

Improved Clock-Gating through Transparent Pipelining

Hans M. Jacobson

IBM T.J. Watson Research Center, Yorktown, NY. {hansj@us.ibm.com}

ABSTRACT

This paper re-examines the well established clocking principles of pipelines. It is observed that clock gating techniques that have long been assumed optimal in reality produce a significant amount of redundant clock pulses. The paper presents a new theory for optimal clocking of synchronous pipelines, presents practical implementations and evaluates the clock power benefits on a multiply/add-accumulate unit design. Transistor level simulations show that dynamic clock power dissipation can be reduced by 40-60% at pipeline utilization factors between 20-60%, on top of traditional stage-level clock gating, without affecting pipeline latency or throughput.

Categories and Subject Descriptors

C.1.3 [Processor Architectures]: Other Architecture Styles - Pipeline processors.

General Terms

Design, Performance.

Keywords

Optimal pipeline clocking, Transparent pipeline, Pipeline stage unification, Adaptive pipeline depth, Dynamic pipeline scaling, Clock gating, Low power, High performance, Microarchitecture, Circuits.

1. INTRODUCTION

Clock power is a key design constraint in modern VLSI design. Despite increases in leakage power, clock power remains a significant part of the total power dissipation in modern microprocessors [1]. Clock gating has shown to be an efficient technique to significantly reduce dynamic power dissipation [3, 5, 8, 2]. However, despite fine grained clock gating, power consumed by the clock remains a major contributor to overall chip power dissipation.

The work presented in this paper re-examines the fundamental clocking principles of pipelines. Our work shows that traditional clock gating techniques that have long been thought to gate the clock optimally produce a significant amount of clock pulses that are redundant to the correct operation of the pipeline. Our observations of requirements for correct pipeline operation arrive at a novel and practical clocking solution that can significantly reduce clock power by relaxing the clocking requirements of pipelines.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISLPED'04, August 9–11, 2004, Newport Beach, California, USA.
Copyright 2004 ACM 1-58113-929-2/04/0008 ...\$5.00.

Related work

The goal of the clocking technique presented in this paper is to reduce the clock power in traditional synchronous pipelines. This is achieved by reducing the number of clock pulses required to propagate a data item through the pipeline. In related work, several techniques have targeted clock power reduction by reducing the number of clock pulses generated to the latch stages in the pipeline.

Clock gating [3, 5, 8, 2] has been used to reduce dynamic clock power through all levels of the design hierarchy from the top-most chip level down to the individual latch level. While these techniques reduce the number of clock pulses generated in the pipeline, they all have one thing in common: To avoid data races through the latches of adjacent pipeline stages, each latch stage is clocked at least once for each data item propagating through the pipeline.

Collapsible pipelining techniques have been presented to reduce clock power in pipelines [6, 7, 4]. During runtime, data latches in collapsed pipeline stages are made “permanently” transparent for a duration of time. The collapsing technique subsequently reduces the number of latches that have to be clocked. These novel techniques can significantly reduce clock power, but are also limited in two ways. First, when collapsing pipeline stages, the logic depth of each collapsed pipeline stage is doubled (or more depending on how many stages in sequence are collapsed). This causes the pipeline operation frequency to be cut in half (or more). Pipeline collapsing techniques subsequently trade frequency for power which may cause a significant drop in performance. Second, while the pipeline can be collapsed for a duration of time during runtime, this collapsing is static in nature and affects throughput for the whole pipeline. Because of fundamental limitations of these collapsing techniques, they cannot be applied dynamically on a cycle-by-cycle basis to save clock power.

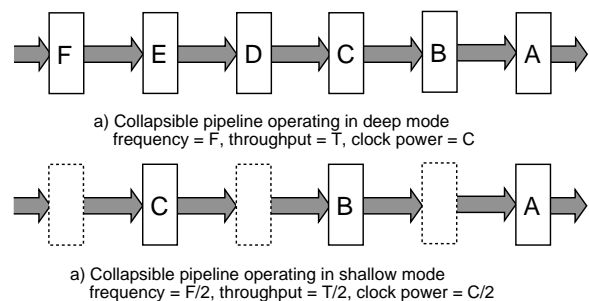


Figure 1: Prior work: a static collapsible pipeline.

