Semicustom Design of Zigzag Power-Gated Circuits in Standard Cell Elements

Youngsoo Shin, Senior Member, IEEE, Seungwhun Paik, Student Member, IEEE, and
Hyung-Ock Kim, Student Member, IEEE

Abstract—Zigzag power gating (ZPG) can overcome the long wake-up delay of standard power gating, but its requirement for both nMOS and pMOS current switches, in a zigzag pattern, requires complicated power networks, limiting application to custom designs. We propose a design framework for cell-based semicustom design of ZPG circuits, using a new power network architecture that allows the unmodified conventional logic cells to be combined with custom circuitry such as ZPG flip-flops, input forcing circuits, and current switches. The design flow, from register transfer level description to layout, is described and applied to a 32-b microprocessor design using a 1.2-V 65-nm triple-well bulk CMOS process. The use of a sleep vector in ZPG requires additional switching power when entering standby mode and returning to active mode. The switching power should be minimized so that it does not outweigh the leakage saved by employing ZPG scheme. We formulate the selection of a sleep vector as a multiobjective optimization problem, minimizing both the transition energy and the total wirelength of a design. We solve the problem by employing multiobjective genetic-based algorithm. Experimental results show an average saving of 39% in transition energy and 8% in total wirelength for several benchmark circuits in 65-nm technology.

Index Terms—Leakage, low-power, power gating, semicustom, standard cell, zigzag power gating (ZPG).

I. 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, as much as 50% in many technologies [1].

Power gating [2]–[4] 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. 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 (VSSV),

Fig. 1. Wake-up delay of power gating and ZPG circuit c7552 in 1.2-V 65-nm technology (a) without and (b) with a package model.

where the footer has its drain and the circuit sinks its current, slowly goes up until it reaches a steady state, which is usually close to VSSV. This implies that all the nets internal to the circuit are charged up to VSSV or close to VSSV, irrespective of their logic states. Many of those nets simultaneously start to be discharged once the footer is turned on. This leads to a large current flowing into the footer, which, together with the parasitic of the power networks and package, is a major cause of wake-up delay in power gating.

As an example, consider the c7552, which is one of the ISCAS benchmark circuits. It consists of 1713 gates after mapping to a commercial 65-nm 1.2-V gate library. We connected an appropriately sized footer [5] to the circuit and measured its wake-up delay. Fig. 1(a) shows the delay, which is the interval from turning on the footer to the point when VSSV settles down to its steady-state potential, which we assume to be within 5% of VDD [6] (0.06 V). This delay is about 0.223 ns. We then combined the RLC model of a package [6] with the c7552, and simulated the whole circuit to obtain a more realistic wake-up delay. Fig. 1(b) makes it clear that the large amount of current flowing into the footer induces ground bounce, which in turn leads to the substantial wake-up time of 44.1 ns.

Zigzag power gating (ZPG) [7]–[10], shown diagrammatically in Fig. 2, can reduce this delay. Before entering standby mode, an input vector, called a sleep vector, is applied to the
Fig. 2. ZPG circuit.

circuit (the block that produces the sleep vector is transparent in active mode). In a sequential circuit, part of the sleep vector corresponds to its primary inputs, and the remainder to the outputs of flip-flops. Since the sleep vector is predetermined, the logic states of all the nets during standby mode are known in advance. For gates with logic high outputs, we use a footer, and the remaining gates are connected to a header. Once the footer and header have been turned off, their drain potentials slowly go up and down until they reach the steady-state potentials. However, in contrast to simple power gating, all the nets keep their logic states.\(^1\) Thus, there is no need to discharge (or charge) the nets to restore their logic states when returning to active mode. Fig. 1 shows that the wake-up delay for a ZPG circuit, which is measured from turning on the footer and header to a point when \(V_{ddv}\) and \(V_{ssv}\) respectively, settle down to within 95% and 5% of \(V_{dd}\) [6], is considerably shorter (0.094 ns without and 8.2 ns with a package model) than that of simple power gating, for the same example circuit.

However, the use of both footers and headers in zigzag fashion has the drawback of complicated power networks, which is a very diverse power requirement: the gates connected to a header need \(V_{ddv}\) and \(V_{ss} \); those connected to a footer need \(V_{dd}\) and \(V_{ssv} \); flip-flops require all four power rails (as will be explained in Section III-C); a header, implemented as a cell, must itself be connected to \(V_{dd}\) and \(V_{ddv} \); and a footer must be connected to \(V_{ssv}\) and \(V_{ss} \). Therefore, we cannot use standard cells in conventional power networks with \(V_{dd}\) and \(V_{ss}\) rails. This effectively deters the widespread use of ZPG circuits, because most modern VLSI design depends on a cell-based top-down design methodology.

