Transactions Briefs

Novel Low-Overhead Operand Isolation Techniques for Low-Power Datapath Synthesis

N. Banerjee, A. Raychowdhury, K. Roy, S. Bhunia, and H. Mahmoodi

***Abstract*—Power consumption in datapath modules due to redundant switching is an important design concern for high-performance applications. Operand isolation schemes that reduce this redundant switching incur considerable overhead in terms of delay, power, and area. This paper presents novel operand isolation techniques based on supply gating that reduce overheads associated with isolating circuitry. The proposed schemes also target leakage minimization and additional operand isolation at the internal logic of datapath to further reduce power consumption. We integrate the proposed techniques and power/delay models to develop a synthesis flow for low-power datapath synthesis. Simulation results show that the proposed operand isolation techniques achieve at least 40% reduction in power consumption compared to original circuit with minimal area overhead (5%) and delay penalty (0.15%).**

***Index Terms*—Low-power datapath synthesis, operand isolation.**

I. Introduction

Present-day circuits contain datapath modules which occasionally perform useful computations but spend a considerable amount of time in the idle states. However, the switching activity at their inputs in their idle states causes redundant computations which are not used by the downstream circuit. This redundant switching significantly increases the power consumption. Operand isolation is an effective technique to prevent unnecessary switching in a module by utilizing isolation circuitry at the input of the module. Enabling the isolation circuitry forces the modules in their idle states. Leakage power, however, becomes an important issue in this idle state of the module. Therefore, it should be ensured that the isolation circuitry is designed in a way that the isolated module consumes minimal leakage power as well.

Operand isolation was first introduced in IBM PowerPC 4xx-based controllers [7], where it was applied manually within a local boundary for isolation of multiplexer steered modules using the multiplexer select signal as a controlling signal. Precomputation-based methods have been applied to turn off sequential circuits based on their precomputed value from a certain number of input bits [9]. However, these methods not only require extra area to route the bits to the precomputation logic but also duplicate the target circuit for multi-output circuits. Moreover, for modules like adders and multipliers this method requires utilization of all the bits to compute the precomputation signal and, hence, the precomputation logic might consume area equivalent to preexisting logic in those modules. Tiwari *et al.* proposed an operand isolation methodology at the RT-level named "guarded evaluation" [8]. This method isolates a circuit by introducing transparent latches on the inputs to the arithmetic blocks and utilizes existing signals in the circuit as control signals for activating the latches. However, applicability of this technique is limited by the existence of such signals. Furthermore, it is difficult to implement logic for automatic selection of latches for large designs and there is also significant area overhead for placing latches in wide datapaths. Kapadia presented a technique for saving power dissipation in large datapath buses by preventing switching activity in bus driver modules [1]. In this scheme, insertion of extra latches to block transition activity was avoided by utilizing the enable signals of steering modules [registers, multiplexers (MUX), tri-state buffers] as isolating signals. However, this method is unable to provide optimal isolation in multiple fan-out steering modules. For instance, in case of a multiplexer-based isolation this method always has to select one of the two inputs for any switching activity in them even when the computation is redundant. This method also does not provide power savings in combinational blocks that are directly connected to primary inputs.

Munch *et al.* [2] addressed some previous limitations and presented a low overhead AND/OR-based isolation technique, which could also be applied to obtain power savings from blocks that are fed from primary inputs. However, leakage power reduction is not addressed in their isolation circuitry. In this paper, we make the following contributions:

- novel supply gating-based isolation circuitry that significantly lowers design overhead compared to existing implementations;
- applying operand isolation at circuit-level granularity to prevent redundant switching inside datapath modules;
- isolation techniques for cases where control signal gating is nonoptimal for operand isolation (e.g., multiplexer logic where one of the operands has to be chosen during computation);
- minimization of leakage power consumption in the idle state;
- integration of proposed isolation techniques with power/delay models to develop a complete synthesis flow at RT-level.

II. Novel Low-Overhead Isolation Circuitry

To reduce the overhead associated with operand isolation techniques (Section I), it is necessary to design low-overhead circuits such that the savings obtained through active power reduction is not offset by other factors (area, delay, leakage). We present a set of low overhead isolation techniques in this section.

A. Input-Isolating MUX (I^2-MUX)

MUX before datapath modules are common due to resource sharing of complex and large modules. MUX can be effectively utilized to prevent redundant switching on the datapath modules by isolation (blocking) of operands from inputs of these functional blocks.

If a conventional MUX is used for operand isolation, the MUX has to select the input that does not switch over a period of time [1]. This limits the possibility of isolation for MUX driven functional units, because the switching of an input operand of the MUX depends on the functional block generating it. Insertion of gating logic (such as latch and OR gates) at the interface from the MUX to a functional block provides more opportunities for operand isolation. That is because in this case the gating logic can prevent the switching at the input of the functional block irrespective of the state of the operands at the inputs of the MUX. However, the extra gating logic can have significant area, power, and delay penalty in the normal mode of operation.

