Energy-Efficient Pipeline Templates for High-Performance Asynchronous Circuits

Basit Riaz Sheikh and Rajit Manohar, Cornell University

We present two novel energy-efficient pipeline templates for high throughput asynchronous circuits. The proposed templates, called N-P and N-Inverter pipelines, use single-track handshake protocol. There are multiple stages of logic within each pipeline. The proposed techniques minimize handshake overheads associated with input tokens and intermediate logic nodes within a pipeline template. Each template can pack significant amount of logic in a single stage, while still maintaining a fast cycle time of only 18 transitions. Noise and timing robustness constraints of our pipelined circuits are quantified across all process corners. A completion detection scheme based on wide NOR gates is presented, which results in significant latency and energy savings especially as the number of outputs increase. To fully quantify all design trade-offs, three separate pipeline implementations of an 8x8-bit Booth-encoded array multiplier are presented. Compared to a standard QDI pipeline implementation, the N-Inverter and N-P pipeline implementations reduced the energy-delay product by 38.5% and 44% respectively. The overall multiplier latency was reduced by 20.2% and 18.7%, while the total transistor width was reduced by 35.6% and 46% with N-Inverter and N-P pipeline templates respectively.

Categories and Subject Descriptors: B.2.1 [**Arithmetic and Logic Structures**]: Design Styles

General Terms: Design, Performance, Reliability

Additional Key Words and Phrases: asynchronous logic circuits; pipeline processing; very-large-scale integration.

ACM Reference Format:
Basit Riaz Sheikh and Rajit Manohar. 2011. Energy-Efficient Pipeline Templates for High-Performance Asynchronous Circuits. ACM J. Emerg. Technol. Comput. Syst. , , Article A (December 2011), 26 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

The scaling of CMOS technology into ultra-deep sub-micron range has posed some serious challenges for digital circuits designers. With the transistor threshold voltage fixed [Horowitz 2007], V_{DD} has been scaling very slowly if at all, which means all performance improvements come at an increased energy consumption. Hence, it is of no surprise that power has become a major design constraint in modern chip design. Furthermore, process variations in deep sub-micron range have made devices far less robust, which is increasingly making it difficult for synchronous designers to overcome the problems associated with clock skew rates and clock distribution [Dally and Poulton 1998].

The research was supported in part by NSF under grants CNS-0834582 and CCF–0428427. Equipment support was provided by NSF infrastructure grant CNS-0708788, and the processors were donated by Intel.
Author's addresses: Basit Riaz Sheikh and Rajit Manohar, Computer Systems Laboratory, School of Electrical and Computer Engineering, Cornell University. Email: basit@csl.cornell.edu, rajit@csl.cornell.edu

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Asynchronous quasi-delay-insensitive (QDI) circuits, with their robustness to process variations, no global clock dependence, and inherent perfect clock gating, represent a highly feasible design alternative for future chip design. The QDI circuits have been used in numerous high-performance, energy-efficient asynchronous designs [Sheikh and Manohar 2010] [D. Fang and Manohar 2005], including a fully-implemented and fabricated asynchronous microprocessor [Martin et al. 1997].

QDI circuits lose some of their energy efficiency gains in implementing handshakes between different parallel pipeline processes. To ensure QDI behavior for each handshake, every up and down transition within a pipeline is sensed, which leads to significant handshake circuitry and energy overhead. High throughput QDI pipelines only include a small amount of logic in each stage. The large number of pipeline stages required for high throughput make the handshake overhead a significant proportion of the total power consumption. In this work, we try to improve the energy efficiency of high performance asynchronous pipelines but without sacrificing robustness. To circumvent the problem of high handshake overhead, we present two novel pipeline templates, which greatly minimize the handshake circuitry by taking advantage of some easily satisfiable timing assumptions. Our proposed pipelines use single-track handshake protocols [van Berkel and Bink 1996]. Logic density is enhanced by packing multiple logic stages in a single pipeline, while still maintaining a very fast cycle time of 18 transitions.

To quantify actual performance and energy efficiency of our proposed pipeline templates, three separate pipeline implementations of an 8x8-bit booth-encoded array multiplier are presented. Compared to a standard QDI pipeline implementation, our proposed pipeline implementations reduced the energy-delay product by 38.5% and 44% respectively. The overall multiplier latency was reduced by 20.2% and 18.7%, while the total transistor width was reduced by 35.6% and 46% with the use of our newly proposed N-Inverter and N-P pipeline templates respectively.

The rest of the paper is organized as follows. Section 2 provides an overview of fine-grain asynchronous pipelines and their limitations. Section 3 discusses ways to improve the energy efficiency of fine-grain pipelines. In Section 4, we introduce two novel energy-efficient pipeline templates based on single-track handshake protocol and multi-stage logic. We also discuss an alternative energy-efficient completion detection scheme. Section 5 presents an in-depth analysis of various design trade-offs associated with our proposed templates. It also quantifies the noise and timing margin constraints of our circuits across all process corners. Section 6 gives a detailed description of an 8x8-bit booth-encoded array multiplier implementation and provides a comprehensive evaluation of our proposed pipeline templates using experimental results.

2. ASYNCHRONOUS PIPELINES

High performance asynchronous circuits are composed of many parallel processes. As opposed to synchronous circuits, which use a global clock to synchronize data tokens between different pipeline stages, these asynchronous parallel processes use handshake protocols to communicate with each other. These parallel processes are often referred to as fine-grain pipelined circuits. The fine-grain pipelined circuits use designated channels for communication between processes. A channel comprises a bundle of wires and a communication protocol to transmit data from a sender to a receiver. There are numerous asynchronous fine-grain pipeline implementations [Lines 1995] [Williams 1991] [Sutherland and Fairbanks 2001] [Ferretti and Beerel 2002]. A robust family of these circuit templates is referred to as quasi-delay-insensitive (QDI) circuits.

2.1. Quasi-Delay-Insensitive Circuits

QDI circuit templates use 1-of-N encoded channels to communicate between different parallel processes. In an 1-of-N channel, a total of N wires is used to encode data with only one wire asserted at a time. Most high throughput QDI circuits either use 1-of-2 (dual-rail) or 1-of-4 encodings. In an 1-of-4 encoded channel communication as shown in Figure 1, *validity* is signified by setting one of the four data rails and *neutrality* is indicated by resetting of all four data rails. In a four phase handshake process, which is commonly used in most high speed QDI circuits, the sender process initiates the communication by sending data over the rails i.e. by asserting one of the data rails. The receiver process detects the presence of data and sends an acknowledge once it no longer needs the data. At this point, the sender process resets all its data rails. The receiver process detects the neutrality of input tokens. It de-asserts the acknowledge signal once it is ready to receive a new data token. The cycle repeats.

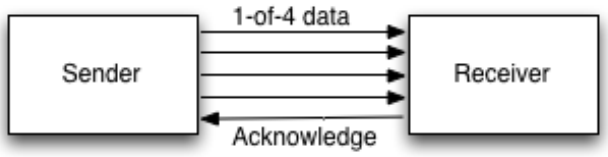

Fig. 1. Asynchronous pipelines: sender-receiver handshake protocol.

2.1.1. Pre-Charge enable Half-Buffer. The pre-charge enable half-buffer (PCeHB) [Fang and Manohar 2004] template, which is a slightly modified version of pre-charge half-buffer (PCHB) template proposed in [Lines 1995] [Williams 1991], is a workhorse for most high throughput QDI circuits. It is both small and fast with a cycle time of 18 transitions. In a PCeHB pipeline, the logic function being computed is implemented by a pull-down NMOS stack. The input and output validity and neutrality are checked using separate logic gates. The actual computation is combined with data latching, which removes the overhead of explicit registers.

A PCeHB template can take multiple inputs and produce multiple outputs. Figure 2 shows a simple two input and one output PCeHB template. *L0* and *L1* are dual-rail inputs to the template and *R* is a dual-rail output. A PCeHB template has a forward latency of two transitions. Each pipeline stage computes logic by using a NMOS pull-down stack followed by an inverter to drive the output.

To understand the cycle time of 18 transitions in a PCeHB template, let us assume two PCeHB pipelines in series with time (t) increments taken in terms of logic transitions.

