Precomputation-based Guarding for Dynamic and Leakage Power Reduction

Afshin Abddollahi, Massoud Pedarm

University of Southern California

{Afshin,Pedram}@usc.edu

Farzan Fallah, Indradeep Ghosh

Fujitsu Labs of America

{farzan,ighosh}@fla.fujitsu.com

Abstract *- This paper presents a precomputation-based guarding technique to reduce both dynamic and static power consumptions in CMOS VLSI circuits. More precisely, a high threshold sleep transistor is placed in series with some portions of the circuit. Based on the input values of the circuit, the sleep transistor is turned on and off, thus, saving both dynamic and static power. We show how to apply this technique to a number of common arithmetic blocks, including comparators, adders and multipliers. Finally, dynamic guarding and sleep transistor activity reduction techniques for improving the performance of the method are presented. Experimental results show 81% reduction in the power consumption of data path modules of a commercial VLIW processor can be achieved using our techniques. This is 20% higher than what has been achieved by previous methods.*

1. Introduction

Low power design has become a very important design technology driver. Widespread use of portable battery-powered electronic systems is one of the key reasons for the increasing importance of low power design. Power dissipation in these kinds of systems may be divided into two major components: dynamic power consumption and leakage power dissipation. There are different factors contributing to the dynamic power consumption in CMOS circuits. Most of the dynamic power is consumed for charging and discharging the load capacitances of logic gates. Therefore, minimizing the number of signal transitions in circuits can be quite effective in reducing this component of power dissipation in these circuits. Another contributor to the dynamic power consumption in CMOS circuits is the short circuit current, which may also be reduced by minimizing the number of signal transitions. In deep sub-micron CMOS process technologies, the largest component of the leakage power in VLSI circuits is the drain-source leakage current of the transistors. This leakage current increases exponentially with the threshold voltage, which is in turn lowered with each new process technology generation in order to increase the gate drive strength.

Another reason why low power design has become important is the rapidly increasing temperature of VLSI chips, which makes the design of cooling systems more challenging and costly as well as decreases the reliability of circuits when combined with the effect of technology scaling [11]. Data path modules in a microprocessor are typically the hottest spots on a chip [10]. The high temperature of data path modules creates reliability problems and makes the cooling of the chip more difficult. The latter has a negative impact on the size and the cost of the system. Furthermore, the large thermal gradient created in the chip complicates the chip design and can create mechanical problems [15]. From the above discussion, the significance of decreasing the power consumption of the data path modules becomes evident.

This paper introduces several techniques for decreasing the power consumption targeting data path modules. Our techniques combine two well-known methods namely pre-computation for decreasing the dynamic power and guarding for decreasing the leakage current. Our main contributions are

1. Using guarding technique to decrease the dynamic power consumption. Therefore, one method can be used to decrease both dynamic and static power
2. Combining the pre-computation method with the guarding method
3. Introducing two new guarding techniques, namely, dynamic and hybrid methods, and finally
4. Presenting a method for reducing the switching activity of the sleep transistor that is used for the guarding purpose.

We first demonstrate the effectiveness of our methods using stand alone circuits. After that we show how our techniques can decrease the power consumption of the integer unit of a commercial VLIW processor by 81%.

The remainder of this paper is organized as follows. Section 2 describes precomputation and operand isolation methods for decreasing dynamic power, while Section 3 describes the guarding method for decreasing the leakage current of a circuit. In Section 4, our precomputation-based guarding is introduced. Section 5 describes the application of our method for comparators, adders and multipliers. In Section 6 two new methods for improving our guarding technique is introduced. Experimental results are presented in Section 7. Section 8 summarizes our findings and outline future work.

2. Dynamic power reduction by precomputation and operand isolation

Pre-computation is a well known technique for reducing the switching activity of a circuit. In this method, some inputs of a circuit are frozen while a smaller circuit computes the output values [7]. The idleness of a part of the circuit is dynamically detected in each clock cycle and the input registers are frozen by disabling their clock signals. This decreases the dynamic power consumption. This method has also been called signal gating [4]. Another method is guarded evaluation [14]. This method involves determining which parts of a circuit are computing useful results and which parts are computing results that are not used. Parts that are not needed are subsequently "shut off". We point out that the term “guarded” will be used in our paper in a different way compared to reference [14].

Precomputation increases the delay of the previous stage in a pipeline. Furthermore, it is effective only if all functional units driven by a register are unused in the current clock cycle. More precisely, if a register feeds a functional unit other than the idle ones, then freezing the inputs of the register will result in incorrect operation of the circuit. Figure 1 shows the integer unit of FR500, a commercial VLIW processor [12]. The integer unit consists of five different modules namely, **Add/Sub**, **Logic**, **Shift**, **Scan**, and **Set**. The **Scan** module finds the bit position of the last (most significant) bit of the REG that is equal to one; it is intended to be used in Huffman coding. The **Set** module determines the low (high) order 16 bits of its operand. The inputs of all modules are tied together and are driven by the same register set, REG. Depending on which operation is used, the output of one of the modules is selected by the multiplexor.