We propose a new MUX circuit, called I^2-MUX, that provides a gating state in which none of the inputs are directed to its output and the state of the output is forced either to "0" or "1" (I^2F-MUX), or the output is held at its previous value before going to the gated mode (I^2H-MUX). The gated mode provides a new state for the I^2-MUX resulting in three states for a 2-to-1 multiplexer. The schematic of the

Manuscript received July 5, 2005; revised March 22, 2006.

The authors are with Purdue University, West Lafayette, IN 47906 USA (e-mail: nbanerje@purdue.edu; araycho@purdue.edu; kaushik@purdue.edu).

Digital Object Identifier 10.1109/TVLSI.2006.884054

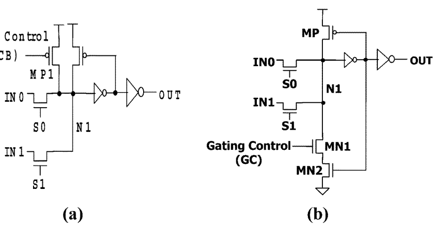

Fig. 1. (a) I^2F-MUX with output forced to 1. (b) I^2H-MUX with output hold.

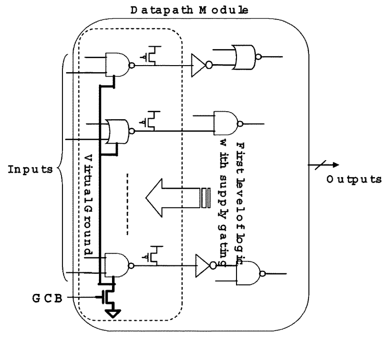

Fig. 2. FLS operand isolation scheme.

I^2-MUX is shown in Fig. 1. The inverter buffers of the conventional select control (decoder) are replaced by the NOR gates [Fig. 1(a)]. The second set of inputs of the added NOR gates are driven by the gating control (GC) signal. If the GC signal is low, the MUX operates similar to a conventional MUX.

If the GC signal is high, both select signals (S0, S1) are forced to zero irrespective of the state of the select input (SIN). This state is not allowed in a conventional MUX. However, in the I^2-MUX, this state isolates the output of the MUX from its inputs. If the GC signal is "1," S0 and S1 are both low and transistor MN1 is ON pulling the node N1 to low, forcing the state of the output of the MUX to "0." A minimum sized transistor for MN1 is large enough to be able to pull N1 to a low value in the gated mode. Therefore, the state of the output of the MUX is isolated from the operands (OP0 and OP1). This scheme does not need any extra control signals and also does not add any gating logic in the datapath. The only delay overhead can be due to a slight increase in the capacitance of the MUX internal node (N1) by the diffusion capacitance of the minimum sized transistor MN1, and replacement of the inverters in the select unit with the NOR gates [NOR0 and NOR1 in Fig. 1(a)]. This delay and power overhead in this method is definitely much smaller than the overhead due to insertion of extra gating logic on all the output bits of the MUX. Moreover, the added transistor (MN1) is a minimum sized transistor and the gating logic (NOR0 and NOR1) is in the select control path and shared by all bit slices of the MUX unit further minimizing the area and power overhead.

In the I^2F-MUX shown in Fig. 1(b), the state of the output of the MUX is forced to "0" in the gated mode. The I^2F-MUX can also be designed to force the state of its output to "1" [Fig. 2(a)] in the gated mode. In this case, the inverted gating control (GCB) signal is applied to a pull-up PMOS (MP1) to pull up the internal node (N1), and, therefore, the output (OUT) to "1" in the gated mode. Fig. 2(b) shows the I^2-MUX with hold capability at the output (I^2H-MUX). I^2H-MUX holds the state of its output in the gated mode. In the gated mode when GC is high (MN1 is ON), transistors MN2 and MP form an inverter holding the state of the internal node N1 by a cross-coupled inverter action. The added transistors (MN2, MN1, and MP1) are all minimum sized transistors, resulting in minimal area, power, and delay overhead.

Operand isolation using I^2F-MUX with output forced to a "0" or "1" value results in an extra switching on the functional block for switching from previous state to forced state (similar to isolation with AND/OR

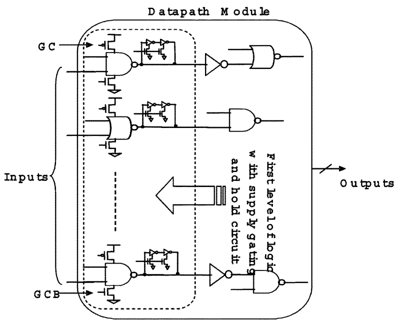

Fig. 3. FLH operand isolation scheme.

gates). Therefore, no energy savings is obtained if the gated mode is applied for only one clock cycle. The advantage of I^2H-MUX is that it blocks the input switching without forcing inputs to any particular state. Therefore, energy is saved even if it is applied for one clock cycle.