Another drawback of ZPG is the need for a sleep vector, which inevitably consumes additional switching power when entering standby mode and returning back to active mode. The selection of an appropriate sleep vector is therefore very important for minimizing the additional switching power requirement, so that it does not outweigh the leakage saved by employing ZPG scheme. Since the sleep vector determines the type of current switch (footer or header) to be connected to each gate, and the gates connected to the same type of current switch will be placed nearby due to the same power requirements, it is easy to see that the sleep vector will affect the layout as well as the switching power; and, as it turns out, the total wirelength as well.

In this paper, we address the question of how to design ZPG circuits using conventional standard cell elements, and how to determine a sleep vector which minimizes the requirement for extra switching power and wirelength during physical design. Our contribution can be summarized as follows.

1) We propose a power network architecture, which allows unmodified standard cell elements to be used for implementing ZPG circuits. A practical implementation of the required additional circuitry, including ZPG flip-flops, input forcing circuits, and current switches is also provided (Section III).
2) We formulate the selection of a sleep vector as a multiobjective optimization problem, minimizing both transition energy and total wirelength. We solve this problem using a multiobjective genetic-based algorithm (Section IV).
3) We describe the complete design flow, from register transfer level (RTL) description to layout, using commercial computer-aided design tools (Section V) and test our proposal on a 32-b microprocessor design using 65-nm commercial technology (Section VI).

The remainder of this paper is organized as follows: In the next section, we review several variants of power gating for fast wake-up. In Section III, we describe our power network architecture for ZPG circuits, the placement of current switches on the network, and the elements of a ZPG circuit. In Section IV, we address the problem of sleep vector optimization, and then present our experimental results. The complete design flow of ZPG circuits is presented in Section V and its application to a 32-b microprocessor is studied in Section VI. We draw conclusions in Section VII.

\(^1\) An example is the inverter in Fig. 2, in which the nMOS remains turned on even though the header, which is in series with the pMOS, is turned off.
II. FAST POWER GATING

There are several alternative schemes (including ZPG) to reduce the large wake-up delay of power gating, and Fig. 3 compares them, in terms of wake-up delay and leakage saving, against unmodified power gating. These figures were obtained using the same circuit (c7552 in 65-nm technology) that we mentioned in the previous section.

Power gating with a two-pass turn-on [11] employs a weak nMOS switch that reduces current surges during wake-up by sinking a small amount of current to restore power initially. Subsequently, a higher capacity nMOS switch delivers the current for normal operation. Fig. 4(a) is a conceptual circuit diagram. When this two-pass turn-on was applied to a test circuit, the wake-up delay was reduced to 30% of that for simple power gating while the leakage saving remained the same, as shown in Fig. 3.

Power gating with multiple sleep modes [12], shown in Fig. 4(b), trades leakage saving for wake-up delay by controlling the potential of the virtual power rail by applying different voltages to the gate of the current switch. One of the modes, called sleep [12], can reduce the wake-up delay by 56% more than power gating (by applying 200 mV to the gate of the footer), but the leakage saving is only 11%.

Virtual power and ground rail clamps [13] limit the potential across both the header and footer by employing a forward-biased diode in parallel with each current switch, as shown in Fig. 4(c). The potential of each virtual rail tends to drop or to rise once the switches are turned off, but the rails are clamped by the built-in potential of the diodes, which helps both to reduce the wake-up delay and to hold the state of the flip-flops. As shown in Fig. 3, this scheme reduces the wake-up delay by 29% with a leakage saving of 44% of that for simple power gating.

The ZPG circuit exhibits good balance between wake-up delay and leakage saving, as shown in Fig. 3: it has the shortest wake-up delay (19% of that of power gating) while its leakage saving is the same as power gating (proportion of gate leakage is small in 65-nm technology we used; leakage saving from ZPG will be less in technologies where gate leakage takes large proportion). This motivates us to focus on ZPG as a candidate for power gating with a short wake-up delay.

