mode and vice versa) is assumed to be the same, and denoted as T_{tr}. From (1), it is clear that transition energy should be minimized for short T_{idle}, such that ZPG circuits can achieve power saving in as many chances of idle period as possible.

Since the choice of sleep vector may affect transition energy, we took s1423 as an example. We first assumed that the signal probabilities of primary inputs when the circuit is in idle are available, which is usually possible from the knowledge of typical usage cases. The state probabilities of flip-flops, which constitute a part of input vectors for combinational portion of the circuit, can be readily derived [9, 10]. Random patterns, corresponding to primary inputs and flip-flop outputs, were generated following the derived signal probabilities. They were used to obtain average transition energy, E_{tr}, for each sleep vector we tried. The same patterns were used to obtain average idle power, P_{idle}, which then gives us T_{idle}. The experiments were performed for 20 randomly generated sleep vectors, and the result is shown in Figure 6(a). It is clear that there is an opportunity to reduce transition energy, thus T_{idle}, via searching various sleep vectors. Figure 6(b) shows the result for other circuit, s820.

3.3 Multiobjective Genetic Algorithm-Based Search of Sleep Vector

From 3.1 and 3.2, we now have two objectives to achieve via exploring sleep vectors: minimizing wirelength and minimizing transition energy. Since two objectives can have conflicts, this casts itself as a multiobjective optimization problem, which we try to solve via multiobjective genetic algorithm [11].

Figure 7 shows the overall algorithm. The input to the algorithm is a gate-level netlist, with signal probabilities of primary inputs when the circuit is in standby mode, as we discussed in 3.2. We then derive the state probabilities of flip-flops (L1) [9]. The probabilities of the primary inputs and the flip-flop states are then propagated [12] through combinational portion of the netlist to obtain the signal probabilities of all internal nets (L2), which are used to obtain transition energy metric (L8).

We initially generate N random sleep vectors $\mathbf{P}$ (L3), from which another N sleep vectors $\mathbf{Q}$ are generated (L5) through crossover and mutation. For crossover, we randomly select two sleep vectors from $\mathbf{P}$, randomly pick one bit position, cut each vector at the bit position, and exchange each other. The mutation is performed (with some fixed probability) after crossover by flipping one randomly-chosen bit of each of the two vectors. For each of $2N$ sleep vectors in $\mathbf{P}$ and $\mathbf{Q}$, we evaluate the metrics of wirelength and transition energy (L7 and L8). For wirelength metric, we check the fan-in and fan-out gates of flip-flops and count the number of them with output of logic high, which will be placed in FCRs (thus, the larger the wirelength metric, the better). From 3.1 and Figure 5,

Sleep Vector Search Based on Multiobjective Genetic
Input: gate-level netlist with signal probabilities of PIs
Output: sleep vector
begin
L1 Derive state probabilities of F/Fs
L2 Derive signal probabilities of internal nets
L3 Generate N initial parent sleep vectors, $\mathbf{P} = (P_1, P_2, \ldots, P_N)$
L4 **for** $i = 1, 2, \ldots, M$ **do**
L5 Generate N offspring sleep vectors, $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_N)$ from $\mathbf{P}$
L6 $\mathbf{R} = \mathbf{P} \cup \mathbf{Q}$
L7 Evaluate wirelength metric $W(R_j), \forall R_j \in \mathbf{R}$
L8 Evaluate transition energy metric $E(R_j), \forall R_j \in \mathbf{R}$
L9 $\mathbf{F} = (F_1, F_2, \ldots)$ = `non_dominated_sorting`$(\mathbf{R}, W, E)$
L10 **if** $i \neq M$ **then**
L11 $\mathbf{P} = \emptyset$ and $j = 1$
L12 **while** $|\mathbf{P}| + |F_j| \leq N$ **do**
L13 $\mathbf{P} = \mathbf{P} \cup F_j$, $j = j + 1$
L14 **end do**
L15 $\mathbf{P} = \mathbf{P} \cup$ `compute_crowding_distance`(F_j)
L16 **else** return `select_one_Pareto_point`(F_1)
L17 **end if**
L18 **end do**
end

Figure 7: Pseudo code of sleep vector search based on multiobjective genetic algorithm.