— At $t = 0$, input tokens arrive at the first PCeHB pipeline block.
— At $t = 2$, first pipeline block produces its output.
— At $t = 4$, second pipeline block produces its output.
— At $t = 5$, $L.e$ in the first block goes low.
— At $t = 7$, $L.e$ in the following block, which is the $R.e$ of the first block, goes low. This indicates that the output from the first pipeline block is no longer needed and can be reset.
— At $t = 9$, *en* signal in the first block is de-asserted.
— At $t = 10$, $\_R$ rails in the first block are pre-charged.
— At $t = 11$, output, R, rails of the first block are reset.

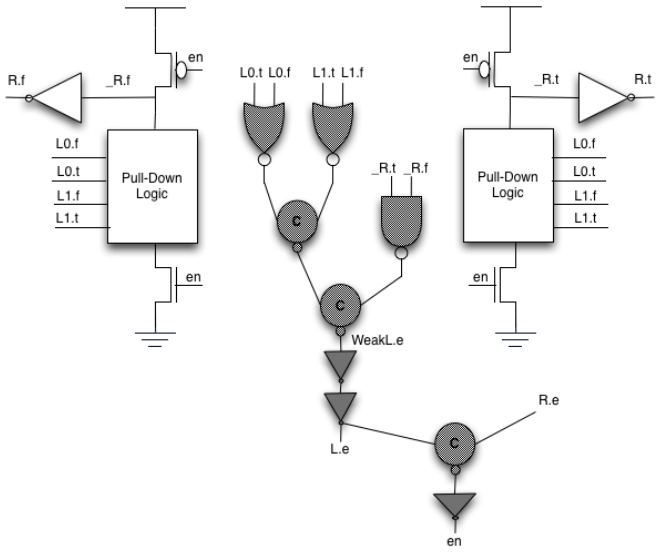

Fig. 2. A two input and one output PCeHB template.

— At $t = 12$, $\_R$ rails in the second block are pre-charged.
— At $t = 14$, $L.e$ in the first block goes high.
— At $t = 16$, $L.e$ in the second pipeline stage goes high. This indicates the neutrality of the inputs in the second pipeline stage.
— At $t = 18$, en is set in the first pipeline block, which indicates that the pipeline is ready to accept new input tokens and compute a new output.

The highlighted logic gates in Figure 2 are not used for the actual computation but are only required for the handshake protocol. This includes the generation of completion detection signal ($L.e$) as well as the en signal that is used to enable computation or latching in the pipeline stage. As the number of inputs into a PCeHB pipeline stage increases, the input validity tree becomes more complex and may require extra stages to compute, which leads to an increase in the cycle time. The same holds true as the number of outputs increase. Hence, for high-througput circuits each PCeHB stage contains only a small amount of logic with only a few inputs and outputs. This leads to significant handshake overhead, in terms of power consumption and transistor count, as tokens may have to be copied for use in separate processes with each process doing its own validity and neutrality checks.

Table I shows the power consumption breakdown of a simple full-adder circuit implemented using a PCeHB template. Only 31% of the total power is consumed in the actual logic, while the rest is spent in implementing the handshake protocol. This is a significant power overhead, which gets worse as the complexity of PCeHB templates increases with more inputs and outputs. The result in Table I was one of the main motivating factors that prompted us to consider alternative pipeline solutions with less handshake circuitry.

2.2. Fine-grain bundled-data pipelines

The fine-grain bundled-data pipelines have an instant area advantage over the QDI pipelines because of their use of single-rail encoded data channels [Sutherland and Fairbanks 2001]. However, the bundled-data pipelines include far more timing

Table I. PCeHB full-adder pipeline: power breakdown.

Circuit	Power
Logic	31%
Handshake	69%

assumptions than QDI circuits which makes them less robust. The bundled-data pipelines contain a separate control circuitry to synchronize data tokens between different pipeline stages. The control circuitry includes a matched delay line, the delay of which is set to be larger than that of the pipeline's logic delay plus some margin. In [Sutherland and Fairbanks 2001], for correct operation, the designer has to ensure that the control circuit delay satisfies all set-up and hold time requirements just like in synchronous design. Since our goal was to design pipeline templates with robust timing and with forward latency similar to that of precharged logic, we did not consider any bundled-data pipeline implementations in our work.

3. IMPROVING ENERGY EFFICIENCY OF FINE-GRAIN PIPELINES

QDI circuits are robust since each up and down transition within a QDI pipeline template is sensed. But this robustness comes at the cost of significant power consumption in pipeline handshake circuitry as shown in Table I. The high handshake overhead is one of the serious constraints hampering the wide-range adoption of QDI circuits especially for logic operations with a large number of input and output signals, such as a 32-bit multiplier.

In this work, we try to improve the energy efficiency of high performance asynchronous pipelines but without sacrificing robustness. To this end, we kept the following objectives for our resulting pipeline templates:

— Keep the cycle time of each stage within 18 transitions.
— Increase the ratio of logic to handshake. The handshake power overhead must account for less than 50% of total pipeline power.
— No increase in the total transistor count is allowed.
— All timing assumptions are either isochronic fork assumption [Martin 1990] or have at least the same timing margin as the half-cycle timing assumption [LaFrieda and Manohar 2009] according to which the difference in number of transitions between any two delay races must be at least 4.5 transitions.
— Stalls on input and output should not impact correct operation.

We envision these circuits being used for large chunks of local logic (e.g. a multiplier) wrapped with QDI interfaces, rather than globally.

In the past, researchers have tried to increase the logic density of QDI pipelines by adding extra logic stages [Beerel et al. 2009], but this still does not yield the desired reduction in the handshake overhead and leads to an increase in cycle time. To analyze this effect, let us suppose we increase the logic depth of a pipeline by adding extra logic stages. To conform to QDI behavior, the up and down transitions of all newly-created internal signals must be acknowledged. This can be done either by explicitly checking for each transition using completion detection logic as is done in the PCeHB template or using *weak conditions* [Seitz 1980] i.e. the output being valid implies that the input is valid (checked by additional n-fets in the logic stack), and the output being neutral implies that the input is neutral (checked by additional p-fets in the logic stack). The limitations of *weak conditions* for performance are elaborated in [Seitz 1980] [Lines 1995]. In the case of explicit checking, there is the associated high handshake overhead because of all the extra validity and neutrality detection logic gates. All these extra

transitions associated with the newly added logic stages and completion detection logic gates limit energy efficiency gains.

There is clearly a need to look beyond just adding extra logic stages to each pipeline stage. To improve the energy efficiency of high throughput asynchronous pipelines, we look at alternative handshake protocols as well as some timing assumptions in QDI circuits.

3.1. Four phase handshake vs. Single-track handshake

In a four phase handshake protocol, the pipeline stage needs to detect the validity and the neutrality of both inputs and outputs. During the second half of the four-phase protocol when the pipeline is waiting for inputs and outputs to be reset, no actual logic is being computed but it still consumes roughly half of the cycle time. Furthermore, the power consumed in detecting the neutrality of inputs and outputs rivals that consumed during their validity detection. Due to these characteristics, the four phase handshake protocol is clearly not an ideal choice for energy efficiency.

Single-track handshake [van Berkel and Bink 1996] protocol tries to overcome this weakness of four phase protocol by practically eliminating the neutrality phase. Figure 3 shows an overview of a single-track handshake protocol. The sender process initiates the communication by sending the data token. The receiver uses the data for computing its logic. Once the data is no longer needed, instead of sending an acknowledge signal back to the sender process, the receiver process resets the input tokens itself by pulling the data wires low through NMOS transistors as illustrated in Figure 3. There are as many NMOS discharge transistors as there are data wires, but for simplicity we show only one discharge transistor in Figure 3. As the data wires pulled low, the sender detects the token consumption and gets ready to send the next token. Hence, eliminating the transitions associated with second part of the four phase protocol.

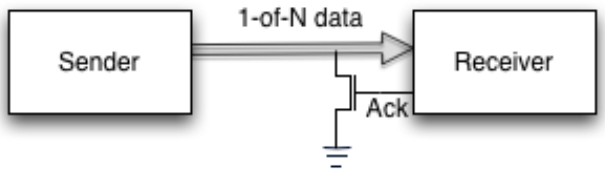

Fig. 3. Single-track handshake protocol.

There has been very limited work on single-track handshake templates. Most of the prior work has focused on using single-track handshake protocol to reduce the cycle time of asynchronous pipelines to less than 10 transitions and not on how to use these extra transitions to improve logic density and energy efficiency. Ferretti et al[2002] provide a family of asynchronous pipeline templates based on single-track handshake protocol. Just like high throughput QDI circuits, each of their pipeline templates contains only a small amount of logic. Furthermore, their 6-transition cycle time pipelines use some very tight timing margins that may require significant post-layout analog verification. Single-track circuits have been used in the control path of GasP [Sutherland and Fairbanks 2001] bundled-data pipelines. However, the actual data path of the pipeline does not use a single-track handshake protocol.