III. CELL-BASED DESIGN OF ZPG CIRCUITS

A. Power Network Architecture

Fig. 5(b) shows the power network architecture that we propose for ZPG circuits based on standard-cell elements. It has two types of circuit row: a header-connected circuit row (HCR) with rails at \( V_{ddv} \) and \( V_{ss} \), containing the gates that are connected to a header [e.g., the inverter and the NOR gate].
in Fig. 5(a)]; and a footer-connected circuit row (FCR) with rails at \( V_{dd} \) and \( V_{ssv} \), containing the gates that are connected to a footer [e.g., the NAND gate in Fig. 5(a)]. The VDD and GND terminals of standard cells are of course connected to the appropriate power rails. For instance, the VDD terminal of the inverter cell in Fig. 5(a) is automatically connected to \( V_{ddv} \), and its GND terminal is connected to \( V_{ssv} \). This arrangement allows us to use unmodified standard cells, which is a major advantage of the proposed power network; however, their placement is restricted to the corresponding type of circuit row. For example, the inverter and the NAND gate are not allowed at the same circuit row. Employing feedthrough cells would allow us to use the same circuit row for both types of gate [14], but this would require the development of a dedicated placement engine.

The number of FCRs and HCRs that will be required is determined by the area of the cells to be placed in each type of circuit row, which in turn is determined by sleep vector following the procedure in Section IV. Once we know the number of each circuit row, we need to determine how to interleave them, i.e., how many FCRs are followed by how many HCRs. This is based on area efficiency and routability. As we increase the number of FCRs (and HCRs), we have more chance to apply double-back layout pattern that helps reduce area, but it degrades the routability since the average distance between the cells in FCRs and in HCRs will increase. We elaborate more on this in conjunction with experiments in Section VI.

The bodies (or substrates) of the standard cells that we use are tied to GND terminals. Thus, the bodies of the gates in an HCR are tied to \( V_{ssv} \), while the gates in an FCR have their bodies tied to \( V_{ssv} \) (i.e., the GND terminals of gates in the FCR are connected to the \( V_{ssv} \) rails). To avoid the possibility of a short between \( V_{ss} \) and \( V_{ssv} \), the body contacts of all the cells in the FCRs are removed after placement. Their substrate bias can be provided by regularly spaced body tap cells, as shown in Fig. 5(b).

The gates in an FCR experience reverse body bias, because the bodies of nMOSes are at \( V_{ssv} \), while their GND terminals are tied to \( V_{ss} \), which is higher than \( V_{ssv} \). However, by properly sizing the footer [5], the increase in delay that results from this reverse body bias turns out to be negligible for most circuits. Nevertheless, we could eliminate this undesirable effect entirely, as well as the extra effort required to remove the body contacts of the cells in the FCRs and the requirement for regularly spaced body tap cells, if we could use triple-well process, which would allow the p-wells of each circuit row to be isolated from each other. Although placement of the logic gates is necessarily restricted, it is important that clocked elements, such as flip-flops, can be placed anywhere in the available placement region. This is because flip-flops are often the leaves of a clock tree network [15]: if their placement is restricted, then the placement of associated circuit elements in the clock tree, such as buffers and clock splitters, will also be restricted, which makes clock design very difficult. To guarantee unrestricted placement and to provide all power rails (as explained in Section III-C), we redesigned the flip-flops so that they are connected to adjacent power rails through their ports instead of their VDD and GND terminals, as shown in Fig. 5(b).

As shown in Fig. 2, an input forcing circuit is located at each primary input to provide a sleep vector. The forcing circuits that provide logic high are placed in the FCRs; those that provide logic low are placed in the HCRs. The footer cells, shown as boxes of hatched patterns, are placed in the FCRs; the header cells in the HCRs.

### B. Placement of Current Switches

The footer cells are placed in the HCRs, where \( V_{ss} \) is available, and their drains are connected to \( V_{ssv} \) in the adjacent FCRs, or directly to the \( V_{ssv} \) rings if the footer cells are
sufficiently close to them, as shown in Fig. 5(b). Similarly, the header cells are placed in the FCRs and draw \( V_{dd} \) from the adjacent HCRs, or are connected to the \( V_{dd} \) rings. The design of the footer and header cells themselves will be explained in the next section.

The current switches can be placed anywhere within the corresponding circuit rows, as shown in Fig. 6(a). This flexibility of placement allows current switches and logic cells to be submitted together to an automatic placement engine, with the constraint that the type of circuit row must be specified for each element. However, if the current switches happen to be placed too far from the power rings, which carry a large current to the logic cells, there is the possibility of a large IR drop on the \( V_{dd} \) and \( V_{ss} \) rails. Fig. 6(a) shows the current path, for example, cells on an HCR and an FCR, and the thick dotted arrows indicate places where an excessive IR drop might occur, since each header needs to supply current to many cells (which also applies to the footers). Therefore, automatic placement can only be allowed when a large number of header and footer cells are dispersed over the placement region, so that each cell only carries the current for a small number of adjacent logic cells.