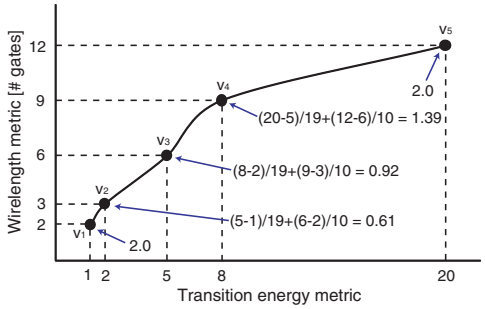

Figure 8: Example of computing crowding distance.

we know that there is a correlation (even though not strong enough for some examples) between the total wirelength and the number of flip-flops' fan-in and fan-out gates placed in FCRs. The transition energy metric is approximated by the sum of switched capacitance of all internal nets:

$$\sum_i C_i \left(l_i + (-1)^{l_i} p_i \right), \tag{2}$$

where C_i denotes the load capacitance consisting of wire capacitance and input capacitance of fan-out gates, p_i is a signal probability obtained from L2, and l_i corresponds to the logic value when the sleep vector is applied. Note that the term in parentheses is evaluated as p_i when $l_i = 0$ and $1 - p_i$ when $l_i = 1$ (recall that p_i is a probability that net i takes logic high).

The $2N$ sleep vectors are classified according to the dominance relation based on the metrics of wirelength and transition energy (L9). The vector R_i is said to be dominated by the vector R_j, if $W(R_i) \leq W(R_j)$ and $E(R_i) \geq E(R_j)$, i.e. if R_i is inferior to R_j in both metrics. The F_1 contains the vectors that are not dominated by any other vectors in $\mathbf{R}$, thus represents Pareto points in current generation. The F_2 contains the vectors that are dominated only by the vectors in F_1. The other classes are defined similarly.

Once the vectors are classified, we select N out of $2N$ vectors (L11 to L15), which then constitute the parent vectors $\mathbf{P}$ in the next

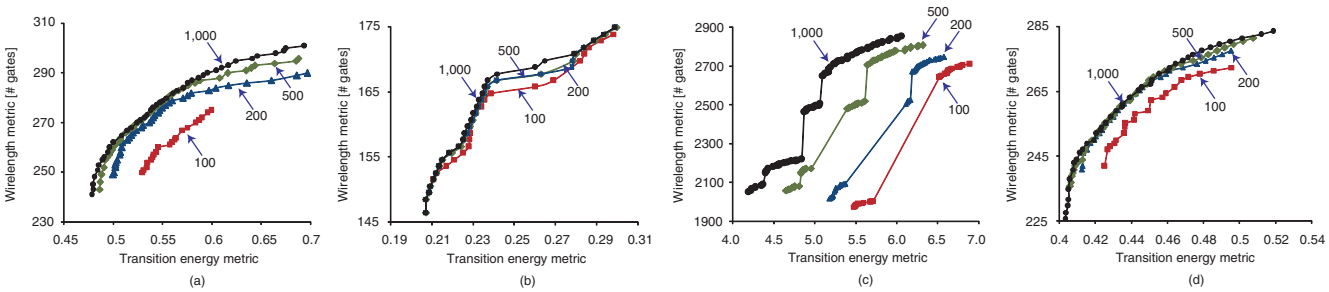

Figure 9: Pareto points, F_1, of four example generations for (a) s5378, (b) s1423, (c) s35932, and (d) s9234

generation (i.e. iteration). This is done by including F_j one by one, starting from F_1 (L12 to L14)). At the end, when we select a subset of vectors from F_j (L15), we try to select them such that they are distributed as evenly as possible in a space spanned by wirelength metric and transition energy metric. This is accomplished by computing *crowding distance* of each vector in F_j. Assume that F_j has n vectors $v_1, v_2, \ldots, v_n$, which are arranged in increasing transition energy metric (they are automatically arranged in increasing wirelength metric as well, due to the definition of F_j). The crowding distance of v_i is defined by:

$$\begin{cases} \frac{E(v_{i+1})-E(v_{i-1})}{E(v_n)-E(v_1)} + \frac{W(v_{i+1})-W(v_{i-1})}{W(v_n)-W(v_1)} & \text{if } 1 < i < n \\ 2 & \text{otherwise} \end{cases}$$

The vectors are selected from the one with the maximum crowding distance. Figure 8 shows an example. If we select three, v_1, v_4, and v_5 will be selected.

Once we reach the end of iterations (L16), we need to select one sleep vector from the Pareto points, F_1. There can be various schemes for this purpose, but currently we sort the points in F_1 in one metric and select the one in median. Note that this also guarantees the median in the other metric, since the points in F_1 are not dominated by any others. Figure 9 shows the Pareto points of four example generations of example circuits, and how they are gradually improved from generation to generation.