We employ single-track handshake protocol for our proposed pipeline templates. However, our design effort focuses on increasing the logic density and energy efficiency of each pipeline stage and not on reducing cycle time.

3.2. Relative Path Timing Assumption

QDI circuits are highly tolerant of process variations as each transition within a QDI pipeline is sensed. The isochronic fork assumption [Martin 1990], which states that the difference in delay between branches of a wire is insignificant compared to the gate delays of the logic reading their values, is the only timing assumption allowed in QDI design. Recently, LaFrieda et al [2009] exposed another timing assumption that is quite commonly used in QDI implementations, which they named as the half cycle timing assumption (HCTA). According to HCTA, the difference in number of transitions between any two delay races must be at least 4.5 transitions for PCeHB-style templates. The resulting templates are referred to as Relaxed QDI templates and are shown to be quite robust.

LaFrieda et al [2009] exploited HCTA to improve energy efficiency of their four phase handshake protocol pipelines. In this work, we look to improve energy efficiency of single track handshake protocol pipelines by introducing timing assumptions with a margin of at least 5 gate transitions between any two relative delay races.

4. HIGH THROUGHPUT ENERGY-EFFICIENT PIPELINE TEMPLATES

4.1. N-P and N-Inverter Pipeline Templates

We use single-track handshake protocol for our proposed pipeline templates. Figure 4 shows a semi detailed depiction of our first proposed template with 5 arbitrary dual-rail outputs indicated by signals *R0* to *R4*. We have named the template N-P pipeline since it computes logic using NMOS pull-down and PMOS pull-up stacks. Each NMOS and PMOS stage can comprise multiple logic stacks. However, for simplicity, we do not show multiple logic stacks and global reset signals.

A PCeHB template has two logic stages per each pipeline, with the second logic stage comprising an inverter to drive the output rails. Hence, there is only one effective logic computation per pipeline block. In contrast, the N-P template has N arbitrary stages of actual logic computations. However, for ease of explanation and to keep cycle time within 18 transitions, we use N-P pipelines with four stages of logic. In the reset state, the NMOS logic nodes in the pipeline are precharged, whereas the PMOS logic nodes are pre-discharged. Each state-holding gate includes a staticizer, which comprises a keeper and a weak feedback inverter, to ensure that charge would not drift even if the pipeline were stalled in an arbitrary state. The staticizers, drawn as two cross-coupled inverters, for the intermediate as well as the final output nodes are shown in Figure 4.

When 1-of-N encoded input tokens arrive, logic is computed in the first stage by pulling down the precharged nodes. This is similar to how logic is computed in QDI templates. We limit the number of series transistors in an NMOS stack to a total of four. The second logic stage uses a stack of PMOS transistors to compute logic by pulling up the pre-discharged nodes. As the PMOS transistors have slower rise times, for throughput purposes we limit the number of series transistors in a PMOS stack to a total of three (including the enable). As the output nodes from the second stage pull up, the pull-down stacks in the third stage get activated and compute logic by pulling down their output nodes. Finally, the fourth stage computes logic by using its pull-up stack of PMOS transistors. The four cascaded stages of logic in our pipeline are similar to cascaded domino logic but without any static inverters in between dynamic logic stages.

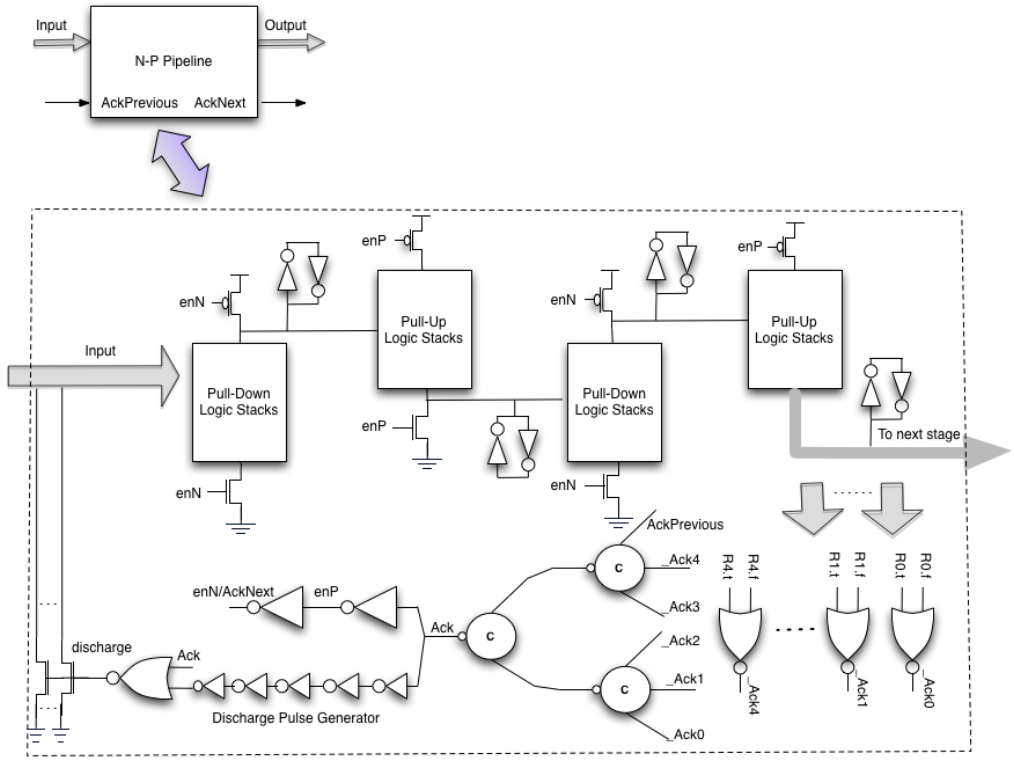

Fig. 4. N-P pipeline template

There are no explicit validity detection gates for the arriving input tokens nor for any intermediate outputs that are being produced. *AckPrevious* (explained later in this section) signifies the validity of input tokens into the pipeline and alleviates the need to explicitly check for validity. For intermediate outputs produced and consumed within the template, validity must be embedded in a pull-up or pull-down logic stack that uses the intermediate output to compute the following stage logic output. This could incur additional cost, depending on the function being implemented. However, for a logic stack inherently embedded with input validity, for example a stack that computes the sum of two inputs, there is zero validity detection overhead. The elimination of explicit validity detection gates for input tokens and intermediate output nodes leads to considerable power savings and minimization of handshake overhead.

There is an explicit completion detection logic for all the outputs that eventually leave the pipeline, either at the end of the second stage or the fourth stage. The completion detection of the final outputs automatically signifies the validity of all intermediate outputs as well as that of all the initial input tokens into the N-P pipeline. The completion detection logic comprises a set of NOR gates and a c-element tree as shown in Figure 4. Each of the c-element gates includes a staticizer in parallel. These staticizers are not shown for simplicity. The outputs from the NOR gates are combined using a c-element tree which de-asserts the *Ack* signal once all outputs are valid. This leads the discharge signal to go high, which initiates the reset of all input tokens. The discharge signal is only set for a short pulse duration. The de-asserted *Ack* signal also sets the *enP* signal to high which discharges all pull-up nodes in logic stage two. The

enN signal is set low, which precharges all pull-down nodes in logic stages one and three. Since the neutrality of the internal nodes is not sensed, we introduce a timing assumption on their transition. The discharge of input tokens with a short pulse signal introduces another timing assumption. These two timing assumptions entail the following constraints:

— The pull-down nodes must be fully precharged before *enN* goes high and pull-up nodes must be fully discharged before *enP* transitions low. This translates into a race condition of 1 pull-up/pull-down transition versus 9 gate transitions, the minimum transition count before both *enN* and *enP* flip when two N-P pipelines are in series.
— All input tokens must be fully discharged within the short pulse discharge period. The pulse has a minimum period of 5 gate transitions. There are as many NMOS discharge transistors as there are input data rails.

The robustness of our pipeline template is not compromised as these timing assumptions satisfy the minimum timing constraint of at least 5 gate transitions between any two relative path delay races.