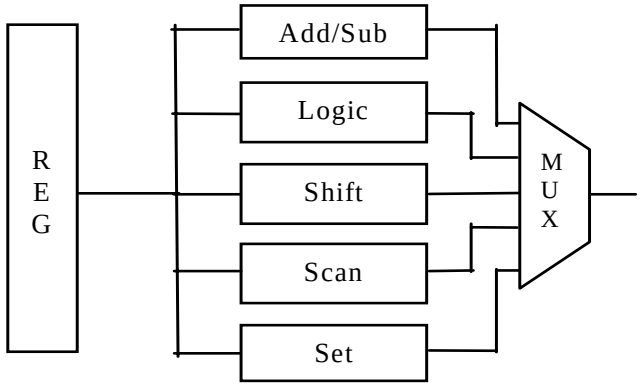

Figure 1. The integer unit of FR500 VLIW processor.

Even when the output of a module is not used, there may exist some switching activity in that module. Consider a clock cycle in which we are only interested in the output result of the **Add/Sub** module. Applying precomputation to this module will result in freezing the values of some bits of REG and will thereby reduce the switching activity in the **Add/Sub** as well as other modules. However, this also implies that there will be switching activity in the other four modules where ideally there should be none (because their outputs are not of interest in that cycle.) We refer to this problem as the *input sharing problem*. In some cases, it is possible to use multiple registers to drive different modules and thereby solve the input sharing problem, but this solution may not be practical if several registers drive the circuit since in such a case the overhead of duplicating all registers can be large. Furthermore, duplication of registers increases the load on the previous stages of a processor pipeline. Finally, the duplication method cannot be used if the inputs to the idle units come directly from another functional unit or a bus. Our technique overcomes the input sharing problem without such large overheads and increases the amount of power saving that can be achieved for such a circuit.

Operand isolation is another method for decreasing the switching activity in a circuit [8]. In this method, the inputs of the circuit are passed through an AND or an OR gate. This makes it possible to set the inputs to zero or one whenever the output of the circuit is not used and decrease the switching activity in the circuit. This method has a higher overhead than signal gating because of adding one extra gate for each input of the circuit. The overhead increases with the increase in the number of inputs. Another problem is that the added gates in the inputs increase the dynamic power consumption when the circuit is being used. On the other hand, unlike signal gating, operand isolation can be used in a circuit configuration such as the one depicted in Figure 1.

3. Leakage power reduction by power supply/ground gating

An effective method for leakage power reduction is power supply or ground gating. There are many ways in which this technique can be implemented, but the basic idea is to disconnect the power supply or ground of the idle units from those of the circuit so that these units do not consume any power. One possible implementation is to use Multiple-Threshold Voltage CMOS (MTCMOS) [6]. In this case, two high threshold transistors are connected in series with low threshold devices used inside the logic block creating two sleep transistors. This means that virtual supply and ground rails are created whose voltage levels are very close to the real supply and ground lines because of the small on-resistance of the sleep transistors. This method is also known as guarding. In practice, only one virtual rail (usually the virtual ground) is used. Normally, one sleep transistor per gate is used. Coarser granularities are also practiced, which require fewer transistors [13].

4. Guarding for decreasing dynamic power and precomputation-based guarding

Although the idea of using sleep transistors was originally developed for leakage power reduction in the standby mode, it can also be used to significantly reduce the dynamic power dissipation by decreasing the switching activity of the guarded circuit. This makes it possible to use the same technique for decreasing both dynamic and static power.

In our proposed method, we insert sleep transistors between the ground and virtual ground terminals of various parts of a circuit that need to be selectively turned off. If the target part (module) should be active in some clock cycles, then the enable signal turns the sleep transistor ON, permitting the normal operation. Otherwise, the sleep transistor disconnects the module from the ground. This eliminates all high to low transitions in the module and decreases the activity in the module outputs. The advantages of this method over the signal gating and operand isolation methods are as follows:

1. This technique reduces the leakage power dissipation of the idle units due to the cutoff of the signal paths to ground in the off mode and due to stack effect in the on mode [5]. Therefore, one technique can be used to decrease both dynamic and leakage power. In contrast, the precompuation and operand isolation cannot decrease the leakage current.

2. Unlike precomputation, this method can be used even when some logic blocks in the fanout of a register are

active (i.e., this technique does not suffer from the input sharing problem.)

3. Adding the sleep transistor may reduce the dynamic power even when the sleep transistor is on. The reason is that in many circuits some internal nodes switch more than once per clock cycle. Adding the sleep transistor increases the resistance between the internal nodes and the ground of the circuit. This reduces glitching in the internal nodes, thereby, decreasing the dynamic power consumption. Neither precomputation nor operand isolation has this feature. In fact the operand isolation method may increase the dynamic power consumption when the circuit is in its normal operation mode because of the extra gates added to the inputs (which increase the total switched capacitance of the circuit.)