4. EXPERIMENTS

4.1 Design Flow

The design flow for ZPG circuits is shown in Figure 10. The register transfer level (RTL) design goes through a standard logic synthesis to create the initial gate-level netlist. From the netlist, we first determine the sleep vector using the method in the previous section. Then, it determines the type of input forcing circuit required at each primary input, as well as the type of ZPG flip-flop (see Figure 3) to be substituted for each flip-flop in the original netlist.

To determine the size of the current switches (i.e. footer and header), which affects the active-mode circuit delay, we first decide on the voltage drop which we will allow at the switches when they are turned on during active mode. We can also determine the average current through the gates connected to the footers and headers by applying random logic patterns to the inputs of a circuit simulation of the netlist. Using this estimate of the average current, and the chosen voltage drop, allows us to size the current switches [13], which in turn determines the number of cells that will be required.

The netlist is then re-synthesized with V_{dd} set to the voltage swing that each gate will experience, which is V_{dd} minus the chosen voltage drop across a footer or a header. If the timing constraint cannot be satisfied by this re-synthesis, we reduce the voltage drop across the current switches, even though this will increase the switch size, and repeat the process.

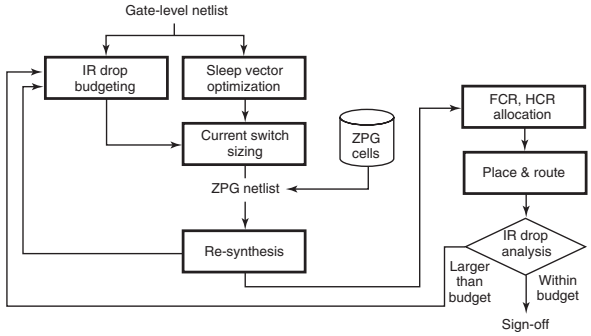

Figure 10: ZPG design flow.

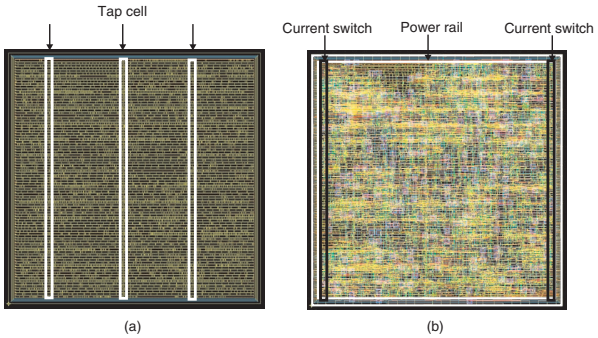

Figure 11: Layout of s38417: (a) after placement and (b) after routing.

We start the physical design stage by determining the number of FCRs and HCRs that will be required, based on the area of the cells to be placed in each type of circuit row. The footer and header cells are fixed in evenly spaced locations, and this is followed by automatic placement and routing. The voltage drop across the current switches is then checked on the layout to see if it is less than the chosen allowance.

Figure 11 shows the final layout of an example circuit s38417, which was obtained following the design flow in Figure 10.

4.2 Results of Sleep Vector Optimization

We performed experiments on a set of sequential circuits taken from the ISCAS and ITC benchmarks. We also included several circuits extracted from an audio codec core [14]. Each circuit was synthesized and mapped into a gate library on a commercial 65-*nm* technology.

Table 1: Experimental results on benchmark circuits

Benchmark				Average of ten random sleep vectors		Sleep vector from our approach		Improvement	
Name	# Gates	# F/Fs	# PIs	Transition power (μW)	Wirelength (μm)	Transition power (μW)	Wirelength (μm)	Transition power	Wirelength
s820	153	5	18	7.0	2926	3.2	2589	53.7%	11.5%
s1423	362	74	17	20.4	6963	12.3	6183	39.6%	11.2%
s5378	811	176	35	42.4	21031	24.0	19309	43.4%	8.2%
s9234	620	145	36	41.1	14387	27.2	13170	33.7%	8.5%
s35932	6309	1728	35	545.0	191010	351.4	177413	35.5%	7.1%
s38417	5488	1564	28	496.3	154082	404.5	134973	18.5%	12.4%
s38584	6357	1275	38	479.8	178802	328.7	165964	31.5%	7.2%
b03	90	30	4	8.5	1878	4.8	1754	42.9%	6.6%
b13	182	53	10	13.9	3302	7.0	3110	49.5%	5.8%
ac97_cra	118	24	42	6.5	3071	4.8	2881	26.9%	6.2%
ac97_soc	115	32	5	7.8	2013	4.6	1855	41.0%	7.9%
can_btl	247	31	29	16.2	3966	8.7	3812	46.3%	3.9%
Average								38.5%	8.0%