The discharge of any of the outputs before the validity of all other outputs has been acknowledged can permanently stall the pipeline. To analyze this effect, let us suppose we have three N-P pipelines *A*, *B*, and *C* as shown in Figure 5. *A* produces two outputs, one of which goes to *B* and the other one to *C*. *B* uses the output from *A* to compute its output. Since *B* has computed its output, it can now discharge the input it received from *A*. If *A*'s other output, which is headed for *C*, is not yet produced or acknowledged by the completion detection logic of *A*, then *B*'s discharge of its input will make the completion detection logic of *A* unstable. To prevent this, we add the *AckNext* signal to our pipeline template. It is sent to all following pipeline stages that consume the outputs from the current N-P pipeline. This signal is referred to as *AckPrevious* in the destination pipeline as shown in Figure 4. It prevents the discharge of the tokens coming from the sender stage before the validity of all outputs in the sender has been acknowledged. As mentioned earlier, *AckPrevious* also signifies the validity of input tokens into the pipeline, hence alleviating the need to check for input token validity in NMOS pull-down stacks. In the case where inputs come from more than one pipeline block, the *AckPrevious* signals from all corresponding pipeline blocks need to be added to the completion detection logic to ensure against any premature discharge of input data rails.

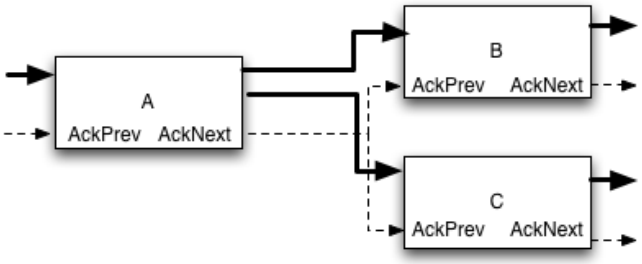

Fig. 5. Ack signals to ensure correctness

Forking of an output to two successors is also not allowed because then the two successors can reset (discharge) the connection at different times, which could lead to

potential conflicts. Hence, we need to create explicit duplicate outputs in the last logic stack for each output that goes to multiple destinations.

To determine the cycle time of the proposed N-P pipeline, let us assume two N-P pipelines in series with time (t) increments taken in terms of logic transitions.

— At $t = 0$, input tokens arrive at the first pipeline block.
— At $t = 4$, first pipeline block produces its output.
— At $t = 7$, *Ack* signal in the first block is de-asserted which signifies the validity of all output signals
— At $t = 8$, second pipeline block produces its output.
— At $t = 9$, input tokens in the first pipeline block are discharged. Internal PMOS logic nodes are discharged.
— At $t = 10$, NMOS logic nodes in the first pipeline are precharged.
— At $t = 13$, output tokens from the first pipeline block are discharged by the second pipeline.
— At $t = 16$, *Ack* signal in the first block is asserted which signifies the reset of all output signals.
— At $t = 18$, *enN* is set and the pipeline is ready to accept new input tokens.

Hence, our proposed N-P pipeline has a cycle time of 18 transitions. Stalls on inputs and outputs do not impact correct operation. The template waits in its present state if inputs arrive at different times. This holds true for outputs being computed at different times as well. The relative path delay assumption has a root, *Ack*, which only changes after all inputs have arrived and all outputs have been computed. As a result, correct operation is not a function of arrival time of signals, which makes the N-P template quite robust.

We could invert the senses of the inputs and outputs by changing the order of the logic stacks within N-P pipeline. With inverted inputs, the first stage comprises PMOS logic stacks and the final logic stage comprises NMOS logic stacks with the outputs produced in inverted sense. This could improve the drive strength of the output signals especially in the case of high fan-out.

Our second proposed pipeline template replaces the PMOS pull-up logic stacks in stage 2 with an inverter, hence the name N-Inverter template, and includes only a single pull-up PMOS transistor in stage 4 as shown in Figure 6. As PMOS logic stacks have slower rise times and relatively weak drive strength, the N-P template cycle time may incur a performance hit. The N-Inverter template addresses this by using inverters with faster switching time and strong drive strength . It also results in better noise margins as discussed in detail in Section 5. However, these improvements come at the cost of reduced logic density as stage 2 and 4 no longer perform any effective logic computation. Despite these alterations, the N-Inverter and N-P templates use exactly the same timing assumptions. The completion detection and handshake circuitry is also identical.

4.2. Completion detection logic for large number of outputs

Since N-P and N-Inverter pipeline templates can pack significant logic in a single pipeline block, there may be cases where a pipeline block has quite a large number of outputs. To detect the validity of these large number of outputs, we may have to expand the c-element validity tree by a couple of extra stages as shown in Figure 7.

As a result of these two extra stages in the completion detection validity tree, the cycle time of N-P and N-Inverter templates is no longer 18 transitions. There are four extra transitions, two each for the validity and neutrality detection of the output signals, which increases the cycle time to 22 transitions. Since our goal was to keep the cycle time within 18 transitions, we explored a number of other completion detection

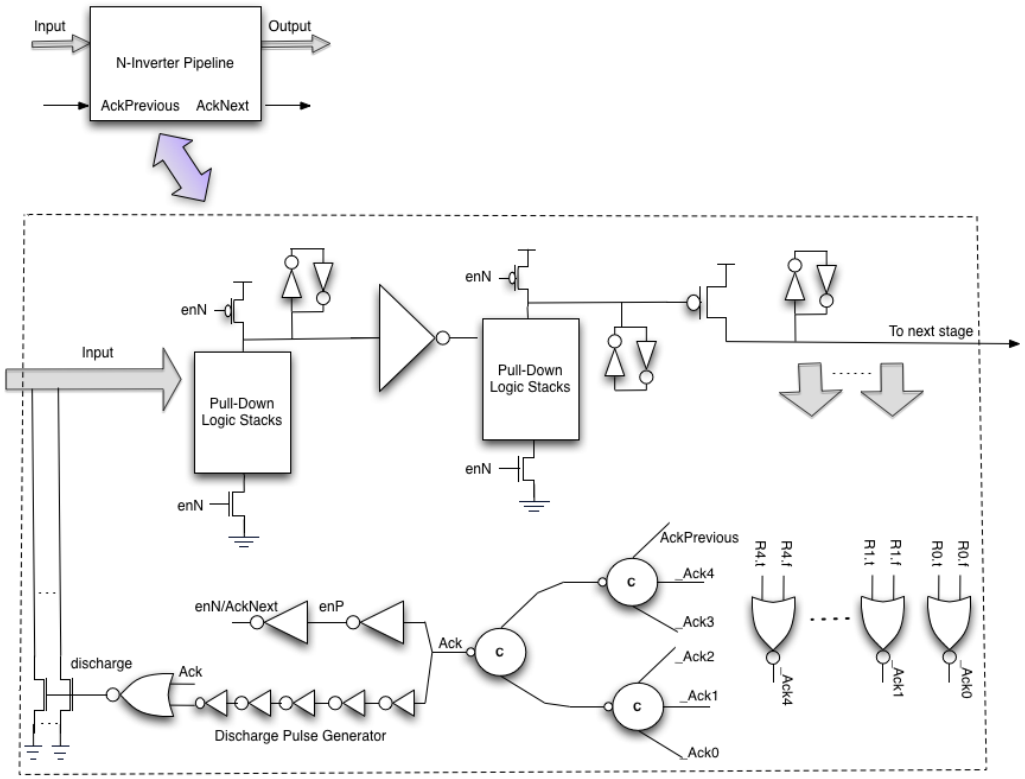

Fig. 6. N-Inverter pipeline template

circuits [Schuster and Cook 2003] [Cheng 1998]. To reduce the cycle time back to 18 transitions, we use wide NOR gates based completion detection circuitry as proposed in [Cheng 1998], but with a couple of optimizations to make the circuitry feasible for our proposed pipeline templates. These optimizations include the use of only one output from the set of outputs destined for the same next pipeline block for neutrality detection and the addition of *enP* and *enN* transistors in the pull-up stacks of *DONE* and *RST* circuits as seen in Figure 8. These optimizations and their benefits are outlined in detail towards the end of this section.

The *Ack* signals are generated using static NOR gates as previously. The validity of the outputs is signaled by the setting of *Done*. To ensure that the *Done* signal is only set once all *Acks* have gone low, the pull-up path resistance of the *Done* circuit is set to be at least 4 times as big the pull-down path resistance when only one pull-down transistor is conducting. To prevent a direct path between V_{DD} and *GND*, the *Ack* from one of the latest (slowest) outputs is used in the pull-up stack.