When guarding a part of a circuit, its internal nodes and outputs may be floating. If a floating node is driving an unguarded gate, a conducting path between the supply and the ground may be formed. As a result, a large static current flows from the supply to the ground of the unguarded gate. It is important to ensure that this situation does not occur. Consider the NAND gate in Figure 2 (a). When A=1, B will be the controlling input of the NAND gate. If B is floating at a value between 0 and 1 logic levels (this could happen when a floating signal line loses some of the charge that is stored on its parasitic capacitances to other signal lines due to effects such as charge sharing or crosstalk coupling), both PMOS and NMOS transistors will conduct and static (rush-thru) current will flow between the supply and the ground. To correct this situation, the NAND gate can be guarded or the gate generating B can be unguarded. Figure 2 (b) shows another example. If the control signal C of the multiplexor is generated by a guarded gate, it may be floating at a value between 0 and 1 logic levels. Therefore, both pass gates may conduct at the same time. If A and B have different values, then a path between the supply and the ground will be formed through the drivers of A and B and the two pass gates. Thus, a static current will flow. To solve this problem, the gate generating C should be unguarded. Another option is to put two guarded buffers/invertors between inputs A and B and the pass gates. If the registers feeding the idle unit do not feed any other units, they can be guarded as well. The data stored in the registers will be lost, but this is not important in most data path modules as the registers are updated at every cycle. The enable signal for the sleep transistors, guarding the registers, may be different from that of the idle units because of the timing considerations. Guarding registers results in further power reduction. Adding the sleep transistor in series with gates reduces the speed of the high to low transition of the gate. To decrease the speed penalty, careful sizing of the sleep transistor is necessary. From this viewpoint, increasing the size of the sleep transistor is helpful as it reduces the delay penalty. However, there is a power dissipation overhead when switching the gate of the sleep transistor as well as an area overhead associated with using a large sleep transistor i.e., there is a tradeoff.

5. Precomputation-based guarding in arithmetic units

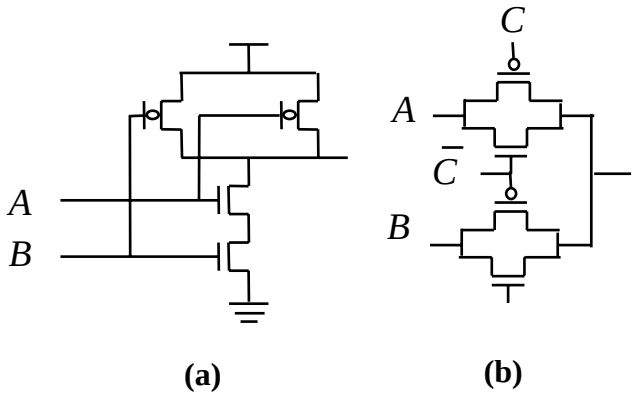

Figure 2. Two circuits that may consume static power after guarding.

5.1. Comparators

In this section, we use the precomputation-based guarding technique for a comparator, which compares two *n*-input numbers. Our approach is based on splitting the inputs into *m* bits in the most significant part (MSP) and *n-m* bits in the least significant part (LSP). As is well known, the idea is that if the MSP of two numbers are not equal, then the output of the comparator can be computed regardless of the value of LSP's. Otherwise, one must compare the LSP's to obtain the correct output of the comparator. To use this fact for reducing power consumption of the circuit, we ought to first split the parts of the circuit corresponding to MSP's and LSP's of its inputs and next disable the LSP of the circuit whenever the output can be determined from the MSP's of the inputs, i.e., when the MSP's are not equal. This is depicted in Figure 3.

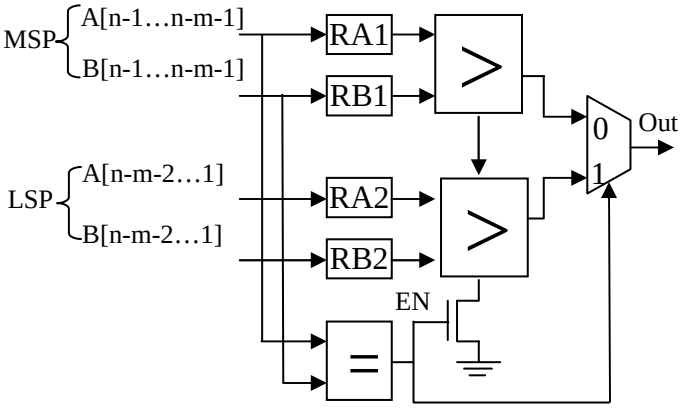

Figure 3. The m-bit precomputation-based guarding for an n-bit comparator.

As one can see in Figure 3, the enable signal *EN*, which controls the sleep transistor of the guarded circuit, also controls a multiplexor which selects the appropriate output of the circuit. As stated previously, the registers corresponding to LSP may also be guarded if they are not used to feed any other functional unit.

The amount of power that is saved depends on the value of *m*. As *m* increases, the probability for the two MSP's to be equal decreases. Consequently, the percentage of time in which the LSP of the circuit becomes active decreases and more power can be saved. On the other hand by increasing *m*, the

size of the circuit that checks the equality of MSP's (i.e., some XOR and AND gates) increases and the size of the circuit that is guarded becomes smaller, which in turn results in lower power saving. Therefore, there is a tradeoff. The optimum point to split the circuit and its inputs can be computed based on the correlation of the inputs, their probability distribution functions, and the technology used to implement the comparator.

5.2. Adders

One of the most prevalent arithmetic units in a digital circuit is the adder. When designing the functional units of a data path, the word length is determined based on the maximum dynamic range of the input data. However, actual data ranges, in most cases, are very small. A two's-complement number can be partitioned into two parts: *sign extension* bits (leading zeros or ones) and *significand* bits. We define the *effective bit range* (EBR) of an input operand as the bit width of the significand part. For the sign extension bits it is not necessary to perform expensive computation.