Figure 1 illustrates the concept of a static collapsible pipeline. In deep mode all latches operate in opaque mode and the pipeline can run at full frequency. In shallow mode, every other latch stage is made transparent, collapsing two pipeline stages into one. In shallow mode, the pipeline runs at half frequency and only every other latch stage is clocked.

Contributions

To summarize, the common limitation of previous work in the area of clock gating is that, for a given operation frequency (that may change during runtime), all latch stages required to be active in order to run at that frequency have to be clocked at least once for each data item passing through the pipeline.

In contrast, the work presented in this paper can avoid clocking active latch stages by dynamically adapting to the current state of the pipeline, on a cycle-by-cycle basis, *without* reducing the operation frequency or throughput of the pipeline. Our technique is based on the observation that a latch stage only needs to go opaque in order to separate closely spaced data items in a pipeline. By keeping latches transparent by default, our technique allows data items that are sufficiently separated in time (clock cycles) to propagate through the pipeline without generating any clock pulses. Such separation occur frequently in, for example, microprocessors due to pipeline stalls caused by data dependencies.

The remainder of this paper first discusses the theoretical clocking requirements of pipelines in Section 2. Section 3 presents transparent pipelines at a conceptual level. Practical realizations for control logic and clock blocks are presented in Section 4. Results are presented in Section 5 and conclusions are given in Section 6.

Assumptions

Throughout this paper we assume timing constraints that are standard for static transparent latch based pipelines. The long path and short path delays through a pipeline are allowed to be arbitrary and standard latch setup and hold times apply. No new timing constraints are introduced by our technique and the technique works with any type of transparent latches.

The correct operation of a transparent pipeline, as presented in this paper, places two behavioral constraints on its input and output environment. First, a transparent pipeline requires that the value of a data item is held stable at the environment input until a next subsequent valid data item arrives. Second, a transparent pipeline requires that the output environment only latches data indicated valid by the transparent pipeline. Simply put, the environment input and output stages need to be clock gated at the stage level in traditional opaque fashion.

2. PIPELINE CLOCKING RE-EXAMINED

In traditional pipeline implementations, a fundamental assumption has been that latch stages of a pipeline must be held opaque by default in order to avoid data races between the latch stages. Such data races may cause data residing in an upstream stage to accidentally overwrite the data in a downstream stage due to differences in the depth of logic paths between latches. By keeping latches opaque by default, data races through latches are avoided. However, this invariably results in a pessimistic clocking model that produces clock pulses that are redundant to the functionality of the pipeline.

The following sections propose a new technique for clocking synchronous pipelines. To eliminate redundant clocking of a pipeline, we first introduce the notion of a *transparent pipeline* in which all latches are transparent by default. Second, we develop a new clocking model in which latches are made opaque (clocked) only when a *true* data race is present in the pipeline. The proposed technique can significantly reduce the number of clock pulses required to propagate a data item through a pipeline.

Pipeline correctness criteria

A true data race exists only if two data items propagate through a pipeline without any opaque latch stage to separate them. Assum-

ing arbitrary min/max delay of logic paths through combinational logic and latches, the following criteria must be met to ensure correct operation of a transparent pipeline.

- Requirements to avoid data races:
 - For each pair of distinct adjacent data items (A,B) propagating through a pipeline, where A is downstream of B, at least one opaque latch stage must separate A from B (and B from A in case of a circular pipeline).
 - For each data item A propagating through a circular pipeline, at least one opaque latch stage must separate A from the tail of A.

Given these requirements to avoid data races, the criteria required to implement optimum clocking of a transparent pipeline can be derived. In this context the concept of a *state holder* is introduced. A state holder for a data item *A* is the opaque latch stage holding the most recent value of *A* stable. This is typically the opaque latch stage closest upstream of the current position of *A*.

- Criteria for optimum clocking of a pipeline:
 - For each pair of adjacent data items (A,B) propagating through a pipeline, where A is downstream of B, the latch stage for A is clocked only when B overwrites the current state holder for A, and
 - For each data item A propagating through a circular pipeline, only one stage for A is clocked, and only once each iteration.

In non-linear pipelines, such as forks and joins, *A* can have multiple state holders each holding a part of the value of *A*. If any of these state holders are overwritten, a new state holder must be provided for *A* at its current location in the pipeline.