The *RST* signal is used to sense the reset of all outputs. The various *R.t* and *R.f* signals correspond to the actual dual-rail outputs being produced. The latest (slowest) signal to reset is put in the pull-up stack. The pull-up path resistance of the *RST* circuit is set to ensure that it only goes high once all pull-down transistors in the *RST* circuit have turned off i.e. all output signals have reset. The *RST* circuit has two pull-down transistors for each dual-rail output and four pull-down transistors for each 1-of-4 output. As the number of outputs increase, the *RST* rise time suffers significantly.

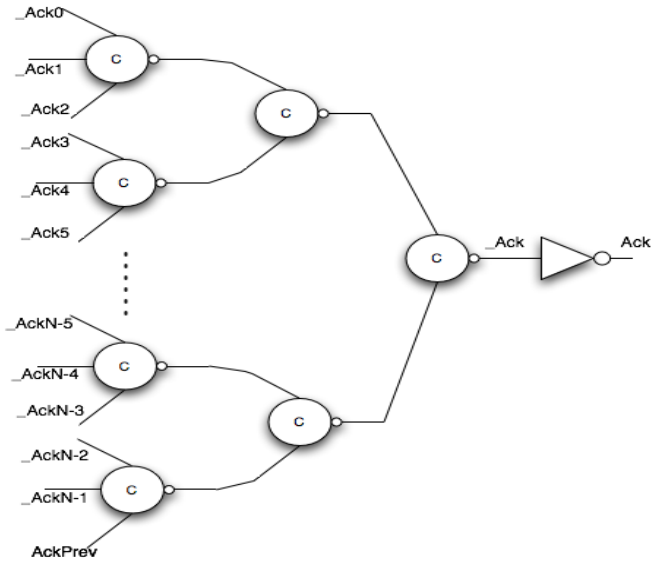

Fig. 7. Multi-stage c-element tree completion detection logic for large number of outputs

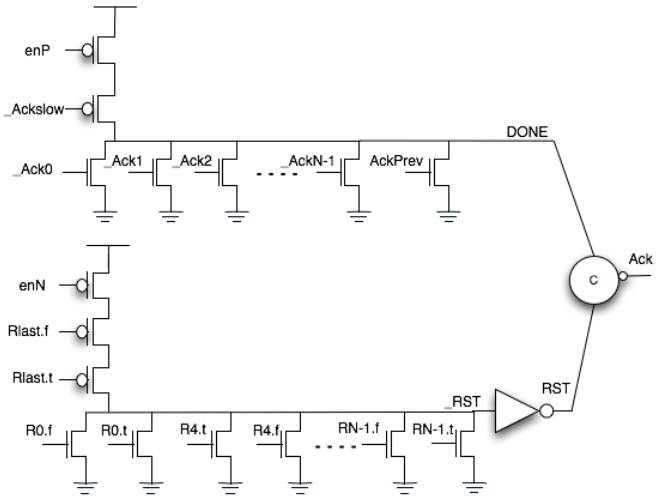

Fig. 8. Completion detection logic for large number of outputs

A close inspection of our proposed pipeline templates made us realize that for outputs destined for the same pipeline block, we only need to check for the reset of one of the outputs and not all because they use the same discharge pulse. Let us assume the dual-rail outputs *R0* to *R3* are all headed for the same pipeline block. We minimize the *_RST* circuit by only using pull-down transistors corresponding to *R0* output. The transistors corresponding to *R1*, *R2*, and *R3* dual-rail outputs are eliminated as shown in Figure 8.

The addition of *enP* and *enN* transistors in the pull-up stacks of *DONE* and $\_RST$ circuits was another optimization we introduced. The *enP* signal cuts off the pull-up path in the *DONE* circuit while the pipeline is waiting for the outputs to be reset. This prevents the occurrence of a direct path between V_{DD} and *GND* if any of the $\_Acks$ other than $\_Ackslow$ goes high first. Similarly, the introduction of *enN* in the pull-up stack of $\_RST$ cuts off the direct path between V_{DD} and *GND* during the evaluation phase.

5. DESIGN CONSIDERATIONS AND ROBUSTNESS TRADE-OFFS

5.1. Completion Detection Circuits

To quantify the trade-offs between the two completion detection schemes, we carried out detailed SPICE level simulations with estimated wire loads for each node. It is assumed that each output goes to a separate pipeline block, hence the discharge of each signal is checked. The wide NOR completion detection circuitry results in lower latency relative to multi-stage c-element tree detection completion scheme across a wide range of outputs as shown in Figure 9. The latency difference increases as the number of outputs increases since c-element completion may require multiple extra stages. For 15 output signals, the wide NOR completion results in 30% less latency.

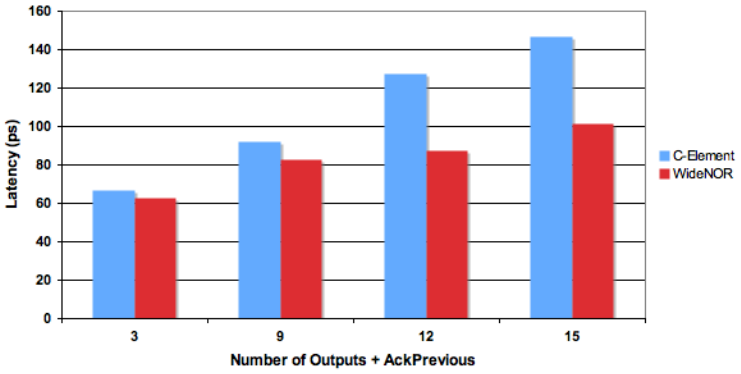

Fig. 9. Latency comparison of completion detection schemes

In terms of energy consumption, the choice of a completion detection scheme depends not only on the number of outputs but also on the arrival order and the delay of the chosen *latest* signal as shown in Figure 10. The x-axis corresponds to the arrival order of the chosen *latest* signal. For example, the data point corresponding to the arrival order of 9 means that our chosen *latest* output was the ninth output to be set or reset. All of the remaining signals arrived after an arbitrary 2-FO4 delay. This corresponds to a period of direct path between V_{DD} and *GND* for wide NOR based completion detection scheme. The c-element based completion scheme consumes the same energy irrespective of the arrival order. It is also more energy-efficient when the number of outputs is 9 or less. However, with a greater number of outputs as may be required for some N-P and N-Inverter pipeline templates, the wide NOR based completion detection scheme consumes significantly less energy. Another noteworthy observation from Figure 10 is that the effect of arrival order on energy consumption is only profound when the *latest* signal is one of the last few signals.

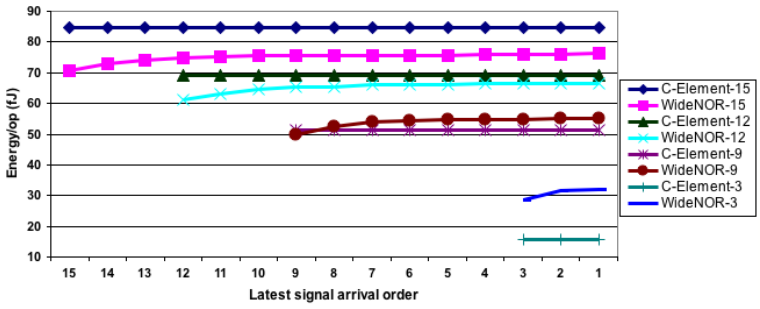

Fig. 10. Completion detection energy consumption for different arrival order of chosen signal

The longevity of the period of direct path between V_{DD} and *GND*, when the chosen *latest* signal is the not the last one, may lead to significant energy consumption for wide NOR based completion detection scheme. To explore this effect, we simulated wide NOR completion circuits for 12 and 15 outputs by varying the delay of late arriving signals as seen in Figure 11 and Figure 12. For 12 outputs, unless any output arrives 3 or more FO4 delays after the chosen *latest* signal, the wide NOR completion consumes less energy compared to the c-element based completion scheme, irrespective of the arrival order of the *latest* signal. For 15 outputs, the margin increases to 5 or more FO4 gate delays for the wide NOR completion to consume more energy than the corresponding c-element based completion detection scheme. These results indicate that the energy consumption of the wide NOR completion scheme is largely a function of delay between the chosen *latest* signal and the actual last signal. In the case of small delay variability between outputs produced within the same pipeline block, for example most arithmetic operations, the wide NOR scheme consumes less energy per operation than its c-element tree counterpart. In the unusual scenario of large delay difference between outputs within the same pipeline, the wide NOR scheme still functions correctly, albeit at higher energy consumption.