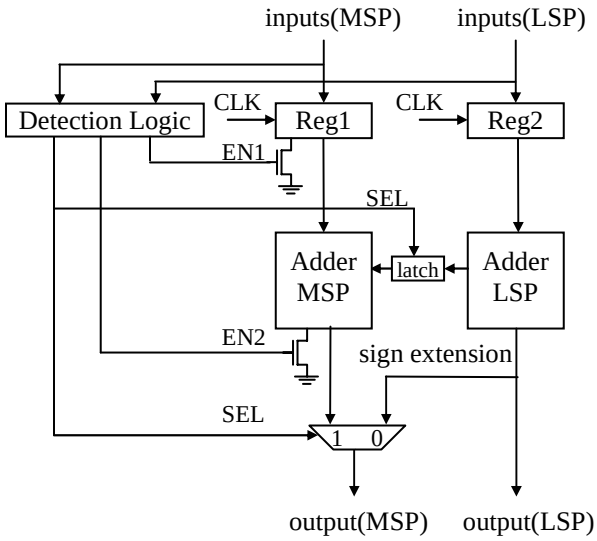

Figure 4. The precomputation-based guarding for an adder.

An effective method for precomputation-based guarding of an adder circuit is to divide the circuit into two parts, one part operating on the LSP's of the two operands while the other part operates on the MSP's and the carry propagated from the first part. This decomposition is done in a way that for most ADD operations performed by the adder, the EBR of the inputs is smaller than the LSP. A logic unit verifies whether the range of the significand bits exceeds the maximum range of the LSP. If this is the case, the functional unit performs computation in both the MSP and the LSP. Otherwise, the functional unit performs computation in only the LSP's of the inputs; the MSP of the circuit is disabled and simple sign-extension logic produces the correct output. Note that it is possible to make the precomputation more complex and detect more special cases. For example, one can devise a precomputation method which detects if one of the inputs is zero and disables the entire adder and sets the output of the circuit equal to the other input. A potential problem with such a method is that the precomputation condition tends to occur infrequently and the amount of power saved may become smaller than the power overhead of adding one or two extra multiplexors to the circuit.[1]

Figure 4 illustrates the implementation of a partially guarded circuit for a ripple carry adder. The inputs of the detection logic are MSP's of the operands and leading bits of LSP's of the two operands. Note that the most significant bits of the LSP's are needed since there may be an overflow when we add the two LSP's. For each input operand, the detection logic checks whether or not all bits in the MSP and the leading bit of the LSP are identically zero or one. If this is true, then the significand bits of both input operands do not exceed the LSP, which in turn implies that the output of the detection logic should be set to zero i.e., Reg1 and Adder MSP will be disabled and the adder will perform the operation with partial precision. Otherwise, this output should be set to one i.e., the adder will perform the operation with full precision.

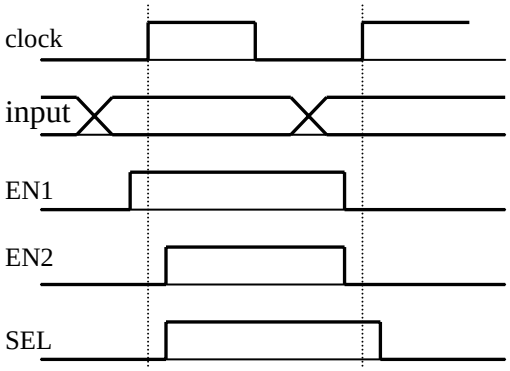

Figure 5. Timing diagram of the guarded adder circuit.

The inputs of the detection logic are connected to the inputs of the registers to have EN1 asserted before the next input data is loaded into the guarded registers, which is crucial for correct guarding. However SEL, which is connected to the sign extension logic and the latch that transfers the carry out from the LSP of the adder to the MSP, is asserted after the rising edge of the clock. This guarantees the output of the functional unit to be loaded correctly into the input register of the other functional units. The detection logic generates EN2 in such a way that the high to low transition of EN2 occurs before the rising edge of the clock to completely block the switching activity in the idle unit. Furthermore, the low to high transition occurs after the rising edge of the clock to prevent the switching activity in the idle unit at the end of the current clock cycle. Figure 5 shows the timing diagram of the circuit operation. In the above discussion, we used a ripple carry adder to illustrate the basic technique. It is straightforward to apply the same technique to other types of adders (carry

[1] Our current precomputation-based guarding technique for an adder can save power even in some cases when the above condition holds (i.e., when one of the inputs is zero and the other satisfies the BER test.)

bypass or look-ahead) by properly partitioning the input operand bits into the MSP and the LSP.

5.3. Multipliers

In this section we discuss the technique for an n-bit multiplier which usually is designed by using an array of single-bit carry save adders (CSA) followed by an n-bit carry propagation adder (CPA) as illustrated in Figure 6. To guard a multiplier, first it is checked whether the EBRs (Effective Bit Ranges) of the input operands are smaller than a predetermined value. If they are, a guarding logic will be activated, the multiplier will work as a low-precision multiplier and the result will be restored to the full-precision by sign extension. Otherwise, the multiplier will work as a normal full-width multiplier. For the multiplier architecture, it is possible to use a two-dimensional guarding technique. Two dimensional guarding inserts two guarding lines for the two input variables. Guarding may occur in one or both directions based on the precision of the input data.