B. Using Supply Gating for Operand Isolation

For circuits where inputs of a datapath module are not provided through MUX or latches, extra masking logic (latch or AND/OR) is added for operand isolation [2]. This extra logic creates significant area, delay, and power penalties in the normal (nongated) mode. To reduce overhead, we propose using supply gating for operand isolation.

1) First Level Supply-Gating (FLS) Operand Isolation Scheme: FLS is originally proposed in [4], where only the first level logic is gated using transistors; this screens the rest of the combinational logic from the input transitions to provide operand isolation. In [4], FLS is used as a low-power scan test technique. Adding an extra transistor at only one logic level renders significant advantages with respect to area, delay and power overhead compared to previous methods, which use gating logic at each input.

Among the various FLS schemes, the gated-GND scheme proposed in [4] is most suitable for supply gating due to smaller area overhead and less delay and power penalties. In this paper, we have used the FLS strategy for operand isolation. Fig. 2 shows the proposed FLS based operand isolation technique applied to a general datapath module. For the implementation of the gating transistors, all first level gates share a single gating transistor through a virtual ground node. Sharing a supply gating transistor reduces area overhead because a shared supply gating transistor can have less size than the sum of sizes of all supply gating transistors in the unshared case.

2) First Level Hold (FLH) Operand Isolation Scheme: Similar to OR/AND based isolation techniques, FLS-based operand isolation cannot prevent one redundant switching at the input of the datapath module when switching to the gated mode. In the FLS scheme, the states of outputs of first level gates are forced to "0" or "1." Applying a fixed state causes a redundant switching on every transition to the gated mode due to which there will not be any energy savings if the isolation is applied for a period of only one clock cycle. In this section, we develop an operand isolation technique based on supply gating that prevents extra switching by holding the state of first level gates in the supply gated mode. To implement this technique, output of the first level gates needs to be held at their initial values when applying this method. This can be achieved by adding a latch element (cross-coupled inverters) at the output node. The latch element needs to be enabled only in the gated mode to hold the output state of the first level gate. This scheme is called FLH and is used in [5] for low-power delay fault testing as an alternative to the enhanced scan-based delay fault testing, with significantly less design overhead. Fig. 3 shows the above method applied to a general circuit for isolation. In this scheme, the sharing of supply gating transistors is not possible because the outputs of first level gates may store different values.

TABLE I
COMPARISONS OF AREA OVERHEAD OF OPERAND ISOLATION TECHNIQUES

Datapath module	(1) Datapath module preceded by MUX						(2) Datapath module not preceded by MUX					
	Conventional		Proposed				Conventional		Proposed			
			I^2F-MUX		I^2H-MUX				FLS		FLH	
	OR	Latch	Area	%improv. over OR	Area	%improv. over latch	OR	Latch	Area	%improv. over OR	Area	%improv. over latch
Comparator	25.8	47.3	1.9	92.6	2.8	94.0	45.2	82.9	18.7	58.6	56.2	32.2
Adder	12.6	23.1	0.9	92.8	1.4	93.9	16.0	29.3	13.3	16.9	40.1	-36.8
Multiplier	0.4	0.8	0.01	97.5	0.01	98.7	0.4	0.8	3.1	-675.0	9.2	-1050.0

TABLE II
COMPARISONS OF DELAY OVERHEAD OF OPERAND ISOLATION TECHNIQUES

Datapath module	(1) Datapath module preceded by MUX						(2) Datapath module not preceded by MUX					
	Conventional		Proposed				Conventional		Proposed			
			I^2F-MUX		I^2H-MUX				FLS		FLH	
	OR	Latch	Delay	%reduction from OR	Delay	%reduction from latch	OR	Latch	Delay	%reduction form OR	Delay	%reduction from latch
Comparator	4.1	2.4	1.2	71.5	1.4	41.8	3.8	2.1	0.4	89.3	0.8	60.6
Adder	2.4	2.0	0.4	82.1	1.1	47.3	2.2	1.7	0.0	100.0	0.0	102.7
Multiplier	1.4	1.0	0.2	82.6	0.7	25.6	1.9	0.9	0.3	86.9	0.0	97.2