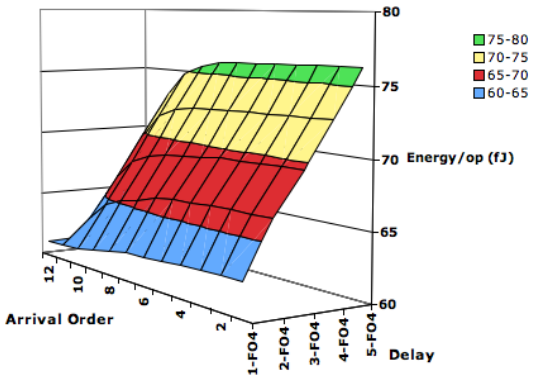

Fig. 11. C-Element vs WideNOR for 12-outputs with varying delay of latest signal

In terms of transistor area, the wide NOR completion detection circuit becomes more efficient as the number of outputs increase as seen in Figure 13. The choice of a par-

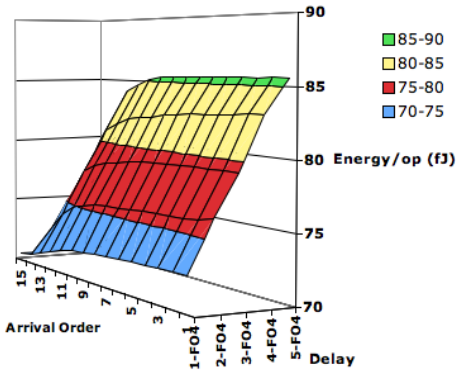

Fig. 12. C-Element vs WideNOR for 15-outputs with varying delay of latest signal

ticular completion detection circuit is therefore a design choice, which may depend on a number of factors: the number of outputs for a pipeline stage, latency and throughput targets, power budget, area constraints, and the delay variability of chosen *latest* output.

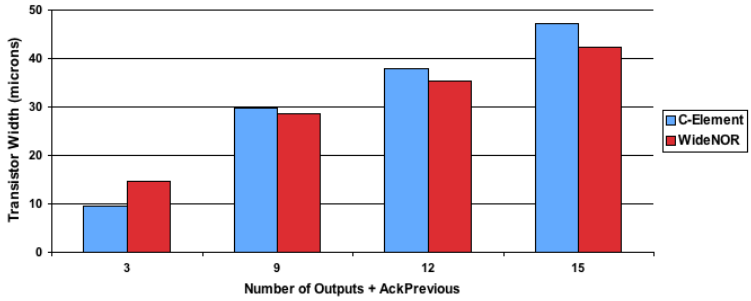

Fig. 13. C-Element vs WideNOR: total transistor width comparison

5.2. Throughput, Energy, and Area Trade-offs

Throughput, energy, and area are critical design considerations for a circuit designer. We choose an 8-to-1 multiplexor design, which produces multiple copies of the output as shown in Figure 14, to highlight some of these trade-offs in our proposed templates. PCeHB, N-P, and N-Inverter pipelined versions of the chosen circuit were implemented. Highest precision SPICE simulations were conducted in 65nm bulk CMOS process with estimated wire loads for each node.

Although, all three templates have a cycle time of 18 transitions, the N-P implementation results in an 8.5% lower throughput. The N-P implementation is slower because it employs some logic computations in PMOS stacks, which have slower slew rates and weaker drive strength than NMOS stacks. In a PCeHB implementation, each 2-to-1 multiplexor represents a separate pipeline stage with each stage incurring a significant handshake overhead as seen earlier in Table I. There is a separate pipeline block

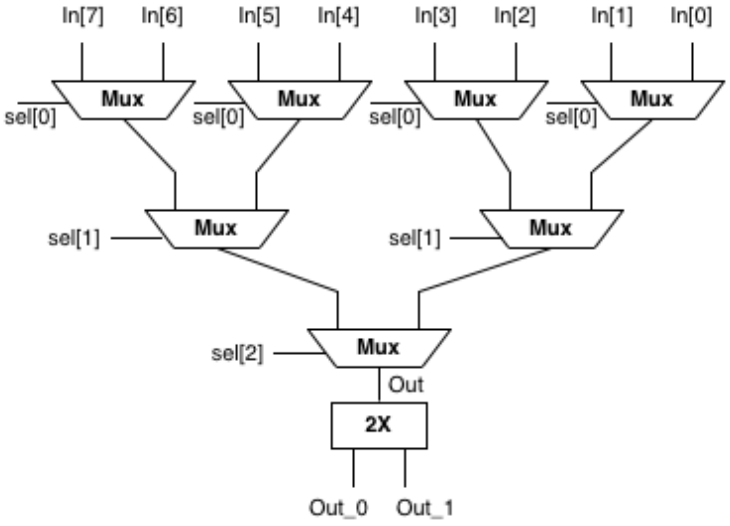

Fig. 14. 8-to-1 multiplexor with 2 copies of Output

for copy logic as well. Whereas, in N-P and N-Inverter implementations, the full 8-to-1 multiplexor circuit including copy logic can be packed completely in one and two pipeline blocks respectively. The effect of this logic compaction on energy efficiency and total transistor width is quite profound. Our N-Inverter implementation, operating at the same throughput as a PCeHB design, consumed 52.6% less energy per operation while using 48% less transistor width. With N-P pipeline, the energy and transistor width savings shoot up to 71.2% and 65% respectively, albeit at an 8.5% throughput penalty.

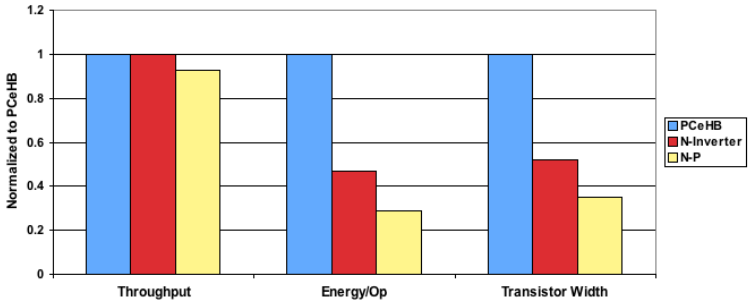

Fig. 15. 8-to-1 multiplexor design trade-offs for different pipeline styles

The proposed N-P and N-Inverter templates enable us to pack more logic computations within a single pipeline stage while maintaining a very high throughput. This flexibility is not available in standard PCeHB designs, which are composed of pipeline stages with only one effective logic computation in a single stage. More logic per a single stage in our proposed templates creates a likelihood of a large number of outputs per pipeline, which may adversely affect overall throughput as shown in Figure 16. The

dependency of absolute throughput on the number of outputs highlights an important design trade-off. With more outputs, although the number of transitions remain the same with the use of a wide NOR completion detection logic, each of these transitions incur a higher latency as shown earlier in Figure 9. The results would be even worse if a c-element based completion logic was used as it would incur 4 extra transitions per each cycle.

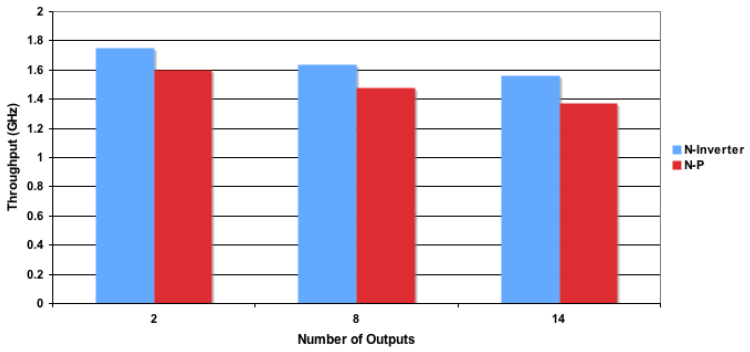

Fig. 16. Throughput dependency on the number of outputs

5.3. Noise Analysis

Noise feedthrough is one of the major concerns when it comes to the use of dynamic gates. Since our proposed pipeline templates use cascaded dynamic gates for logic computations, we carried out comprehensive noise margin analysis of our circuits. Dynamic gates from each pipeline template were simulated across all process corners, typical-typical (TT), slow-fast (SF), fast-slow (FS), slow-slow (SS), and fast-fast (FF), in a 65nm bulk CMOS technology with highest-precision SPICE configuration at 1V nominal V_{DD} and 85°C operating temperature. Since SPICE simulations do not account for wire capacitances, we included additional wire load in the SPICE file for every gate in the circuit. For each pipeline template, the lowest value of noise margin amongst all process corners was chosen.