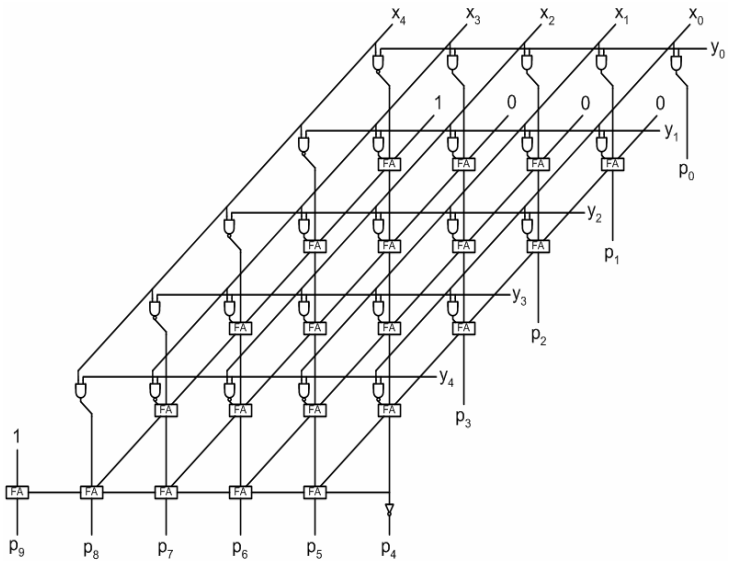

Figure 6. A multiplier using CSA array and one CPA.

The precisions of the inputs are detected by using the previously-described logic. The two guarding signals g_1 and g_2 are generated corresponding to $input_1$ (y_i) and $input_2$ (x_i), respectively (generated with a detection logic similar to the detection logic of adder circuit in Figure 4.) The CSA is partitioned into four regions (see Figure 7.) When g_1 is active, the left two regions are guarded. When g_2 is active, the bottom two portions are guarded. The right upper region is always unguarded and the left bottom region will be guarded if any of g_1 or g_2 is active. The inputs of the CPA are selected through the multiplexor MUX_1. If g_2 is inactive, the inputs will be the data from the last row of CSA. Otherwise, the inputs will come from the output of an intermediate set of adders (i.e., the output of the adders in the last row of upper region) of the CSA corresponding to the guarding line of $input_2$. In this case, the outputs of the CPA have to be shifted right by k bits where k is the number of precomputation bits for $input_2$. This shift is done by MUX_2. If g_1 is active, the left parts of MUX_1, CPA and MUX_2 are disabled. Finally, sign restoration is performed by MUX_3 using the sign bit (sb), which is easily done by computing the XOR of the most significant bits of the input data. In the above discussion, we used a carry save multiplier to illustrate the basic technique. The method can be applied to other types of multipliers e.g., Wallace tree by properly partitioning the input operand bits into the MSP and the LSP. Note that it is possible to use a precomputation scheme where the multiplier is turned off when one of the inputs is 0, $+2^p$, or -2^p, where p is some integer. The problem with such a scheme is that the precompuation condition does not happen very frequently. Therefore, the amount of power which can be saved is less than the power overhead of adding extra multiplexors in the output.

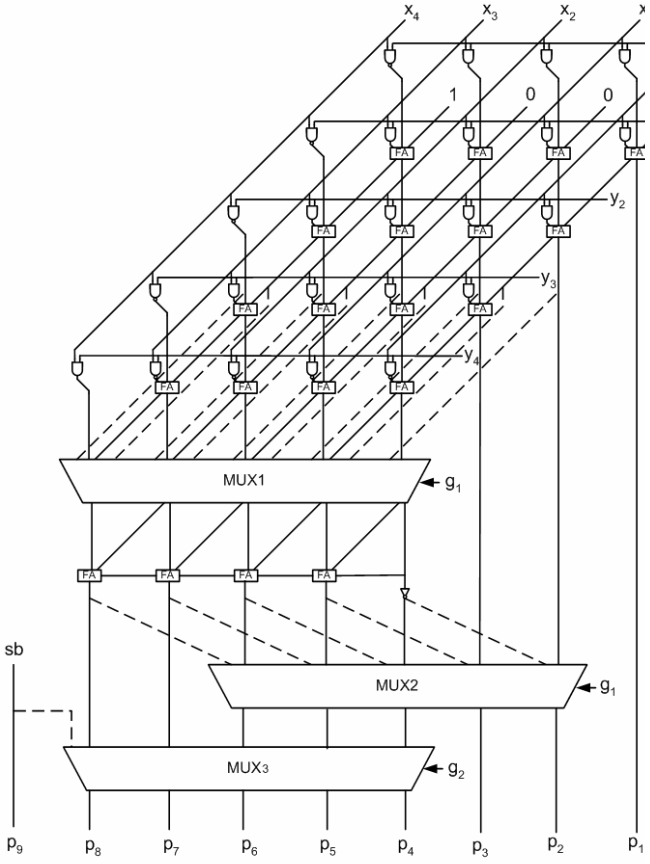

Figure 7. The precomputation-based guarding for a multiplier.

6. Improving the precomputation-based guarding technique

In this section two modifications to the basic technique are introduced to improve the power saving.