TABLE III
COMPARISONS OF POWER OVERHEAD (NORMAL MODE) OF OPERAND ISOLATION TECHNIQUES

Datapath module	(1) Datapath module preceded by MUX						(2) Datapath module not preceded by MUX					
	Conventional		Proposed				Conventional		Proposed			
			I^2F-MUX		I^2H-MUX				FLS		FLH	
	OR	Latch	Pow.	%reduction from OR	Pow.	%reduction from latch	OR	Latch	Pow.	%reduction from OR	Pow.	%reduction from latch
Comparator	41.9	52.0	3.0	92.7	0.6	98.9	117.2	201.5	3.3	97.2	38.6	80.8
Adder	16.8	29.6	3.5	79.3	0.2	99.3	44.1	71.0	1.2	97.3	32.8	53.8
Multiplier	27.8	28.2	0.4	98.6	0.1	99.6	6.1	6.6	-1.7	127.5	3.9	40.0

III. COMPARISON WITH EXISTING ISOLATION METHODS

To estimate the effectiveness of proposed operand isolation schemes, we simulated a set of datapath benchmark circuits using BPTM 70-nm models (to observe sub-100-nm effects) and obtained power and performance in normal mode of operations and area overhead due to operand isolation circuit. The gate-level netlists were first technology-mapped to a LEDA 0.25-μm standard cell library[1] using Synopsys design compiler with the mapping effort at medium. The benchmark circuits are then translated to Hspice and scaled to 70 nm. Power is measured in NanoSim by applying 200 random vectors to the inputs and delay is measured by Hspice simulation of the critical paths of a circuit.

We consider two scenarios for our comparisons: 1) if datapath modules are preceded by MUX, the conventional OR and latch (inserted between multiplexer and datapath module)-based operand isolation techniques are compared with the proposed I^2-MUX technique (Fig. 1) and 2) if datapath modules are not preceded by MUX, the conventional OR and latch (inserted at the inputs of datapath modules) techniques are compared with the proposed FLS and FLH operand isolation schemes (Figs. 2 and 3). Tables I–III show the results of comparisons of the various techniques (area, delay, power).

Table I compares these techniques in terms of area overhead [70-nm transistor active area ($W \times L$)]. The proposed I^2F-MUX technique exhibits the smallest area overhead for all datapath circuits. It shows 92.8% reduction in area overhead as compared to the existing OR-based gating technique, which has the least area penalty among the conventional techniques. I^2H-MUX also shows significant reduction in area overhead compared to the latch-based gating. It can be noted that, for the OR- or latch-based method, area overhead is proportional to the number of inputs of the datapath module. However, in FLS, gating logic is inserted in all first level gates (Fig. 2), the number of which depends on the number of first level gates of the datapath module. Therefore, for a datapath module with a large number of first level gates, such as multiplier, there will be additional area overhead when implementing operand isolation utilizing FLS/FLH schemes [Table I(b)].

[1]Leda Design Inc. [Online.] Available: http://www.leda-design.com.

Table II shows comparative impact of the existing and proposed operand isolation techniques on circuit delay for different benchmarks. It is observed that the OR-based gating has the largest increase in delay. Compared to the OR-based gating, I^2F-MUX exhibits delay overhead reduction of up to 82%. I^2H-MUX exhibits delay overhead reduction of up to 47%, when compared to the latch-based gating. As observed from Table II, the delay overhead of the FLS technique is less than 0.4% for all the benchmark circuits. Compared to the OR and latch-based gating, FLS and FLH techniques exhibit significant delay overhead reduction.

Table III compares power consumption for various implementations in normal mode of operation. The I^2-MUX techniques have considerably less power overhead ($> 90\%$) compared to conventional techniques. The power dissipation of the FLS circuits is very close to the power dissipation of the original combinational circuit without any gating technique because in FLS the gating transistor and the pull-up PMOS do not switch in the active (normal) mode. Interestingly, for large benchmark circuits such as multiplier, power of the FLS circuit is even less than the power of the original circuit (negative overhead or gain) because the gating transistor results in leakage reduction (due to stacking effect [3]) for the idle gates. This leakage is called *active leakage* since it occurs in the active mode for the idle gates and it is a significant part of the overall active power in the 70-nm technology

TABLE IV
COMPARISONS OF LEAKAGE POWER (μW) FOR DIFFERENT OPERAND ISOLATION SCHEMES IN GATED MODE

Datapath module	(1) Datapath module preceded by MUX						(2) Datapath module not preceded by MUX					
	Conventional		Proposed				Conventional		Proposed			
			I^{2}F-MUX (out='0')		Mixed I^{2}F-MUX				FLS (Gated GND)		Mixed FLS	
	OR	Mixed AND/ OR		%reduction from OR		%reduction from mixed AND/OR	OR	Mixed AND/ OR		%reduction from OR		%reduction from mixed AND/OR
Comparator	46.7	45.4	18.7	60.0	13.9	69.4	9.1	7.8	6.4	30.0	5.0	35.4
Adder	58.1	56.9	33.8	41.8	24.8	56.5	20.5	19.3	18.7	8.6	13.8	28.3
Multiplier	648.5	638.9	574.0	11.5	401.8	37.1	611	601.2	560.0	8.3	555.1	7.7

node. Reducing the active leakage on the first level gates can result in overall power reduction for large circuits. FLS shows overall power reduction of up to 127% compared to the OR-based technique.