If the current switches are restricted to the boundary of the circuit rows, as shown in Fig. 6(b), then the IR drop due to the current switches can be kept small, at the cost of restricted placement. The IR drop across this form of power network will be analyzed in conjunction with the application of the proposed design methodology in Section VI.

C. ZPG Cell Design

Fig. 7 shows the use of D flip-flops in a ZPG circuit. The flip-flop outputs contribute part of the sleep vector, as shown in Fig. 2. The type of flip-flop shown in Fig. 7(a) is employed where a bit of the sleep vector requires logic low, which is provided by a NOR gate at its output. We use the type of flip-flop shown in Fig. 7(b) when we want logic high in the corresponding bit of the sleep vector. Note that the two inverters in the second latch (enclosed in dashed boxes in Fig. 7) are directly connected to \( V_{dd} \) and \( V_{ss} \), instead of to \( V_{ddv} \) or \( V_{ssv} \), so they are not power gated. This allows them to retain the state of the flip-flop during standby mode, which is then restored after wake-up. In addition, the second latch employs high \( V_{th} \) to reduce its subthreshold leakage. Note also that the NOR and NAND gates at each output are connected to the header and footer through \( V_{ddv} \) and \( V_{ssv} \), respectively, to reduce their subthreshold leakage.

The sleep vector determines the flip-flop input (marked D in Fig. 7) during standby mode (see Fig. 2), which implies that the logic states of the three inverters at the front of the flip-flop during standby will be known in advance. Those inverters are therefore connected either to the header or to the footer, depending on the flip-flop input during standby mode. Thus, we need four types of flip-flop in total, to allow for the two possible flip-flop inputs, and the two flip-flop outputs that may be enforced during standby. For example, the flip-flop in Fig. 7(b) is used where the input is logic low but we want logic high in the corresponding bit of the sleep vector. Since our flip-flops require all four power rails, and we do not want to have any restrictions on their placement, for the reasons discussed in the previous section, the power rails are connected to ports instead of to the VDD and GND terminals. This is shown in the layouts.

One last issue on ZPG flip-flops is n-well isolation. The type of flip-flop that we are proposing has its n-well tied to \( V_{dd} \) due to the two inverters in the second latch, which are responsible for state retention. When it is placed in an HCR, in which the other cells have their n-wells tied to \( V_{dd} \), its n-well has to be isolated, which incurs an area overhead. On the other hand, when a flip-flop is placed in an FCR, we do not need n-well isolation. This observation will be used in the next section to reduce the area by regulating the proportion of FCRs to HCRs by putting flip-flops into FCRs wherever possible.

Fig. 8 shows the input forcing circuits which are used at the primary inputs to provide part of the sleep vector. They are transparent during active mode (standby is low). In standby mode, the transmission gate decouples the primary input from the circuit, and then the nMOS and pMOS devices provide the sleep vector with logic low and high, respectively. Note that the circuit in Fig. 8(a) which enforces logic low can only be placed in an HCR, because nMOS needs \( V_{ss} \) rails. Similarly, the circuit in Fig. 8(b) has to be placed in an FCR.

Fig. 9 shows the cell layouts of a footer and a header. Each slice is a unit current switch; when slices are abutted as shown in the figure, they become a larger current switch (the blocks on the left and right supplement the layouts). This facilitates the

\footnote{If n-wells of the cells in HCR are to be connected to \( V_{dd} \), this problem does not happen. However, extra body tap cells (in addition to those in FCRs) need to be placed in HCRs to provide well bias [see Fig. 5(b)]; pMOS devices in HCR experience reverse body bias.}
construction of a current switch cell of a size which depends on the total number of individual current switches and how we distribute them over placement region.