For noise feedthrough analysis of N-P template, we analyzed a full-adder NMOS logic stack followed by a two-input AND gate in a PMOS pull-up stack. The noise margin, as defined in [Weste and Harris 2004], of this cascaded N-P configuration is the difference in magnitude between the minimum low input voltage recognized by NMOS logic stack on one of the inputs at unity gain point and maximum low output voltage of the driving PMOS pull-up stack. For N-Inverter and PCeHB templates, we analyzed a full-adder NMOS logic stack followed by a static CMOS inverter, with noise margin defined as the difference in magnitude between the minimum low input voltage recognized by NMOS logic stack on one of the inputs and maximum low output voltage of the driving output inverter. The results are shown in Figure 17 which also shows the noise margin of a two-input static CMOS NOR gate for comparison.

The N-P template has the lowest noise immunity. However, the noise margin can be significantly improved by increasing the relative drive strength of the staticizers to dynamic logic stacks. But this improvement comes at the cost of throughput degradation and a slight increase in energy per operation as shown in Figure 18 and Figure 19

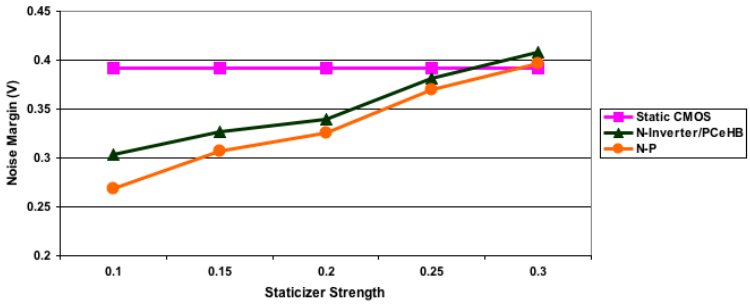

Fig. 17. Noise margin analysis

respectively. The energy per operation results are normalized to a PCeHB implementation energy per operation at a staticizer strength of 0.1. As seen from these results, the choice of an exact strength value for a staticizer represents a design trade-off, which should be made on the basis of final throughput target, desired robustness, as well as circuit application.

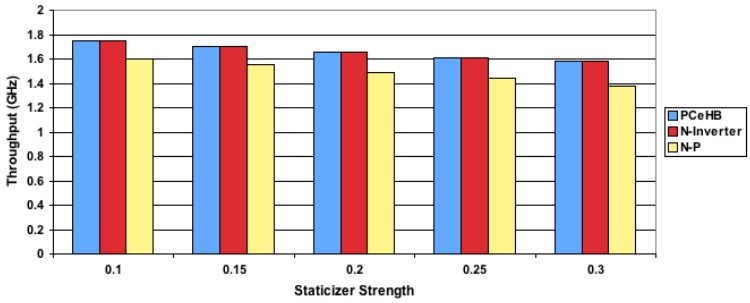

Fig. 18. Effect of staticizer strength on pipeline throughput

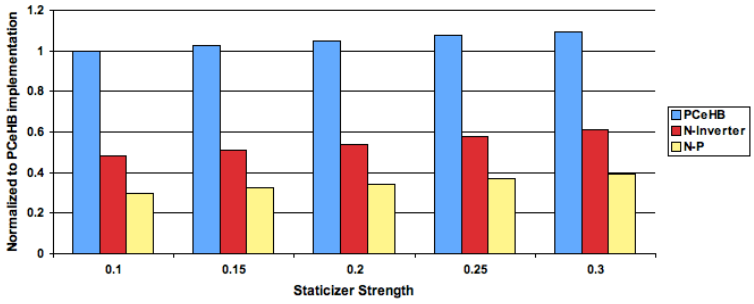

Fig. 19. Effect of staticizer strength on energy/op

5.4. Timing Margin

The N-P and N-Inverter pipeline templates include multiple timing assumptions, the breach of which could impact correct operation or stall the pipeline. In Section 4, we discussed the timing margins necessary to ensure correctness, but these timings margins were given in terms of gate transitions. To ensure sufficient robustness of our templates, we analyzed the exact timing constraints of full transistor-level implementations of our proposed pipelines in a 65nm bulk CMOS technology with highest-precision SPICE configuration at 1V nominal V_{DD}, 85°C operating temperature, and estimated wire loads for each gate. The timing constraint of 9 gate transitions for precharge and discharge of internal nodes translated into 14.8 FO4 and 12.2 FO4 delays for N-P and N-Inverter pipelines respectively, whereas the worst case transition corresponding to precharge or discharge of an internal node took no longer than 2.67 FO4 delays. This yields a very safe timing margin of over 12 FO4 and 9.5 FO4 delays for N-P and N-Inverter pipelines respectively.

The second timing assumption in the N-P and N-Inverter pipelines pertains to the full discharge of all input tokens within the short pulse discharge period. The 5 transition discharge pulse translates into 5 FO4 delays for both N-P and N-Inverter templates. The discharge pulse timing margin is a function of input load, which in turn is a function of input gate and wire capacitances. Since we envision our proposed templates to be used for large chunks of local computation and not for global communication, we found the short pulse period sufficient for full input token discharge including the added wire capacitance for each node, which corresponds to 12.5 μm wire length. In the worst case, an input token took no longer than 2.5 FO4 delays to fully discharge, which yields a timing margin of 2.5 FO4s. Since the discharge pulse period is not on pipeline critical path for both forward latency and throughput, the timing margin could be improved by adding two extra inverters to the pulse generator inverter chain without affecting performance. With these two extra inverters, the timing safety cushion increases from 2.5 FO4 to 4.5 FO4 delay, which makes the templates significantly more robust.

6. 8X8-BIT BOOTH-ENCODED ARRAY MULTIPLIER

High performance multiplier circuits are an essential part of modern microprocessors [Schmookler et al. 1999] [Trong et al. 2007] and digital signal processors [Tian et al. 2002]. To achieve high throughput and low latency, most high performance chips use some form of booth encoded array multiplication hardware [Booth 1951]. The array multiplier architecture requires a large number of tokens to be in flight at the same time. Each multiplication operation produces a number of partial products which are then added together to produce the final product. In terms of its usefulness to a wide-range of applications and significant circuit complexity, a high throughput array multiplier is a good candidate to effectively highlight the trade-offs between PCeHB and our proposed pipeline templates. In this case study, we focus on improving energy-efficiency by packing considerable logic within each pipeline stage, even at the cost of incurring throughput degradation of up to 25% compared to PCeHB style pipelines.

We implemented an 8x8-bit radix-4 booth-encoded array multiplier (at the transistor level) using PCeHB pipelines to act as our baseline. Figure 20 shows the top-level specification of our 8x8-bit multiplier. The top part of Figure 20 shows the partial product generation for the array multiplier. Each of the *Y* inputs is in a radix-4 format. The multiplicand bits are used to generate the booth control signals for each partial product row. Since a PCeHB pipeline can only compute a small amount of logic, each of the rectangular boxes labeled *PP* represents a separate pipeline stage. The booth

control signals and multiplier input bits are sent from one pipeline stage to another, while each pipeline stage produces a two bit partial product.

The second half of Figure 20 shows the order in which the partial products are produced and summed up. The horizontal dotted lines separate different time periods. Each of the dotted polygons represent a separate PCeHB pipeline stage. The entries inside each polygon represent the inputs which are added together to produce the sum and carry outputs for the next pipeline stage. *PP* stands for two-bit partial product entry, *C'* corresponds to sign bit for each partial product row, *SS* stands for two-bit sum output from a previous stage, and *C* stands for a single-bit carry output from a previous stage sum computation. The final product bits are generated in a bit-skewed fashion, indicated by the symbol *RR*. Hence, we need to add slack-matching buffers on the outputs as well as some of the inputs to optimize the multiplier throughput [Cummings et al. 1994]. For simplicity, we do not show these slack-matching buffers in Figure 20. Our baseline multiplier is highly pipelined but contains very little logic in each pipeline stage. While this helps to achieve a very high throughput of 18 transitions per cycle, there is a large handshake overhead per each pipeline stage.