6.1. Dynamic and hybrid guarding

In the precomputation-based guarding technique which was described previously, the value for BER is pre-selected. If the BER of the actual data is larger than the pre-selected BER, the operation is performed as usual and no power is saved. On the other hand, if the BER of the actual data is much smaller than the pre-selected BER, some opportunity for power saving is lost. To improve the performance, a finer granularity method

can be used for the precomputation. In the dynamic guarding method, we detect the sign extension portion of the input data and deactivate the corresponding parts. Such dynamic guarding requires gating logic for every bit. In this case, the "right" portion of the logic is dynamically guarded, which results in the maximum power saving because the guarding signal is activated more often. This case should be contrasted with the static guarding where the guarding signal is activated only if the sign extension portion of the input data is larger than a fixed, pre-selected length (cf. Section 7.) In some cases, the dynamic guarding technique may result in an unacceptable delay, area and power dissipation overhead. For example, this method is ineffective for adders because the value of the sign bit depends on one of many internal carry signals of the adders. Which internal carry is used depends on the length of the sign extension portion of the inputs. This implies that a large multiplexor and a large number of wires are required. On the other hand, in multipliers, the sign extension bit is the XOR of the most significant bits of the input data which can easily be calculated. Therefore, using this technique for adders requires considerable amount of interconnect and consequently a large area overhead. However, this method is appropriate for guarding multipliers in the direction corresponding to $input_1$ (cf. Figure 7.) The reason is that signal g_1 only controls MUX_3, which selects the output between its inputs from the upper stages and signal sb, which as mentioned before can be easily computed. On the other hand, dynamic guarding for $input_2$ requires significant wiring overhead since MUX_1 would be required to select between several rows of CSA.

In cases that the overhead of the dynamic guarding is too large, a hybrid method can be used. In hybrid guarding, multiple values for BER are selected and depending on the input values, one guarding line is selected (see Section 7).

6.2. Reducing the switching activity of sleep transistors

Every time a sleep transistor is turned on and off, some dynamic power is consumed. If the guarding signal switches frequently, the power dissipation of switching the sleep transistor may overweigh the power saving due to the guarding. To address this problem, we use a circuit to limit the switching frequency of the sleep transistor. The proposed technique is based on generating a new guarding signal depending on the recent values of the original guarding signal. In this method the circuit is guarded if the guarding signal was active for a number of consecutive cycles as illustrated in Figure 8. In this figure, D units represent one cycle delay, which can be implemented by flip-flops. The circuit depicted in Figure 8 acts as a low pass filter and reduces the number of times a sleep transistor is switched. The number of delay units depends on the ratio of the delay overhead associated with switching the sleep transistor in comparison to the power saving in each clock cycle. In addition, it depends on the expected switching frequency of the original guarding signal. For example a more frequently-changing guarding signal requires more delay units in the circuit. The number of times a sleep transistor switches for a given trace and a given number of delay elements can easily be found by simulating the circuit for a given input trace. From simulation results, the percentage of time that the sleep transistor is off can be determined. This

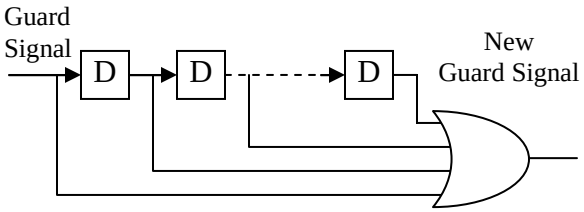

Figure 8. Limiting the switching frequency of the guarding signal.

value can then be used to estimate the power that is saved in the circuit and the power that is consumed to switch the sleep transistor for a given number of delay elements. From these estimations, a small set of candidate values for the number of delay elements can be determined. More accurate calculations can be performed by power simulating the circuit with different number of delay elements.

7. Results

We applied our techniques to several 32-bit functional units implemented in a 70nm technology with 0.9V supply voltage and 0.2V and -0.22V threshold voltage for NMOS and PMOS transistors, respectively [16]. The threshold voltage of the sleep transistors was 0.5V. We used PowerMill to estimate the power in gate-level and included the power of the added circuitry in all experiments. To compute the delay of circuits, we used SPICE. Figure 9 shows the percentage of the total power saving resulted from applying our guarding method on a 32-bit comparator for different number of bits in the MSP of the circuit. We drove the comparator using a trace of 1000 vectors extracted from a JPEG encoder. As one can see, it is possible to reduce the power by about 62% using our method.

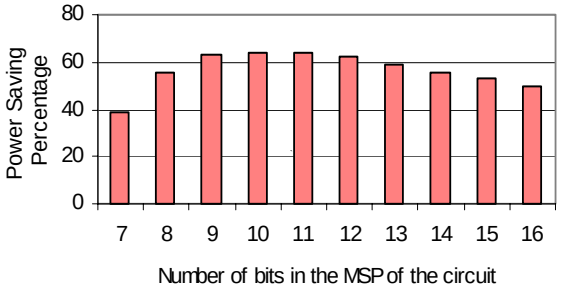

Figure 9. Power saving for a 32-bit comparator.

Furthermore, the saving is relatively insensitive to the number of bits used in precomputation. In our example, if the number of bits changes between 8 to 14 bits, the saving stays around 60%. The delay and the area overhead of using the sleep transistor for the comparator were 25% and 30%, respectively.