IV. SLEEP VECTOR OPTIMIZATION

A. 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 an area overhead due to n-well isolation, as we discussed in the previous section. We tested several ISCAS benchmark circuits in 65-nm technology. Fig. 10(a) compares the area of three ZPG circuits implemented in different styles (white and black bars) with that of the original non-ZPG circuit (gray bar). An increase in area for ZPG is inevitable, as there is more circuitry, and the presence of the FCRs and HCRs restricts the flexibility with which cells can be placed. When we compare only two styles of ZPG implementations, it can be readily seen that it is better to restrict flip-flops to FCRs. This may conflict with our initial goal of designing ZPG flip-flops such that they can be freely placed. However, even if they have to be placed in FCRs, the flip-flops are not greatly restricted in their placement. In these three circuits, on average, they can be freely located in 77% of the placement region. This is because we recomputed the area of the cells that will be placed in FCRs and HCRs, on the basis that flip-flops will be placed in FCRs, and then determined the number of FCRs and HCRs we need for each design.

We also compared the total wirelength (after detailed routing) of two implementation styles [the white and black bars in Fig. 10(b)]. The wirelength tends to decrease in most benchmarks (including other circuits not shown in Fig. 10) when the placement of flip-flops is restricted to FCRs (black bars), which is also preferable in terms of area. This is because most of the placement region is now assigned to FCRs, and the additional flexibility in placement translates into shorter wires in the designs.

To investigate the potential for further reduction in wirelength, while keeping the flip-flops in the FCRs, we experimented with s5378. The total wirelength of the non-ZPG implementation of this circuit is about 12.6 mm, while that of the ZPG implementation is about 21.4 mm. Of the extra 8.8 mm, about 16.1% is due to the standby signal (which is routed to the footers, headers, flip-flops, and input forcing circuits) and about 6.4% is caused by the input forcing circuits; these increases are inevitable. The remaining 77.5% of the extra wirelength was mainly a result of the restriction on placement (thus hard to categorize), but it turned out that the wires between flip-flops, now all in FCRs, and the gates in the HCRs, which are either fan-ins or fan-outs of the flip-flops, take a large proportion of it. Note that flip-flops usually have many gates connected to them (either D or Q), so that a significant wirelength can be attributed to flip-flops, particularly in flip-flop-dominated
controllers. The fact that the type of circuit row assigned to each gate is determined by the sleep vector suggests that the 77.5% of extra wirelength required for the ZPG implementation of s5378 can be controlled to some extent.

We explored this possibility by generating 20 random sleep vectors, for each of which the circuit was placed and routed. Then, we measured total wirelength and counted the number of flip-flops’ fan-in and fan-out gates that were placed in FCRs. Fig. 11(a) shows the results. 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 clear that the sleep vectors that lead to more number of flip-flops’ fan-in and fan-out gates to be placed in FCRs (the y-axis) tended to produce shorter wirelength, which is also confirmed by the result of the same experiment on s1423, as shown in Fig. 11(b).

In summary, we place flip-flops in FCRs in favor of area. We aim to select a sleep vector which lead to as many flip-flops’ fan-in and fan-out gates as possible being placed in FCRs, in the expectation of a reduction in wirelength.
B. 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. Switching power is also consumed when the sleep vector is removed as the circuit returns to active mode. The energy used in these transition needs to be minimized so that it does not outweigh the leakage saved by employing ZPG scheme. The minimum idle time for which ZPG scheme saves energy can be readily derived by equating the areas under the two curves in Fig. 12 and solving for $T_{idle}$

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

where $E_{tr \_2 \_sb}$ and $E_{tr \_2 \_at}$ are, respectively, the amount of transition energy required to enter standby mode and to return to active mode. These quantities are assumed to be equal and are both denoted by $E_{tr}$. The power consumption of the non-ZPG circuit is denoted by $P_{idle}$ and that of the ZPG circuit is denoted by $P_{sb \_zpg}$, which is usually negligible compared to $P_{idle}$. The transition times from active to standby and back again are assumed to be the same, and are denoted by $T_{tr}$. From (1), it is clear that transition energy should be minimized for short $T_{idle}$, allowing a ZPG circuit to enter standby as many chances of idle period as possible, to maximize the power saved.

We experimented with s1423 to see how the choice of sleep vector affects transition energy. We assumed that the signal probabilities for the primary inputs when the circuit is in idle are available, which is usually possible from the knowledge of typical usage. The state probabilities of flip-flops, which constitute a part of the input vector for the combinational part of a circuit, can be obtained by solving a nonlinear system equations using the Picard–Peano method [17], [18]. Random patterns for the primary inputs and flip-flop outputs were generated from the signal probabilities and used to obtain the average transition energy $E_{tr}$ for 20 random sleep vectors. The same patterns were used to obtain average idle power $P_{idle}$ which then gives us $T_{idle}$. The computed values of $T_{idle}$ are shown in Fig. 13(a) for s1423 and in Fig. 13(b) for s820. It is clear that a felicitous choice of sleep vector can reduce the transition energy and, hence, $T_{idle}$.