The consequence of the stated clocking criteria for a transparent pipeline is that a latch stage only needs to be clocked in order to *separate* a pair of data items moving concurrently through the pipeline. This is in contrast to the traditional clocking criteria of an opaque pipeline that states that a latch stage needs to be clocked in order to *propagate* a data item moving through the pipeline. The first criteria is the significantly more relaxed of the two and allows for a tangible reduction in required clock pulses.

3. TRANSPARENT PIPELINE: CONCEPT

A transparent pipeline keeps its latch stages transparent by default. This default state represents the transparent clock gated mode of the latch stage (transparent mode). Data races between latches are avoided by physically separating each pair of data items concurrently propagating through the transparent pipeline. A pair of data items are separated by forcing a latch stage residing between the pair to enter an opaque state. This opaque state can be either the opaquely clock gated mode of the latch stage (opaque mode), or the normal clocking of the latch stage (clocked mode). A latch stage in a generalized transparent pipeline can thus operate in three different modes.

Figure 2 illustrate the behavior of a five stage linear pipeline. The three middle latch stages, 2, 3, and 4 form a transparent pipeline segment. These latch stages operate in transparent mode by default. Latch stages 1 and 5 form the input and output environment of the transparent pipeline. These latch stages operate in traditional opaque mode by default. A valid latch is associated with each stage to keep track of the location of valid data in the pipeline. The valid latches are clocked each clock cycle. In the figure, dotted lines indicate that the latch is clock gated in transparent mode. Solid lines

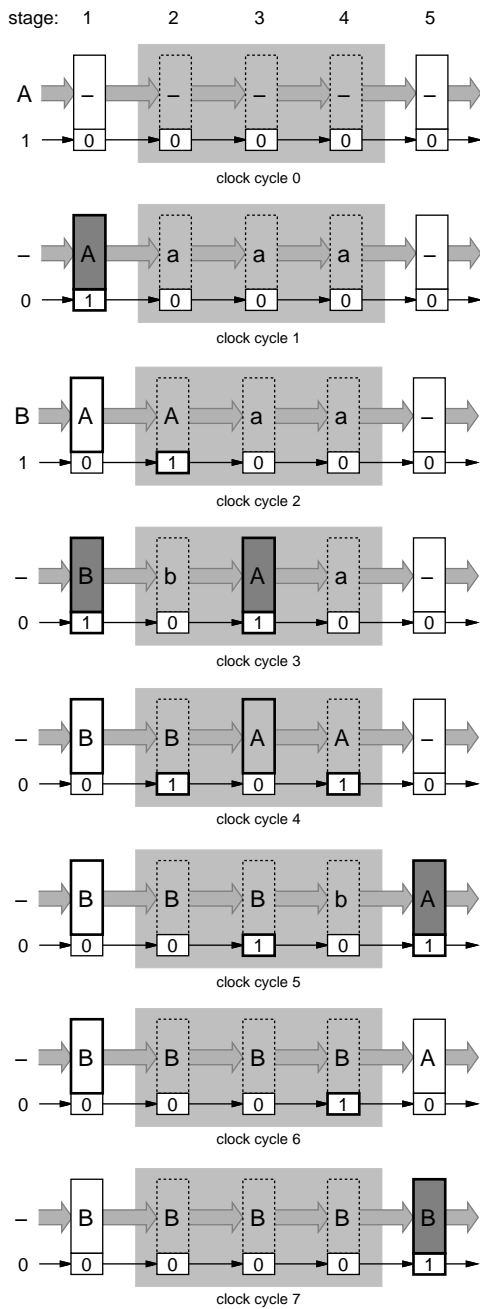
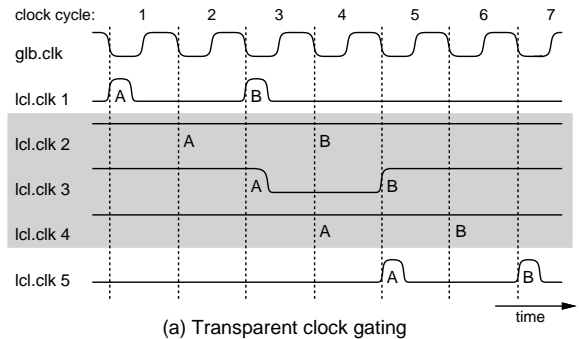