IV. LEAKAGE REDUCTION IN OPERAND-ISOLATED MODE

In this section, we explain how our operand isolation techniques can be used for significant savings of active leakage power compared to existing implementations. In the gated mode, the functional block does not switch; however, it still dissipates power due to standby leakage which becomes significant in scaled technologies [3]. Leakage of a combinational circuit is a strong function of the state of its inputs [6]. Therefore, by selecting the best input vector for a combinational circuit in standby mode, its leakage power can be significantly reduced. In this section, we show that leakage reduction is an additional advantage of the I^{2}F-MUX and FLS operand isolation techniques on top of the benefits in terms of area, delay, and power.

A. Mixed I^{2}F-MUX Operand Isolation Scheme

By selective use of I^{2}F-MUX with output forced to "0" and I^{2}F-MUX with output forced to "1" [Fig. 1(a)] on individual inputs, the best input vector for leakage minimization in the gated mode can be applied to the datapath modules that are preceded by MUX. We refer to this isolation method as mixed I^{2}F-MUX. If the leakage energy dissipation during the gated (standby) mode is larger than the energy associated with one extra switching associated with the use of I^{2}F-MUX, it would make sense to apply mixed I^{2}F-MUX to save leakage by applying the best vector to the functional block in the gated mode. The decision whether to use I^{2}H-MUX or mixed I^{2}F-MUX with output forced to the best vector depends on the relative magnitude of leakage power with respect to the switching component of power and also the cycle time. It is worth noting that the longer the cycle time, the larger the ratio of leakage to the switching power.

B. Mixed FLS Operand Isolation Scheme

The OR-based operand isolation technique fixes the state of all inputs to "1" in the isolated mode, which might not be the best input vector that minimizes overall leakage power for the module. For latch-based operand isolation, the state of the inputs cannot be set to the best vector since the inputs are fixed at their right state before going to the gated mode. However, AND and OR gating together can provide the best input vector for the datapath module by OR masking the inputs that are to be at logic state of "1," and AND masking the inputs that are to be at the logic state of "0." However, even though the mixed AND-OR system forces the inputs to be in minimum-leakage states, the blocking gates (AND, OR) themselves dissipate considerable leakage power.

In the proposed FLS operand isolation scheme, the outputs of all first level gates are forced to logic level "1" or "0," respectively. However, this state of inputs may not correspond to the best input vector for minimum leakage. By selective use of gated-GND or gated-VDD [4] for individual inputs, the state of the datapath module can be assigned to the best input vector during operand isolation to minimize leakage. This scheme is called the mixed FLS operand isolation scheme.

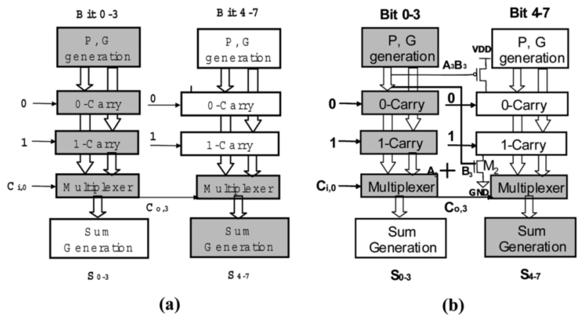

Fig. 4. (a) 8-bit CSA. (b) Reduced redundant switching by supply gating.

C. Results and Comparison of Power in Gated Mode

The results of leakage reduction by input vector control using mixed OR/AND, mixed I^{2}F-MUX, and mixed FLS for different benchmark circuits are shown in Table IV. The best input vectors are found using algorithms described in [6]. Depending on the benchmark, significant savings can be achieved by applying the best input vector using mixed I^{2}F-MUX (module preceded by multiplexer) and mixed FLS (module not preceded by multiplexer). The mixed I^{2}F-MUX operand isolation technique shows improvements of 69%, 56%, and 37% in leakage power compared to the mixed AND/OR-based technique for the benchmarks. The mixed FLS technique shows improvements of 35%, 28%, and 7.7% in leakage power compared to the mixed AND/OR-based technique. The FLS technique eliminates the extra gating logic circuits (AND/OR) and also reduces the leakage of first level gates due to the stacking effect [3], thus, improving the power dissipation. Due to the exponential increase of leakage with technology scaling and temperature increase, the leakage reductions of the mixed I^{2}F-MUX and mixed FLS become more effective as the technology scales or the temperature increases.