C. Multiobjective Genetic Algorithm-Based Search of Sleep Vector

Sections IV-A and B set two objectives for a sleep vector: to minimize wirelength and transition energy. Since these objectives can conflict with each other, this casts itself as a multiobjective optimization problem, which we can approach with a multiobjective genetic algorithm [19].

Fig. 14 summarizes our algorithm. Its input is a gate-level netlist, with the signal probabilities of the primary inputs when the circuit is in standby mode, as discussed in Section IV-B. We then derive the state probabilities of the flip-flops (L1) [17]. The probabilities of the primary inputs and the flip-flop states are then propagated [20] through a combinational portion of the netlist to obtain the signal probabilities of all the internal nets (L2), which are used to obtain transition energy metric (L8).

We initially generate $N$ random sleep vectors $P$ (L3), from which another $N$ sleep vectors $Q$ are generated (L5) through crossover and mutation. In the crossover operation, we randomly select two sleep vectors from $P$, randomly pick one bit position, cut each vector at that position, and swap the pieces to make two new vectors. Mutation is performed (with some fixed probability) after crossover by flipping one randomly chosen bit of each of the two new vectors.

For each of the $2N$ sleep vectors in $P$ and $Q$, we evaluate the metrics of wirelength and transition energy (L7 and L8). The wirelength metric is obtained by counting the number of flip-flops’ fan-in and fan-out gate which have logic high output, so that they will be placed in FCRs. The larger the wirelength metric, the better, because we know that there is some level of inverse correlation (even though it is not strong in some examples) between the total wirelength and the number of flip-flops’ fan-in and fan-out gates placed in FCRs (see Section IV-A and Fig. 11). The transition energy metric is approximated by the sum of the switched capacitance of all internal nets

$$\sum_i C_i (l_i - (1)^i p_i) \quad (2)$$

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

The $2N$ sleep vectors are classified by a dominance relation, which is based on the wirelength and transition energy metrics (L9). A vector $R_i$ is dominated by a vector $R_j$ if $W(R_i) \leq W(R_j)$ and $E(R_i) \geq E(R_j)$, so that $R_i$ is inferior to $R_j$ in both metrics. The class $F_1$ contains the vectors that are not dominated by any other vectors in $R$, and thus it represents the Pareto points [21] in the current generation. A second class $F_2$ contains the vectors that are dominated only by the vectors in $F_1$. Further classes are defined similarly.

Once the vectors have been classified, we select $N$ out of $2N$ vectors (L11 to L15), which then constitute the parent vectors $P$ of the next generation (i.e., the next iteration of L4), which is shown in Fig. 15. This selection is done by including vectors in the classes one by one, starting from $F_1$, until we have chosen $N$ vectors (L12 to L14). We usually have to select a subset of the vectors from some set $F_j$ (L15), and
Fig. 13. Minimum idle time $T_{idle}$ for various sleep vectors. (a) s1423. (b) s820.

### Sleep Vector Search Based on Multi-objective Genetic

**Input:** gate-level netlist with signal probabilities of PIs  
**Output:** sleep vector  

```
begin  
L1 Derive state probabilities of FFs  
L2 Derive signal probabilities of internal nets  
L3 Generate $N$ initial parent sleep vectors, $P = (P_1, P_2, \ldots, P_N)$  
L4 for $i = 1, 2, \ldots, M$ do  
L5 Generate $N$ offspring sleep vectors, $Q = (Q_1, Q_2, \ldots, Q_N)$  
L6 $R = P \cup Q$  
L7 Evaluate wirelength metric $W(R_j), \forall R_j \in R$  
L8 Evaluate transition energy metric $E(R_j), \forall R_j \in R$  
L9 $F = (F_1, F_2, \ldots) = \text{non-dominated.sorting}(R, W, E)$  
L10 if $i \neq M$ then  
L11 $P = \emptyset$ and $j = 1$  
L12 while $|P| + |F_j| \leq N$ do  
L13 $P = P \cup F_j, \ j = j + 1$  
L14 end do  
L15 $P = P \cup \text{compute.crowding.distance}(F_j)$  
L16 else return select.one.Pareto.point($F_1$)  
L17 end if  
L18 end do  
end
```

Fig. 14. Pseudocode of sleep vector search, based on a multiobjective genetic algorithm.