Figure 2: A linear three-stage Transparent Pipeline

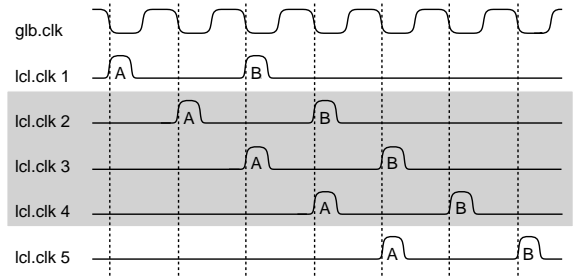
indicate that a latch is clock gated in opaque mode (or is clocked the current cycle). Thick lines indicate that the latch is the current state holder for the data item marked inside it, i.e., the latch is responsible for holding the value for that data item stable. A dark shadow indicates that the latch in question captures a data item, and also represents the completion of one cycle of its local clock.

Consider a stream of data items $A, -, B, -, \dots$ entering the pipeline, where $-$ represents the absence of valid data (a bubble). Assume the pipeline is initially empty. Stages 1 and 5 are then clock gated in opaque mode, and stages 2, 3, and 4 are clock gated in transparent mode.

Clock cycle 1: As data item A enters stage 1, at clock cycle 1, it is captured and held stable in the data latches. Stage 1 is now



(a) Transparent clock gating



(b) Traditional opaque clock gating

Figure 3: Clock waveforms corresponding to Figure 2.

the state holder for A . Since the data latches for stages 2, 3, and 4 are clock gated in transparent mode A can now propagate freely through this transparent segment of the pipeline. In the figure, the short path propagation of A is indicated in lower case a . The long path propagation of A is indicated in upper case A . Assuming arbitrary delay on short paths through the logic the data inputs to the latches of stage 5 can change at any time. Since stage 5 is clock gated in opaque mode, these unstable values are not latched. Subsequently, there is no risk for metastability to occur. At the end of clock cycle 1, the longest path through the logic in stage 1 has completed and the output of the stage is now valid.

Clock cycle 2: At the start of clock cycle 2, the associated valid bit is captured by the valid latch in stage 2 to indicate the new position of A (note that the valid latch is always clocked). Note, however, that since no valid data item immediately follows A in the pipeline, stage 1 continues to hold A stable. There is subsequently no need to clock the data latches of stage 2. The data latches of stage 2 therefore remain clock gated in transparent mode.

Clock cycle 3: In clock cycle 3, stage 1 latches data item B . Since stage 1 no longer holds data item A stable, stage 3, where A currently resides, must capture and hold A stable. The data latches for stage 3 are therefore clocked this cycle and are thereafter held in opaque gated mode. Stage 1 is now the state holder for B , while stage 3 is the state holder for A . As stage 2 is transparent, B can propagate freely through stages 1 and 2 during this clock cycle. The short path propagation of B is indicated with lower case b . Since the data latches for stage 3 are opaque, b cannot propagate further than stage 2, and there is subsequently no risk for data races between data items A and B . The valid latches are updated to indicate that B resides in stage 1 and A resides in stage 3.

Clock cycle 4: During clock cycle 4, no data latches have to be clocked as stage 1 continues to hold B stable and stage 3 continues to hold A stable. The valid latches are updated to indicate that B now resides in stage 2 and A resides in stage 4.

Clock cycle 5: In clock cycle 5, the data latches for stage 3 are forced transparent to let B propagate through and thereafter remain

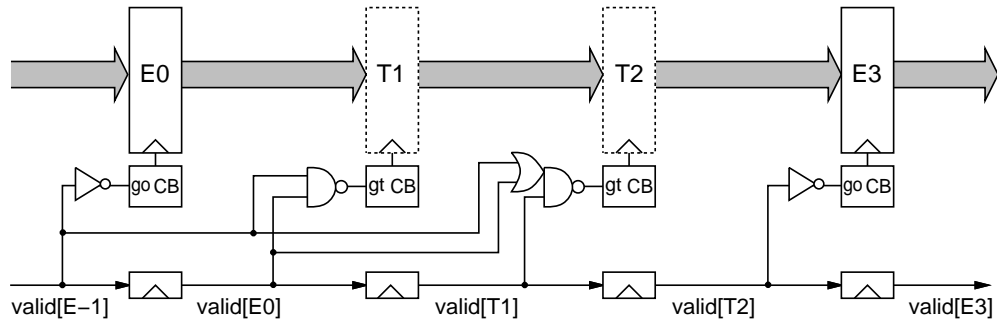


Figure 4: Control logic implementation for a two-stage transparent pipeline.

in transparent mode. At the same time, the valid bit associated with A indicates to stage 5 of the output environment that the output of the transparent pipeline is now valid. Stage 5 is subsequently clocked and captures A . The valid latches are updated to indicate that B resides in stage 3 and A resides in stage 5.