V. OPERAND ISOLATION AT BIT-LEVEL

Besides block level isolation techniques, it is possible to apply certain bit-level isolation techniques to achieve further power reduction, even while circuit is doing useful computation. In this section, we introduce a novel methodology for reducing redundant switching in datapath modules by efficient supply/GND gating at the bit-level, even when they are performing useful computations for downstream circuits.

A. Operand Isolation for Carry-Save Adder Circuit

To demonstrate the reduction of redundant switching in carry select adders (CSA) by selective gating, let us consider an 8-bit CSA which has been split up into two 4-bit ripple carry full adders [Fig. 4(a)]. The topmost block is the propagate (P) and carry generator (G) blocks. The critical path of the circuit has been shaded. When both A_3 and B_3 are "1," carry propagated to the second stage will always be "1" and

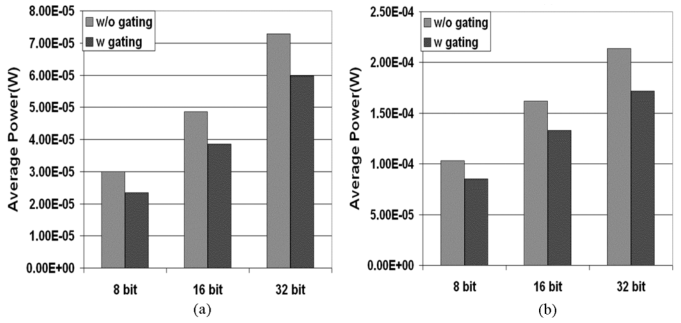

Fig. 5. Average power of 8-, 16-, and 32-bit carry select adders with and without bit-level gating at clock frequencies (a) 100 and (b) 500 MHz.

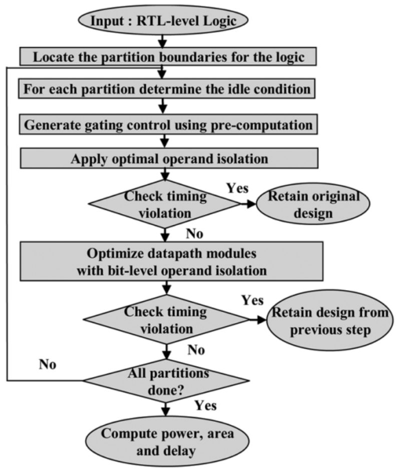
Input : RTL-level Logic
Locate the partition boundaries for the logic
For each partition determine the idle condition
Generate gating control using pre-computation
Apply optimal operand isolation
Check timing violation
Yes
Retain original design
No
Optimize datapath modules with bit-level operand isolation
Check timing violation
Yes
Retain design from previous step
No
No
All partitions done?
Yes
Compute power, area and delay

Fig. 6. Overall synthesis methodology.

switching in the 0-carry block for the bits 4 to 7 is redundant. Similarly, if both A_3 and B_3 are "0" then switching in the 1-carry block is redundant. To eliminate this redundant switching, we can use NMOS GND gating of the 1-carry block (using $P_3 = A_3 + B_3$ as the gating signal) and PMOS supply gating of the 1-carry block by $G_3 = A_3 \cdot B_3$. When G_3 = "1," the transistor M_1 is turned off thereby eliminating switching in the 0-carry block. Similarly, when P_3 is "0," it turns off M_2 and eliminates switching in the 1-carry block. If we consider that all the bits can be 0 or 1 with equal probability then, this technique can remove redundant switching in the second stage (bits 4–7) in 50% of the cases. It should be noted that the same technique can be used to supply/GND gate the stage 1 (bits 0–3) with the gating control being the input carry in $C_{i,0}$. Simulations were carried out on 8-, 16-, and 32-bit carry select adders at three different frequencies (Fig. 5) and results show an average power reduction of 20%. It can be noted that the supply/GND gating transistors of the second and the subsequent stages are added in the noncritical path of the circuit.

Hence, if supply/GND gating is not used in the first stage, then there is no performance penalty in our proposed technique. However, if supply/GND gating technique is used in the first stage of the circuit too (corresponding to the simulation results in Table IV), then there is an overall delay increase of approximately 2%.