Next we applied our method to a 32-bit adder for two different numbers of bits in the MSP of the circuit and compared the results of our method with the results of the signal-gating [4] and the operand isolation method. For the operand isolation, we used AND gates in the input and used the precomputation logic to selectively freeze the inputs. Table 1 shows the percentage of power saving in the registers and the adder separately. Test data corresponds to a comparator and an

adder in the data-path of a processor executing a JPEG decoder program. Because of higher overhead of operand isolation than other methods its power saving is smaller. This is due to the dynamic power consumption of the added gates in the inputs.

Method	Regs	Adder	Total Saving
11-bit signal-gating	32%	21%	26%
11-bit operand-isolation	0%	17%	14%
11-bit guarding	41%	24%	37%
12-bit signal-gating	32%	25%	26%
12-bit operand-isolation	0%	21%	17%
12-bit guarding	44%	25%	38%
13-bit operand-isolation	0%	24%	14%
13-bit guarding	47%	27%	40%
14-bit signal-gating	34%	35%	27%
14-bit guarding	50%	28%	41%
15-bit signal-gating	34%	40%	27%
15-bit operand-isolation	0%	32%	15%
16-bit signal-gating	35%	45%	27%
16-bit operand-isolation	0%	35%	15%
16-bit guarding	55%	52%	48%
17-bit signal-gating	7%	46%	1%
17-bit operand-isolation	0%	35%	2%
17-bit guarding	11%	22%	3%
18-bit signal-gating	8%	49%	1%
18-bit operand-isolation	0%	36%	2%
18-bit guarding	12%	23%	3%

Table 1. Power saving for a 32-bit adder.

Table 2 shows the results of using the improvement methods described in Section 6 on a 32-bit adder. We used hybrid 18&11-bit precomputation method in which based on the input data, the portion of the adder corresponding to 18 or 11 most significant bits of operands are disabled. As one can see, the hybrid method significantly increases the amount of the power saving.

Method	Regs	Multiplier	Total Saving
22_bit_input1 16_bit_input2 gating	41%	58%	49%
22_bit_input1 16_bit_input2 guarding	43%	60%	52%
22_bit_input1 guarding input2_dynamic guarding	49%	67%	56%
22&16_bit_input1 hybrid guarding input2_dynamic guarding	51%	71%	60%

Table 2. Power saving for a 32-bit adder using the improved techniques.

Precomputation-based guarding performs better than signal gating. The reasons are that,

1- by adding the sleep transistor, the dynamic power consumption of both the adder and the registers in the normal mode are decreased. The reason is the decrease in the glitch of internal nodes.

2- The leakage current is reduced due to the use of the sleep transistor.

The saving is more than what can be achieved by signal gating or our non-hybrid method (see Table 2). Also, limiting the switching frequency of the guarding signal reduces the power overhead although the power saving may reduce also. However in most cases using this technique improves the overall power saving. We used two delay units for the 18-bit precomputation and results are shown in Table 2. As one can see this method can improve the result, but in this specific experiment the saving achieved was less than the hybrid method. Table 3 shows the results of using our improved methods on a 32-bit multiplier. As one see, using dynamic and hybrid guarding techniques improve the power saving in comparison to the vanilla precomputation-based guarding technique.

	Regs	Adder	Total Saving
18&11-bit hybrid_guarding	57%	59%	56%
18-bit guarding (reducing switching activity)	10%	19%	15%

Table 3. Power saving for a 32-bit multiplier using the improved technique

Table 4 includes delay and area overhead of our method for three different circuits. As one can see the amount of saving achieved for the multiplier is more than the adder.

Circuit	Guarding Method	Delay Overhead	Area Overhead
Comparator	10_bit guarding	25%	30%
Adder	18-bit guarding (reduced switching activity)	15%	10%
Multiplier	22&16_bit_input1 hybrid guarding input2_dynamic guarding	9%	6%

Table 4. Comparing the delay and the area overhead

Finally, we applied our techniques to the integer unit of FR500 VLIW processor depicted in Figure 1. The processor was implemented in a 0.18um technology with 1.8V supply voltage. We used SPICE to exhaustively simulate modules using some worst case or near-worst case input vectors with and without the sleep transistors and conservatively sized the sleep transistors. When sizing the sleep transistors, we took advantage of the timing slack of the faster modules to decrease the width of sleep transistors. We used an instruction set simulator to generate a trace for the integer unit. The power consumption of the unit under the trace was 1.012mW. For

Add/Sub module, we used the precompuation-based guarding method to disable a part of the circuit. For other modules, we used full guarding to disconnect the entire modules when they were not used to execute any instruction (i.e., we used one sleep transistor for each module.) The reasons for these choices were that additions and subtractions are used much more frequently than the other integer instructions [9]. This means that the percentage of the time the Add/Sub module can be fully turned off is relatively small. Therefore, the dynamic power overhead of turning the module on and off can be large. On the other hand, the inputs to the Add/Sub module are usually small positive or negative numbers, which makes the module a good candidate for applying the precomputation-based guarding technique. At the same time, the other modules are rarely used and, to the best of our knowledge, their inputs distribution does not have a well behaved statistical pattern which can be exploited for precomputation-based guarding. Therefore, we opted to use full guarding for these modules (i.e., we turned the entire module off if it was not used, and turned it on otherwise).