Clock cycle 6: During clock cycle 6, no data latches need to be clocked as stage 1 continues to hold B stable. The valid latches are updated to indicate that B now resides in stage 4.

Clock cycle 7: At the start of clock cycle 7, the valid bit associated with B indicates to stage 5 of the output environment that the output of the transparent pipeline is again valid. Stage 5 is subsequently clocked and captures B . The valid latches are updated to indicate that B now resides in stage 5.

Figure 3 illustrates the clock waveforms generated for the example illustrated in Figure 2. Figure 3(a) illustrates the clock waveforms when stages 2, 3, and 4 make use of transparent mode clock gating. Figure 3(b) illustrates the clock waveforms when stages 2, 3, and 4 make use of traditional opaque mode clock gating. As can be readily observed from Figure 3(a), the three-stage transparent segment of the pipeline generates the equivalent of only *one* clock pulse in order to let data items A and B propagate through it. In comparison, the traditionally opaque clock gated pipeline in Figure 3(b) has to generate a total of *six* clock pulses to propagate A and B through it. The example clearly illustrates the potential benefit of transparent mode clock gating in pipelines with moderate utilization factors.

4. TRANSPARENT PIPELINE: IMPLEMENTATION

There are many ways to implement a transparent pipeline. To better focus the presentation on the main concepts of transparent pipelining, a simple and efficient special case implementation for a two-stage non-stallable transparent pipeline is given in this paper. The presented technique can be readily extended to cover general N -stage stallable pipelines as well.

The special case implementation covered in this section is possible when the transparent pipeline segment has two stages or less. In this case, the behavior of the opaque clock gated mode equals the behavior of the clocked mode. The number of operation modes that a latch stage has to support can thus be reduced from three down to two, providing a solution very similar to traditionally clock gated pipelines. The two operation modes that need to be supported are (1) normally clocked mode, and (2) transparent mode clock gating.

4.1 Control logic

A pipeline stage that is clock gated in transparent mode has to be able to detect whether it should switch to clocked mode or stay transparent. Clocked mode only has to be entered in order to sepa-

rate two data items propagating concurrently through the transparent segment of the pipeline. For a given clock cycle, any transparent pipeline stage should enter a clocked operation mode only under the following conditions:

- A transparent pipeline stage T should be clocked only if:
 1. valid data is present at the input of T , and
 2. valid data is present at the input of any transparent stage upstream of T , or valid data is present at the input of the environment input stage.

Whenever the stated conditions do not hold, the transparent pipeline stage should operate in a transparent clock gated mode. Given this simple condition, it is possible to provide a straightforward implementation. The first condition can be implemented by observing the valid bit feeding into stage T . From the second condition, it is clear that a look-behind function is needed to detect whether there is another data item upstream of stage T . This look-behind function can be implemented by observing the valid bits of the upstream pipeline stages.

The clock gating conditions for the two transparent stages T_1 and T_2 , with the input environment E_0 and output environment E_3 , then become:

$$\begin{aligned}
 gate_{E_0} &= \text{not}(valid_{E_{-1}}) \\
 gate_{T_1} &= \text{not}(valid_{E_{-1}} \text{ and } valid_{E_0}) \\
 gate_{T_2} &= \text{not}(valid_{T_1} \text{ and } (valid_{E_{-1}} \text{ or } valid_{E_0})) \\
 gate_{E_3} &= \text{not}(valid_{T_2})
 \end{aligned}$$

Figure 4 illustrates the necessary control logic for an implementation of a synchronous pipeline with two transparent stages.