we do this in such a way that the vectors are distributed as evenly as possible in the space spanned by the wirelength and transition energy metrics. This is accomplished by computing a **crowding distance** for each vector in $F_j$. Assume that $F_j$ has $n$ vectors $v_1, v_2, \ldots, v_n$, which are arranged in order of 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

$$d(v_i) = \begin{cases} \frac{E(v_{i+1})-E(v_{i-1})}{E(v_n)-E(v_1)} + \frac{W(v_{i+1})-W(v_{i-1})}{W(v_n)-W(v_1)}, & \text{if } 1 < i < n \\ 0, & \text{otherwise.} \end{cases}$$

The vectors with the greatest crowding distance are selected first. Fig. 16 shows an example. If we require three vectors, $v_1$, $v_4$, and $v_5$ will be selected. Note that the first and the last vectors ($v_1$ and $v_n$) are always selected, unless we only need one vector, in which case we choose either $v_1$ or $v_n$.

![Figure 15](image1)

**Fig. 15.** Parent sleep vectors $P$, offspring sleep vectors $Q$, and nondominated classes $F$.

![Figure 16](image2)

**Fig. 16.** Example computation of crowding distance.

When the iterations are complete (L16), we need to select one sleep vector from the Pareto points in $F_1$. There are various ways to do this, but currently we sort the points in $F_1$ in one metric and select the median. This is also the median in the other metric, since the points in $F_1$ are not dominated by any others. Fig. 17 shows the Pareto points of four sample generations of four example circuits, and demonstrates how they gradually improve from generation to generation.

### D. Experimental Results

We performed sleep vector optimization on a set of sequential circuits taken from the ISCAS and International Test Conference (ITC) benchmarks. We also included several circuits extracted from an audio codec core [22]. Each circuit was...
synthesized and mapped into a gate library using a commercial 65-nm technology.

The second column of Table I shows the number of gates in the combinational subcircuit, and the third and the fourth columns, respectively, are the number of flip-flops and primary inputs. The next two columns show the average transition power and average total wirelength, which were obtained using ten randomly generated sleep vectors for each circuit, which represents the conventional approach. Columns 7 and 8 show the transition power and total wirelength for the sleep vector generated by the optimization process described in Section IV-C; and the last two columns show the improvement over the conventional approach. The transition power was reduced by 38.5% on average, and the total wirelength by 8.0%. The circuit can_btl benefits least in terms of wirelength. This is because it exhibits relatively weak correlation between the wirelength metric (which is the number of flip-flops’ fan-in and fan-out gates placed in FCRs) and actual wirelength.

The last column of Table I shows the standby leakage of ZPG circuit designed with sleep vector from our approach, as a factor of the standby leakage of nonpower-gated circuit.

The last column of Table I shows the standby leakage of ZPG circuit designed with sleep vector from our approach, as a factor of the standby leakage of nonpower-gated circuit.

V. DESIGN FLOW FOR STANDARD CELL-BASED ZPG CIRCUITS

The design flow for ZPG circuits is shown in Fig. 18. The RTL design goes through a standard logic synthesis to create the initial gate-level netlist. To determine the size of the current switches (i.e., footers and headers), which affects the active-mode circuit delay, we first decide on the voltage drop, called IR drop budgeting, which we will allow at the switches when they are turned on during active mode. The netlist is then resynthesized with \( V_{dd} \) set to the voltage swing that each gate will experience, which is the \( V_{dd} \) minus the chosen voltage drop across a footer or a header. If the timing constraint cannot be satisfied by this resynthesis, we reduce the voltage drop across the current switches, even though this will increase the switch size, and perform resynthesis again.

We then determine the sleep vector, using the method described in the previous section. The sleep vector determines the type of input forcing circuit required at each primary input, as well as the type of ZPG flip-flop (see Fig. 7) to be substituted for each flip-flop in the original netlist. 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, we can size the current switches [5], which in turn determines the number of slices (see Fig. 9) that will be required.

Once we obtain ZPG netlist, 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

\[4\] Note that each gate is connected either to a footer or to a header, but not to both. We choose the same voltage drop for the footers and headers.
Fig. 18. Design flow of a standard cell-based ZPG circuit.

Fig. 19. Block diagram of the 32-b microprocessor used as a case study.

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; current switch is resized if the voltage drop is larger than the allowance.