Based on the simulation results for the original module plus the extra circuitry, we were able to decrease the power consumption of the module by 81%. The power saving is mostly because of the reduction in dynamic power as the sub-threshold leakage in 0.18um technology is small. Unfortunately, we did not have FR500 processor design in technologies with smaller feature sizes. The area and delay overheads of adding the extra circuitry (i.e., the sleep transistors and the logic for controlling them) were 9% and 12%, respectively. This 12% delay overhead may not be serious with regard to the performance of the microprocessor as the critical path usually corresponds to the cache circuitry. Even if the Add/Sub module in Figure 1 is on the critical path, the other modules can be guarded without any delay penalty as they have some time slack. Note that only a part of the Add/Sub module is guarded. Therefore, for technologies that have high sub-threshold leakage, the leakage of the module will not be zero if the circuit is in the standby mode. This can be solved by adding one extra sleep transistor to the module to disconnect the low bits of the module from the ground whenever the circuit goes to the standby mode. We also power simulated the Add/Sub module when the sleep transistor was always on. The results showed an 8% reduction in the dynamic power. In other words, just adding a sleep transistor that is always turned on helps reduce the dynamic power consumption. We applied the operand isolation method to the modules of Figure 1. We used some AND gates to freeze the inputs of each module if its output was not used in the current cycle. The power reduction and the area overhead achieved by this method were 58% and 11%, respectively. Next, we combined the precomputation idea and the operand isolation to selectively freeze some inputs of the adder to see if it helps to improve the results. In this case, the power saving and the area overhead were 61% and 14%, respectively.

8. Conclusions

This paper described the idea of precomputation-based guarding. In this method precomputation is used to control a sleep transistor, which turns off parts of a circuit based on its input values. This reduces both the dynamic and the leakage power of the circuit while the circuit is working. Our current results are for modules used in the data-path of a processor. This approach results in a significant power saving in the case that many of the input vectors of the module are not at full bit-level precision. We are planning to evaluate our method for the comparators and adders used in other parts of a processor (e.g., memory-address calculator) and other data-path modules (e.g., divider).

9. Acknowledgment

We would like to thank Zhijun Huang form UCLA who kindly provided us with the trace of input vectors extracted from a JPEG decoder.

References

[1] A. Raghunathan and N. K. Jha, “Behavioral synthesis for low power,” *Proceedings of International Conference on Computer Desig*n, pp. 318-322, Oct. 1994.

[2] L. Benini, P. Vuillod, G. D. Micheli, and C. Coelho, “Synthesis of low power selectively-clock systems from high-level specification,” *Proceedings of International Symposium on System Synthesi*s, pp. 57-63, Nov. 1996.

[3] D. Kim and K. Choi, “Power conscious high level synthesis using loop folding,” *Proceedings of Design Automation Conferenc*e, pp. 441-445, 1997.

[4] Zhijun Huang; Ercegovac, M.D., “On signal-gating schemes for low-power adders”, Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on, 2001 Page(s): 867 -871 vol.1

[5] Johnson, M., Somasekhar, D. and Roy, K., "Leakage Control with Efficient Use of Transistor Stacks in Single Threshold CMOS ", Proceedings of the 36th Design Automation Conference (DAC), June 1999, pp. 442-445.

[6] Mutoh, S., Douskei, T., Matsuya, Y., Aoki, T., Shigematsu, S. and Yamada J., "1-V Power Supply High-Speed Digital Circuit Technology with Multi-threshold Voltage CMOS", IEEE Journal of Solid-state Circuits, pp. 847-854, August 1995.

[7] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power", IEEE Transactions on VLSI Systems, pp. 426-436, December 1994.

[8] M. Münch, B. Wurth, R. Mehra, J. Sproch, N. When, "Automating RT-level operand isolation to minimize power consumption in datapaths", Proceedings of the conference on Design, automation and test in Europe January 2000.

[9] Hennessy, Patterson, *Computer Architecture, A Quantitative Approach*, Second Edition, Morgan Kaufmann Publishers, 1996.

[10] J. D. Warnock, et al., “The Circuit and Physical Design of the Power4 Microprocessor”, IBM Journal of Research and Development, Volume 46, Number 1, 2002.

[11] Amir H. Ajami, Kaustav Banerjee, Amit Mehrotra, and Massoud Pedram, "Analysis of IR-Drop Scaling with Implications for Deep Submicron P/G Network Designs," *Proc. IEEE/ACM International Symposium on Quality Electronic Design (ISQED), San Jose, CA, March 2003*

[12] Takao Sukemura, "FR500 VLIW-architecutre High-performance Embedded Microprocessor", Fujitsu Scientific & Technical Journal, Vol.36, No.1, pp.31-38, June 2000.

[13] Mohab Anis, Mohamed Mahmoud, Mohamed I. Elmasry, Shawki Areibi, “Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique”, *Proceedings of Design Automation Conferenc*e, pp. 480-485, 2002.

[14] V. Tiwari, S. Malik and P. Ashar, "Guarded Evaluation: Pushing power management to the Logic Synthesis/Design Level", *International Symposium on Low Power Design*, 95.

[15] Amir H. Ajami, Kaustav Banerjee, and Massoud Pedram, "Non-Uniform Chip-Temperature Dependent Signal Integrity," *Proc. IEEE Symposium on VLSI Technology, Kyoto, Japan, June 2001*.

[16] Y. Cao, T. Sato, D. Sylvester, M. Orchansky, and C. Hu, "New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design," *Proc. IEEE CICC*, Orlando, FL, June 2000.