4.2 Clock block

A clock block (CB) supporting transparent mode clock gating in a two stage transparent pipeline is straightforward to implement. Figure 5 illustrates a transparent and an opaque mode clock block for a two-phase clocked master/slave pipeline. The transparent mode clock block contains one master and one slave latch internally. These latches are used to latch the clock gating signal to prevent glitches on the clock. Since both the master and slave data latches (not shown) are gated in transparent mode, the master and slave clock signals both need to be gated in their high state (logic 1). In traditional opaque clock gating the gating signal for the master and slave clocks can both be taken from the internal master latch. However, when clock gating in transparent mode, the clock polarity at the input to the clock gating point for the slave clock, gate gs in Figure 5, makes gate gs sensitive to glitches on the clock gating input while the internal master latch is transparent. The gating

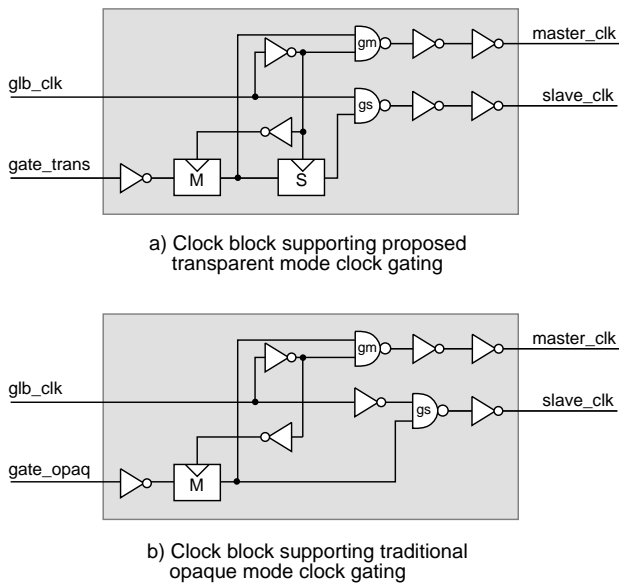


Figure 5: Two-phase clock block implementations.

signal for the slave clock must therefore be taken from the internal slave latch instead.

5. RESULTS

5.1 Power contributors

The power savings achievable in a transparent pipeline depend on two factors. First, power is saved by reducing the number of clock pulses that are generated in the pipeline. This is due to keeping latches transparent so that data items that are sufficiently separated in time can propagate through the pipeline without requiring the latch stages to be clocked. Second, however, additional power is consumed as an effect of increased glitching on data signals. Opaque latches act as barriers and prevent glitches on data signals to propagate down the pipeline. In transparent segments of the pipeline, latches are held transparent. Glitches can therefore propagate down the pipeline and cause extra switching of wires and transistors. The power savings achievable in transparent pipelines is subsequently a tradeoff between how much power is saved by reducing the number of clock pulses that have to be generated versus how much additional power is consumed by additional glitching on data signals.

5.2 Evaluation results

For a technique that adapts dynamically to the current utilization of the pipeline, it is important to evaluate the amount of saved clock power over a range of pipeline utilization factors. Introduced glitch power depends on both pipeline utilization and data switching factors. Glitch power therefore has to be evaluated over a range of both these factors. It is important to estimate worst case bounds for the introduced glitch power in order to determine the practical applicability of transparent pipelines. The design chosen for evaluation purposes is therefore based on logic with a high glitch tendency.

The transparent pipeline techniques were evaluated on a high frequency Multiply/Add-Accumulate (MAAC) unit illustrated in Figure 6. This type of design was chosen since add and multiply functions are based heavily on XOR-type logic that has a high glitch tendency. The unit implements a 32x32 fix-point Booth encoded multiplier with final adder. The unit features a bypass path to allow

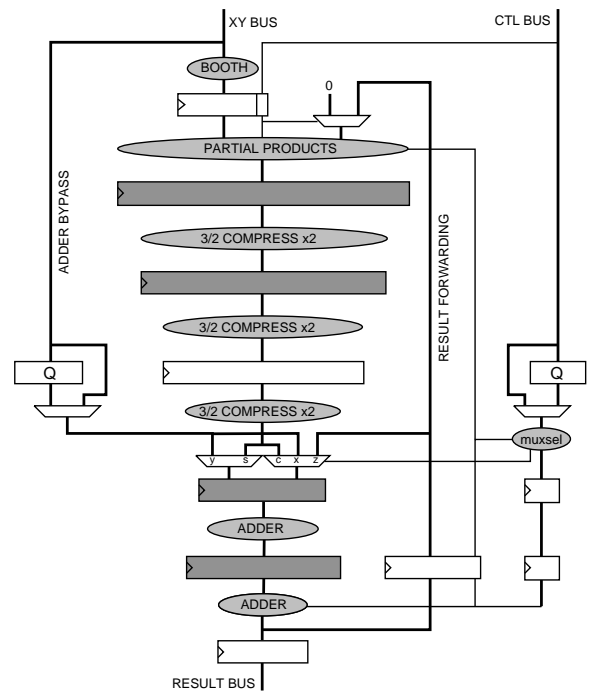


Figure 6: Transparently pipelined MAAC unit.