VI. Integrated Synthesis Flow

We have developed a synthesis methodology for integrating the application of I^2-MUX-based, FLS, and FLH-based and bit-level operand isolation techniques at RT-level. The complete design flow for insertion of isolation circuitry is shown in Fig. 6. First, we partition the RTL-level circuit into modules based on sequential boundaries and perform isolation on the combinational logic bounded by sequential logic or those connected to the primary inputs. Our assumption, in this case, is that logic circuits across sequential boundaries do not affect each other. The idle condition for the outputs of each partition is then determined. In the next step, we generate the gating control signals using precomputation logic. We then identify isolation condition and the best isolating candidate for each circuit formed by partitioning. While applying the gating signals, the inputs are classified in two categories—1) shared and 2) nonshared. Nonshared inputs are those inputs that are not shared by more than one logic block. Therefore, isolation can be performed on such inputs without affecting the functionality of other blocks. Shared inputs, on the other hand, are simultaneously shared by more than one logic block. Therefore, while isolating these inputs, attention should be taken so that the functionalities of the other blocks sharing that input are not affected.

Since identification of optimal operand isolation candidates is critical for maximum power savings, we choose an optimal isolation candidate as follows. First, we determine if flip-flops or latches are available for isolation and apply clock gating or control signal gating to them. If latches are not available, we perform isolation by I^2-MUX in cases where a multiplexer precedes the datapath. If switching probability of the multiplexer output is very high (e.g., it switches every clock cycle), we use an I^2H-MUX to hold the state of the circuit. Otherwise, we use an I^2F-MUX to force it to the minimum leakage state. In absence of MUX or latches, our algorithm locates tri-state buffers (e.g., buses have tri-state buffers) and applies control signal gating to the enable signals of the buffers to prevent unnecessary propagation of switching. In case none of these steering modules are available and the logic is connected to primary inputs, we apply the FLS or FLH method to isolate them depending on their switching probability as in the case of I^2-MUX. After the insertion of isolation circuitry, we estimate whether the timing constraint is violated. If the timing constraint for the module is violated, we retain the original nonisolated circuit. Otherwise, we apply isolation. The next step involves optimization of datapath modules with bit-level supply gating for further power reduction. The timing constraint of the respective modules is verified again after applying bit-level operand isolation and the outcome determines whether this optimization is performed on the modules or the design obtained from the previous state is retained. The additional area and the power reduction of the optimized modules (by either operand isolation alone or bit-level isolation only or both) are computed in the final step.

We applied the optimal isolation synthesis flow to two benchmarks. 1) A pipelined complex multiplier with a multiplexer as isolation candidate, choosing either adder or multiplier at any time for valid computation. Power savings in this case is 40% with negligible area (0.95%) and delay penalty (0.2%). 2) An ALU core where FLS is used for isolation. We obtain 50% power savings with minimal area (13%) and delay (0.1%) overhead

VII. Conclusion

We have presented novel operand isolation circuits (at block and bit level) that provide more power savings with significantly lower design overhead compared to the existing isolation schemes. We have developed an integrated synthesis methodology to automate the application of the proposed operand isolation techniques at the RT-level.

References

[1] H. Kapadia *et al.*, "Reducing switching activity on datapath buses with control-signal gating," *IEEE J. Solid State Circuits*, vol. 34, no. 3, pp. 405–414, Mar. 1999.
[2] M. Munch *et al.*, "Automating RT-level operand isolation to minimize power consumption in datapaths," in *Proc. DATE*, 2000, pp. 624–631.
[3] K. Roy *et al.*, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proc. IEEE*, vol. 91, no. 2, pp. 305–327, Feb. 2003.

[4] S. Bhunia *et al.*, “A novel low-power scan design technique using supply gating,” in *Proc. ICCD*, 2004, pp. 60–65.
[5] S. Bhunia *et al.*, “First level hold: A novel low-overhead delay fault testing technique,” in *Proc. DFT*, 2004, pp. 314–315.
[6] M. C. Johnson *et al.*, “Models and algorithms for bounds on leakage in CMOS circuits,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, no. 6, pp. 714–725, Jun. 1999.
[7] A. Correale, “Overview of the power minimization techniques in the IBM powerPC 4xx embedded controllers,” in *Proc. ISLPED*, 1995, pp. 75–80.
[8] V. Tiwari *et al.*, “Guarded evaluation: Pushing power management to logic synthesis/design,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 17, no. 10, pp. 1051–1060, Oct. 1999.
[9] M. Alidina *et al.*, “Precomputation-based sequential logic optimization for low power,” in *Proc. ICCD*, 1994, pp. 74–81.

