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

Seunghun 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 multi-objective 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. 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 multi-objective 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.

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_{SS}) and V_{DD}) 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_{SS}, the gates connected to a header need V_{DD} 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 1: Block diagram of zigzag power-gated circuit.](image-url)
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 $V_{dd}$ 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 $V_{ssv}$. 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.

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. Figure 3 shows one of ZPG D flip-flops, 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

---

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

2There are four of them, depending on the corresponding bit value of the sleep vector bit it provides, and the D-input in standby mode.
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. This is because HCR gates have their n-well tied to Vdd; the n-well of the flip-flop should be at Vdd because of isolation, which is responsible for state retention (see Figure 3). This fact has important implications 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 zero in Figure 2, is placed in FCR and the one that provides logic low, a block labeled input forcing one in Figure 2, is placed in FCR and the one that provides the corresponding bit value of the sleep vector, is placed in HCR due to the availability of Vdd and Vss, respectively. Footer cells and header cells are placed in HCRs and FCRs, respectively, as shown in Figure 2, as they require access to Vdd and Vss. Although current switches can be placed anywhere within corresponding circuit rows, the boundary is preferred to minimize current path from virtual power rails (Vdd or Vss) to real power rails (Vdd or Vss), which helps minimize IR drop on power rails.

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 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 savings from ZPG scheme, \( T_{idle} \), can be readily expressed by [8]:

\[
T_{idle} = \frac{E_{tr_{2\rightarrow 2}} + E_{tr_{2\rightarrow 0}} - P_{idle}(2T_{tr})}{P_{idle} - P_{idle}} = \frac{2E_{tr}}{P_{idle}}
\]

where \( E_{tr_{2\rightarrow 0}} \) and \( E_{tr_{2\rightarrow 2}} \) are transition energy to enter standby and to return back to active mode, respectively. They are assumed to be equal in magnitude, thus denoted as \( E_{tr} \).
power consumption of non-ZPG circuit is denoted as $P_{idle}$; that of ZPG circuit is denoted as $P_{idle,ZPG}$, 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 $T_{T}$. 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 PIs and POs. 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 $P$ (L3), from which another $N$ sleep vectors $Q$ are generated (L5) through crossover and mutation. For crossover, we randomly select two sleep vectors from $P$, 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 $P$ and $Q$, 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, 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:

$$\sum_{i} C_{l} \left( l_{i} + (-1)^{j} p_{i} \right),$$  

where $C_{l}$ 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 $R$, 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 $P$ in the next generation. The next generation is then generated via crossover, mutation, and selection (L16 to L21).

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

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

**Figure 8**: Example of computing crowding distance.
generation (i.e. iteration). This is done by including \( F_1 \) 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:

\[
\frac{E(v_{i+1}) - E(v_i)}{E(v_n) - E(v_1)} + \frac{W(v_{i+1}) - W(v_i)}{W(v_n) - W(v_1)} \quad \text{if } 1 < i < n
\]

\[
\text{otherwise}
\]

The vectors are selected from the one with the maximum crowding distance. Figure 8 shows an example. If we select three, \( v_1, v_3 \), 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.

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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th># Gates</th>
<th># F/F's</th>
<th># PIs</th>
<th>Transition power (µW)</th>
<th>Wirelength (µm)</th>
<th>Sleep vector from our approach</th>
<th>Transition power (µW)</th>
<th>Wirelength (µm)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>s820</td>
<td>153</td>
<td>5</td>
<td>18</td>
<td>7.0</td>
<td>2926</td>
<td>3.2</td>
<td>2589</td>
<td>33.7%</td>
<td>11.5%</td>
</tr>
<tr>
<td>s1423</td>
<td>362</td>
<td>74</td>
<td>17</td>
<td>20.4</td>
<td>6963</td>
<td>12.3</td>
<td>6183</td>
<td>39.6%</td>
<td>11.2%</td>
</tr>
<tr>
<td>s5378</td>
<td>811</td>
<td>176</td>
<td>35</td>
<td>42.4</td>
<td>21031</td>
<td>24.0</td>
<td>19309</td>
<td>43.4%</td>
<td>8.2%</td>
</tr>
<tr>
<td>s9234</td>
<td>620</td>
<td>145</td>
<td>36</td>
<td>41.1</td>
<td>14387</td>
<td>27.2</td>
<td>13170</td>
<td>33.7%</td>
<td>8.5%</td>
</tr>
<tr>
<td>s35932</td>
<td>6309</td>
<td>1728</td>
<td>35</td>
<td>545.0</td>
<td>191010</td>
<td>351.4</td>
<td>17413</td>
<td>35.5%</td>
<td>7.1%</td>
</tr>
<tr>
<td>s38417</td>
<td>5488</td>
<td>1564</td>
<td>28</td>
<td>496.3</td>
<td>154082</td>
<td>404.5</td>
<td>134973</td>
<td>18.5%</td>
<td>12.4%</td>
</tr>
<tr>
<td>s38584</td>
<td>6357</td>
<td>1275</td>
<td>38</td>
<td>479.8</td>
<td>178802</td>
<td>328.7</td>
<td>165964</td>
<td>31.5%</td>
<td>7.2%</td>
</tr>
<tr>
<td>b03</td>
<td>90</td>
<td>30</td>
<td>4</td>
<td>8.5</td>
<td>1878</td>
<td>4.8</td>
<td>1754</td>
<td>42.9%</td>
<td>6.6%</td>
</tr>
<tr>
<td>b13</td>
<td>182</td>
<td>53</td>
<td>10</td>
<td>13.9</td>
<td>3302</td>
<td>7.0</td>
<td>3110</td>
<td>49.5%</td>
<td>5.8%</td>
</tr>
<tr>
<td>ac97_cra</td>
<td>118</td>
<td>24</td>
<td>42</td>
<td>6.5</td>
<td>3071</td>
<td>4.8</td>
<td>2881</td>
<td>26.9%</td>
<td>6.2%</td>
</tr>
<tr>
<td>ac97_soc</td>
<td>115</td>
<td>32</td>
<td>5</td>
<td>7.8</td>
<td>2013</td>
<td>4.6</td>
<td>1855</td>
<td>41.0%</td>
<td>7.9%</td>
</tr>
<tr>
<td>can_btl</td>
<td>247</td>
<td>31</td>
<td>29</td>
<td>16.2</td>
<td>3966</td>
<td>8.7</td>
<td>3812</td>
<td>46.3%</td>
<td>3.9%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>483</strong></td>
<td><strong>76</strong></td>
<td><strong>54</strong></td>
<td><strong>20.5</strong></td>
<td><strong>3059</strong></td>
<td><strong>17.2</strong></td>
<td><strong>1754</strong></td>
<td><strong>38.5%</strong></td>
<td><strong>6.0%</strong></td>
</tr>
</tbody>
</table>

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