In Table 1, the second column shows the number of gates in the combinational subcircuit, and the third and the fourth columns are the number of flip-flops and primary inputs, respectively. The next two columns show the transition power and total wirelength, which were obtained following 4.1, on average, when we used ten randomly generated sleep vectors for each example, which represent the conventional approach. Columns 7 and 8 show the transition power and total wirelength, when we used the sleep vector obtained from the optimization process in 3.3; the last two columns show the improvement over conventional approach. The transition power was reduced by 38.5% on average, and the total wirelength by 8.0%.

The can_btl benefits least in wirelength. This is because it has relatively weak correlation between wirelength metric (recall that we used the number of flip-flops' fan-in and fan-out gates placed in FCRs as our metric) and actual wirelength.

5. CONCLUSION

Although zigzag power gating has previously been proposed to reduce the wake-up delay of power gating, the requirement for a zigzag arrangement of power rails has limited its use to custom circuits. We have proposed a complete design framework for ZPG circuits, that starts from a non-power-gated circuits down to the final layout of ZPG circuits. In the proposed design framework, we focused on selecting sleep vector, which was formulated as multiobjective optimization problem minimizing both wirelength and transition energy. The multiobjective genetic algorithm was employed to solve the problem, and we observed 38.5% and 8% of saving, on average, in transition energy and wirelength in 65-*nm* technology.

References

[1] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "A 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 8, pp. 847–854, Aug. 1995.

[2] K. Kumagai, H. Iwaki, H. Yoshida, H. Suzuki, T. Yamada, and S. Kurosawa, "A novel powering-down scheme for low Vt cmos circuits," in *Proc. Symp. on VLSI Circuits*, June 1998, pp. 44–45.

[3] P. Royannez, H. Mair, F. Dahan, M. Wagner, M. Streeter, L. Bouetel, J. Blasquez, H. Clasen, G. Semino, J. Dong, D. Scott, B. Pitts, C. Raibaut, and U. Ko, "90nm low leakage SoC design techniques for wireless applications," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2006, pp. 138–139.

[4] K. Agarwal, H. Deogun, D. Sylvester, and K. Nowka, "Power gating with multiple sleep mode," in *Proc. Int. Symp. on Quality Electronic Design*, Mar. 2006, pp. 633–637.

[5] M. Horiguchi, T. Sakata, and K. Itoh, "Swichted-source-impedance CMOS circuit for low standby subthreshold current giga-scale LSI's," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 11, pp. 1131–1135, Nov. 1993.

[6] K.-S. Min, H. Kawaguchi, and T. Sakurai, "Zigzag super cut-off CMOS (ZSCCMOS) block activation with self-adaptive voltage level controller: An alternative to clock-gating scheme in leakage dominant era," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2003, pp. 400–401.

[7] T. Miyazaki, T. Q. Canh, H. Kawaguchi, and T. Sakurai, "Observation of one-fifth-of-a-clock wake-up time of power-gated circuit," in *Proc. Custom Integrated Circuits Conf.*, Oct. 2004, pp. 87–90.

[8] D. Duarte, Y.-F. Tsai, N. Vijaykrishnan, and M. J. Irwin, "Evaluating run-time techniques for leakage power reduction," in *Proc. Int. Conf. on VLSI Design*, Jan. 2002, pp. 31–38.

[9] C. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. M. Despain, and B. Lin, "Power estimation methods for sequential logic circuits,," *IEEE Trans. on VLSI Systems*, vol. 3, no. 3, pp. 404–416, Sept. 1995.

[10] L. Benini and G. D. Micheli, "State assignment for low power dissipation," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 258–268, Mar. 1995.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Tr. on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[12] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Riccó, "Estimate of signal probability in combinational logic networks," in *Proc. European Test Conf.*, Apr. 1989, pp. 132–138.

[13] S. Mutoh, S. Shigematsu, Y. Gotoh, and S. Konaka, "Design method of MTCMOS power switch for low-voltage high-speed LSIs," in *Proc. Asia South Pacific Design Automation Conf.*, Jan. 1999, pp. 113–116.

[14] "Opencores," http://www.opencores.org/.