Hybrid-Scheduling for Reduced Energy Consumption in High-Performance Processors

Madhavi Valluri, Lizy John, and Heather Hanson

***Abstract*—This paper develops a technique that uniquely combines the advantages of compile-time static scheduling and hardware dynamic scheduling to reduce energy consumption in dynamically scheduled processors. In this hybrid-scheduling paradigm, regions of the application containing large amounts of parallelism visible at compile-time bypass the dynamic scheduling hardware and execute in a low-power static mode. Experiments on several media and scientific benchmarks demonstrate that the proposed scheme can provide significant reduction in energy consumption with negligible performance degradation.**

***Index Terms*—Dynamic scheduling, instruction-level parallelism, out-of-order issue processors, static scheduling.**

I. Introduction

Modern high-performance microprocessors are typically centered around a sophisticated out-of-order execution core that can issue multiple parallel instructions each cycle. Due to its highly associative nature, the issue logic is one of the most power-hungry units on the processor core today [5]. The energy consumption of this dynamic-scheduling hardware accounts for nearly 30% of the overall energy in existing processors [2], [4], [5], and is projected to grow further with increasing issue widths and window sizes [17]. Several research efforts in the recent past have focused on reducing the energy consumption of the dynamic scheduling logic [1], [3], [4], [6], [7], [11]–[14]. A notably large number of the previously proposed schemes are microarchitectural-level solutions. This paper proposes a different approach: a cooperative hardware/software approach. The advantage of engaging the compiler is that the compiler has accurate information regarding the region to be executed in the near future and, hence, can apply energy saving resource adaptations at finer granularities when compared to runtime hardware techniques [7], [13], [15]. Further, since we have the opportunity to shift some of the burden to the compiler, there are new avenues to simplify the hardware beyond what is achievable with traditional hardware-only schemes.

For regular and well-structured programs such as media and scientific programs, the compiler can generate efficient (almost optimal) schedules for considerable portions of the code. Dynamic issue hardware is useful in extracting parallelism in irregular programs where the compiler is unable to find parallelism due to a lack of sufficient information at compile-time. However, in processors with dynamic scheduling logic, the hardware searches for parallel instructions, *irrespective* of whether the compiler-generated schedule is perfect or not. Hence, in regular applications or regions, the dependence analysis and scheduling performed and, consequently, the energy expended by the dynamic scheduling hardware is unnecessary. The technique presented in this paper attempts to eliminate this redundancy. The proposed scheme combines the advantages of both compile-time static scheduling and runtime dynamic scheduling to lower the energy consumption in an out-of-order issue processor.

In this hybrid-scheduling paradigm, regions of code containing large amounts of parallelism that can be identified and exploited at compile-time bypass the out-of-order issue logic and are issued exactly in the order prescribed by the compiler. The processor runs in the superscalar mode with dynamic scheduling until a special instruction that indicates the beginning of a statically scheduled (S-Region) is detected, at which point, the out-of-order issue engine is shut down, and the processor switches to a VLIW-like *static mode* in which instruction packets scheduled by the compiler are issued sequentially in consecutive cycles with minimal hardware support. High-performance general-purpose systems cater to diverse application domains. The hybrid-scheduling architecture, thus, allows use of aggressive and power-hungry scheduling hardware for applications that warrant it such as integer applications, while facilitating low-energy execution for structured applications such as media and floating-point applications, that do not require such hardware.

II. Hybrid-Scheduling Microarchitecture

The hybrid-scheduling microarchitecture extends the features of a generic superscalar processor with different low energy structures to support static mode issue. These structures include a small buffer to hold S-Regions and a low-energy mechanism to support precise exceptions for instruction execution in the static mode.

A high-level view of the hybrid-scheduling microarchitecture is shown in Fig. 1. The program begins execution in the normal superscalar fashion, i.e., decoded instructions are dispatched to the instruction window where they wait for their operands, ready instructions are issued to the functional units and, finally, instructions retire in-order from the reorder buffer. Execution continues in the superscalar mode until the beginning of an S-Region is detected. S-Region instructions are scheduled by the compiler into groups or *packets* of independent instructions that can be issued in their statically-scheduled order to reduce power. The compiler annotates S-Regions with special “start-static” and “end-static” instructions to indicate the beginning and the end of an S-Region.

All instructions following the “start-static” instruction, i.e., S-Region instructions, are stored in the S-Buffer, a circular buffer where each line holds one instruction packet. The packet size is fixed and is determined at design time. The maximum packet size is the number of

Manuscript received November 17, 2003; revised May 9, 2004, July 28, 2004, March 21, 2005, and November 23, 2005. This work was supported in part by the National Science Foundation under Grant 0429806 and by the Defense Advanced Research Projects Agency under Contract F33615-03-C-4106. This research was conducted while M. Valluri was a Ph.D. student at The University of Texas at Austin.

M. Valluri is with the Systems and Technology Group, IBM Corporation, Austin, TX 78712 USA (e-mail: mvalluri@us.ibm.com).

L. John and H. Hanson are with the Electrical and Computer Engineering Department, The University of Texas at Austin, Austin, TX 78712 USA.

Digital Object Identifier 10.1109/TVLSI.2006.884055