add instructions to enter the final adder directly without having to pass through the multiply stages. A forwarding path is provided to the multiplier to allow multiply-accumulate instructions. The unit was implemented as a seven stage pipeline in a 0.13 micron technology with a target frequency of 3 GHz.

Two implementations of the unit are evaluated. The first design implements traditional opaque stage level clock gating of all pipeline stages. The second design implements stages 2, 3, 5, and 6 in a transparent clock gated mode (shadowed latch stages in Figure 6). The unit was simulated at the transistor level under a range of pseudo randomly generated input vectors for the data inputs and valid inputs. The valid signals indicate the utilization of the pipeline. The simulation includes all datapath logic, control logic, latches, clock gating logic, clock blocks, and clock buffers.

The graph in Figure 7 illustrates the clock power of the transparently clock gated pipeline stages as compared to the same pipeline stages clock gated in traditional opaque mode over a pipeline utilization (valid switch factor) range of 0-50%. The average clock power saving over the given utilization range is 52%. The relative clock power saving peaks at 61% at a pipeline utilization of 20%.

The graph in Figure 8 illustrates the absolute clock power savings of the transparently clock gated pipeline stages over a pipeline utilization range of 0-100%. The two curves in Figure 8 illustrate the transparent clock power savings when the data input switching factor is at 0% and at 100% respectively. The difference between the curves illustrates the range between best and worst case glitching power introduced as a result of keeping latches transparent. From this graph it can be observed that the introduced glitch power is never more than 10% of the clock power savings. This is an important result as it shows that transparent pipelining techniques can be applied advantageously even in situations where the data logic has a high glitch tendency. Figure 8 also clearly illustrates the range of pipeline utilization for which transparent pipelining is most effective. The transparent pipeline operates most efficiently in the utilization range between 20-60% where the clock power dissipa-

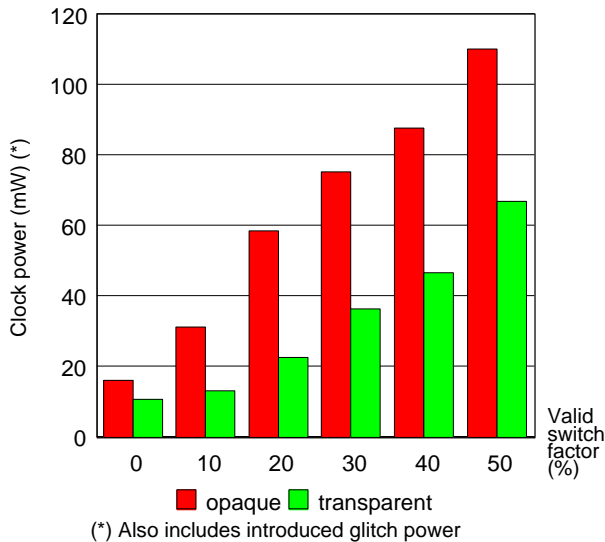


Figure 7: Clock power dissipation for MAAC unit over a pipeline utilization range of 0-50% (opaque = traditional opaque stage-level clock gating, transparent = proposed transparent stage-level clock gating).

tion is reduced by 40-60%. The absolute power saving peaks at a pipeline utilization of 50%.

Another important thing to note in Figure 8 is that as the pipeline utilization factor increases, the introduced data glitch power decreases as more latch stages are opaque and therefore act as glitch barriers. As can be observed in the figure, the transparent pipeline therefore always performs as good as, or better, than the opaque clock gated pipeline in terms of power.

Lastly, it is important to keep in mind that since the presented transparent pipelining technique adapts dynamically to the current utilization state of the pipeline, on a cycle-by-cycle basis, latency, throughput, and IPC of the pipeline are not affected. The power saving benefits of transparent pipelining are achieved without any performance degradation. The only limitation in the application of transparent pipelining techniques is in keeping the delay on the clock gating signals within the target cycle time. This limits the practical length of a transparent pipeline segment. In high frequency pipelines this limit is typically between three to five pipeline stages and depends greatly on signal arrival times, latch count, and wire distribution.