VI. CASE STUDY: A MICROPROCESSOR

We tested the design flow proposed in the previous section and the sleep vector optimization of Section IV-C on a 32-b microprocessor [23]. Fig. 19 shows its block diagram. The microprocessor is a reduced-instruction-set-computer (RISC)-type processor with five pipeline stages, supporting the Microprocessor without Interlocked Pipeline Stages (MIPS) instruction set architecture. The original design used in this experiment consists of 26 K gates after mapping on to a commercial 65-nm 1.2-V gate library. The power managing unit is responsible for generating a STANDBY signal when it receives the external SLEEP signal. The processor is returned to normal active mode by deasserting the SLEEP signal.

The original processor was transformed to a ZPG circuit following the design flow of Fig. 18. We determined the sleep vector using the procedure described in Section IV-C. Then, an input forcing circuit was associated with each of the 135 primary inputs, and ZPG flip-flops were substituted for 1196 storage elements. The average current [5] drawn by the gates connected to footers was about 1.6 mA, and those connected to headers drew about 1.5 mA. This current, along with the selected voltage drop of 50 mV across each header and footer switch, required the switches at the header and footer to be constructed from 21 and 29 slices, respectively. The modified design was then resynthesized with $V_{dd}$ set to 1.15 V (1.2 V–50 mV), using a target frequency of 200 MHz.

After resynthesis, we first determined the number of FCRs and HCRs from the area of the cells to be placed in each type of circuit row: this required 75 and 36 circuit rows, respectively. The footer and header slices were distributed on the left- and right-hand side of the placement region, as discussed in Section III-B. After they were fixed in position, automatic placement and routing were applied to the whole design. Fig. 20(a) shows the final layout. The STANDBY signal is routed to many circuit elements including flip-flops, input-forcing circuits, and current switches. This requires a large number of buffers, which are not power gated, so that they are directly

Authorized licensed use limited to: Korea Advanced Institute of Science and Technology. Downloaded on December 9, 2009 at 19:54 from IEEE Xplore. Restrictions apply.
connected to \( V_{dd} \) and \( V_{ss} \). These buffers are only placed in FCRs, because otherwise there would be an overhead for n-well isolation.

The area of the design is compared to the original non-power-gated processor as well as to the processor with standard power gating implementation. The second column of Table II shows the sum of the areas of all the cells in each design. The ZPG implementation has 8.4% more area: 4.8% is attributed to the ZPG cells, 2.9% to the body tap cells [see Fig. 5(b)], and 0.7% to the current switches. However, the overhead is smaller than that from power gating implementation. In the fourth column, we compare the area of placement region, which is defined as a rectangular region that covers all the circuit rows and routing channels between them. For each design, we force about 80% of circuit rows to be occupied by the cells, which is a tight placement. Double-back layout pattern was applied to both non-power-gated and power-gated designs for efficient use of area. For ZPG design, each 12 FCRs were followed by six HCRs, implying that double-back was applied to consecutive 12 FCRs and six HCRs, but not to whole circuit rows; this turned out to be a reasonable choice for both area efficiency and routability as we discussed in Section III-A. The ZPG implementation now has 10% more area, since double back was not applied to whole circuit rows, but still has smaller overhead of area than power-gated design.

The voltage variations across all four power networks were checked, as shown in Fig. 20(b). The maximum drop occurred on the \( V_{ddv} \) network, and was 11 mV, which is well within allowable voltage drop of 50 mV. Note that the maximum drop includes the IR drop in the \( V_{dd} \) and \( V_{ddv} \) networks, as well as that across the header. The maximum voltage rise occurred on the \( V_{ssv} \) network, and was 38 mV, as shown in Fig. 20(b). Again, this is well within the initial allowance of 50 mV.

The original non-power-gated processor draws about 126 \( \mu A \) of standby leakage; the ZPG processor draws about 10 \( \mu A \), which is 7.9% of leakage.

### VII. Conclusion

Although ZPG 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 application to custom circuits. We have proposed a power network architecture for ZPG circuits which allows standard cells to be used without any modification. The flip-flops, input forcing circuits, and current switches in our circuit have been designed to complement a cell-based ZPG design. We formulated the selection of a sleep vector as a multiobjective optimization problem, minimizing the transition energy and the total wirelength of the design, and solved it using a multiobjective genetic algorithm. Both our cell-based ZPG design and the sleep vector optimization have been tested on several benchmark circuits. The design flow, from RTL to layout, has been successfully applied to a 32-b microprocessor implemented in 65-nm CMOS technology.

### Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions.

### References