6. CONCLUSIONS

This paper has re-examined the clocking principles of traditional synchronous pipelines. It has been shown that clock activity can be significantly reduced, on top of traditional stage level clock gating, through use of transparent pipelining techniques. The paper presents a new theory for optimal clocking of synchronous pipelines and outlines the conceptual operation of such optimally clocked pipelines. Practical circuit implementations for control logic and clock blocks are presented and the clock power benefits are evaluated on a high frequency multiply/add-accumulate unit design. Transistor level simulations of datapath, control logic, and clock blocks show that dynamic clock power can be reduced by 40-60% at pipeline utilization factors between 20-60% on top of traditional stage level clock gating. This power benefit is achieved without negatively affecting pipeline latency or throughput. It is also shown that data glitch power introduced as a result of keep-

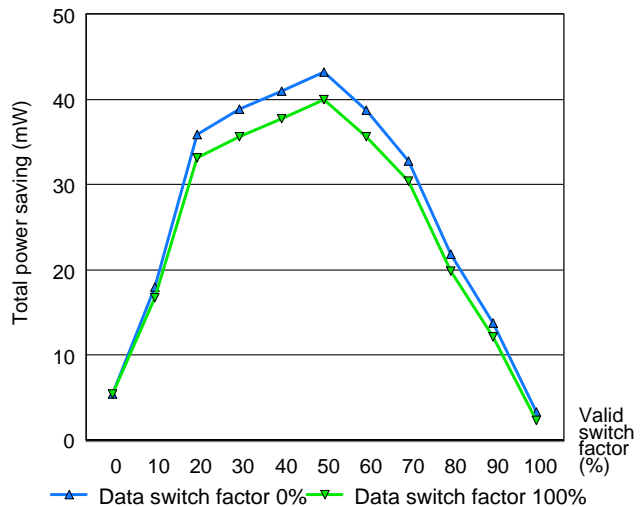


Figure 8: Absolute clock power savings for transparently clock gated MAAC unit, on top of traditional stage-level clock gating, over a pipeline utilization range of 0-100%. The two curves illustrate best/worst case bounds for introduced data glitch power.

ing latch stages transparent is limited to below 10% of the achieved clock power savings, making transparent pipelining techniques applicable even for logic with high glitch tendencies.

Acknowledgements. The authors wish to thank Peter Cook for providing the combinational dataflow for the MAAC unit.

7. REFERENCES

- [1] BROOKS, D. M., BOSE, P., SCHUSTER, S. E., JACOBSON, H., KUDVA, P. N., BUYUKTOSUNOGLU, A., WELLMAN, J., ZYUBAN, V., GUPTA, M., AND COOK, P. W. Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors. *IEEE Micro* 20, 6 (November/December 2000), 26–44.
- [2] CHANDRAKASAN, A., AND BRODERSEN, R. *Low Power CMOS Design*. IEEE Press., 1998.
- [3] CORREALE JR., A. Overview of the power minimization techniques employed in the IBM PowerPC 4xx embedded controllers. In *Proc. International Symposium on Low Power Electronics and Design (ISLPED)* (1995), pp. 75–80.
- [4] EFTHYMIU, A., AND GARSIDE, J. Adaptive Pipeline Depth Control for Processor Power-Management. In *ICCD* (2002).
- [5] GOWAN, M., BIRO, L., AND JACKSON, D. Power Considerations in the Design of the Alpha 21264 Microprocessor. In *DAC* (June 1998), pp. 726–731.
- [6] KOPPALIL, J., RAMRAKHYANI, P., DESAI, S., VAIDYANATHAN, A., AND ROTENBERG, E. A Case for Dynamic Pipeline Scaling. In *CASES* (October 2002).
- [7] SHIMADA, H., ANDO, H., AND SHIMADA, T. Pipeline Stage Unification: A Low-Energy Consumption Technique for Future Mobile Processors. In *ISLPED* (August 2003), pp. 326–329.
- [8] TIWARI, V., SINGH, D., RAJGOPAL, S., MEHTA, G., PATEL, R., AND BAEZ, F. Reducing Power in High-Performance Microprocessors. In *DAC* (June 1998), pp. 732–737.