To quantify the energy efficiency and other characteristics of our proposed low-handshake pipeline templates, we implemented similar full transistor level 8x8-bit radix-4 booth-encoded array multipliers using N-P and N-Inverter pipeline templates. Figure 21 shows an overview of N-P pipelines and their logic stacks for the 8x8-bit array multiplier. Both N-P pipelines have four stages of logic. The first stage of the first pipeline generates all partial product entries. This is clearly a big power saving, as booth control signals and multiplier inputs need to be generated only once and not for each separate pipeline block as in the PCeHB implementation. Each dotted polygon represents a logic stack and not a separate pipeline stage, which leads to very high logic density in each pipeline block. Each *RR*, *SS*, and *C* signal represents a single output channel, which translates into 14 outputs for the first N-P pipeline block and 4 outputs for the second N-P block. The N-Inverter pipeline implementation, not shown due to space constraints, requires twice as many pipeline stages as N-P implementation since no effective logic computation is performed in its PMOS pull-up stacks. However, the rest of the design is similar to N-P pipeline implementation with considerable logic within each pipeline stage.

In contrast to the large number of fine-grain pipeline blocks in the PCeHB implementation, we only need two N-P and four N-Inverter pipeline stages to implement the bulk of 8x8-bit multiplication logic. The inputs to the first pipeline for both N-P and N-Inverter implementations are four radix-4 multiplier bit entries and booth control signals for all rows, which are generated separately using PCeHB style pipelines. Since PCeHB pipelines follow a four phase handshake protocol, we use four phase to single-track conversion templates similar to those in [Ferretti and Beerel 2002] but with a few modifications. Due to space constraints, we do not discuss the conversion templates. For pipeline blocks with more than nine outputs, we use wide NOR completion detection scheme. For outputs destined for the same pipeline block, we only track the neutrality of one of the outputs going to the second pipeline. This optimization greatly reduces the complexity of *_RST* circuitry, reduces power consumption, and increases the throughput by up to 6.3% for our proposed pipeline templates. To highlight the seamless integration of N-P and N-Inverter pipelines within any four phase handshake environment, we convert the resultant product outputs into four phase *1-of-4* encoding.

6.1. Evaluation of asynchronous pipeline templates using 8x8-bit Multiplier

The transistors in our baseline PCeHB multiplier implementation and our proposed N-P and N-Inverter pipeline implementations were sized using standard transistor siz-

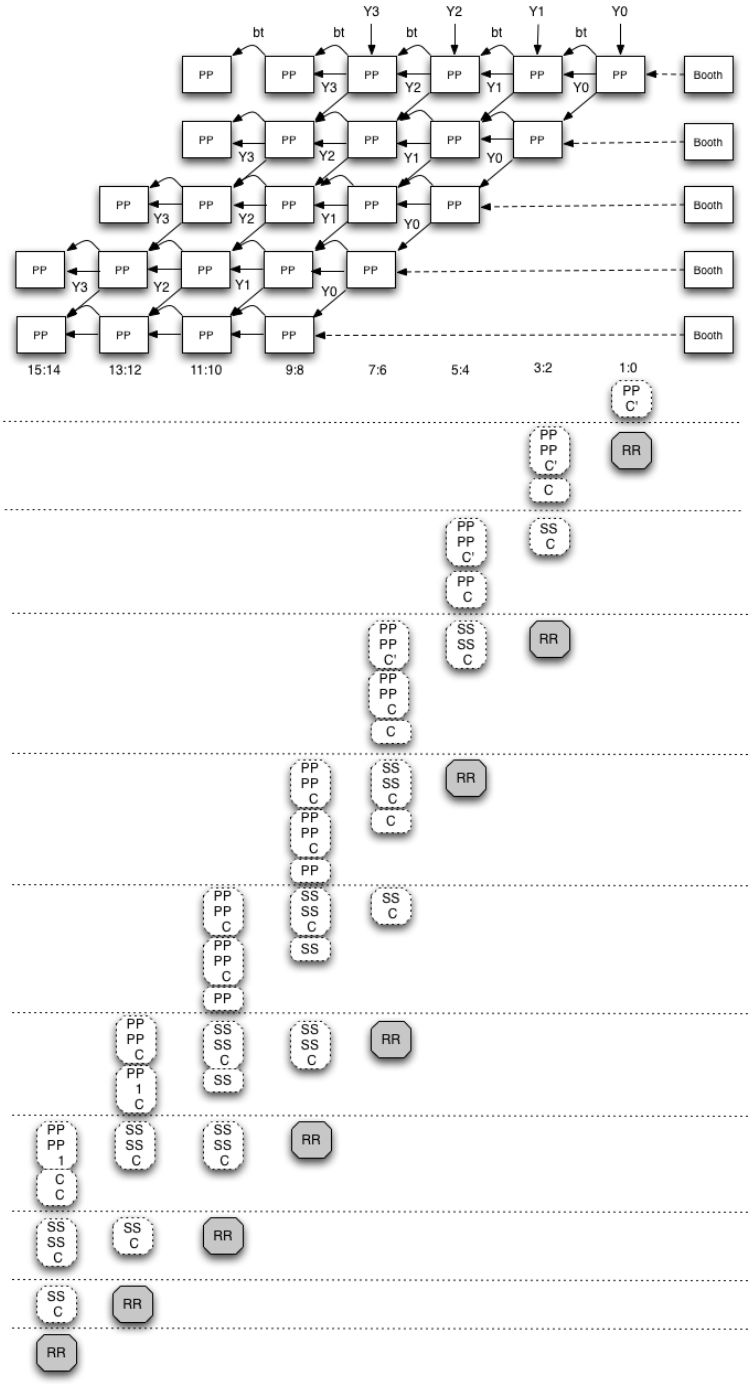

Fig. 20. 8x8-bit multiplier architecture using PCeHB pipelines

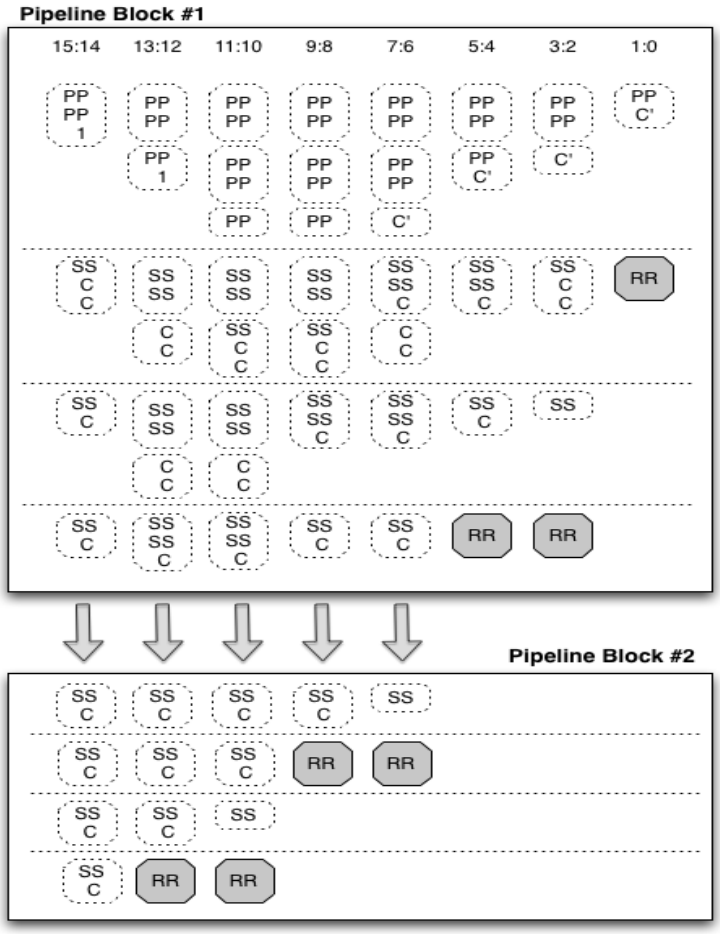

Fig. 21. 8x8-bit multiplier using N-P pipelines

ing techniques [Weste and Harris 2004]. The slow and power-consuming state-holding completion-elements were restricted to a maximum of three inputs at a time. Keepers and weak feedback inverters were added for each state-holding gate to ensure that charge would not drift even if the pipeline were stalled in an arbitrary state.

Since HSIM/HSPICE simulations do not account for wire capacitances, we included additional wire load in the SPICE file for every gate in the circuit. Based on prior experience with fabricated chips and post-layout simulation, we have found that our wire load estimates are conservative, and predicted energy and delay numbers are typically 10% higher than those from post-layout simulations. Our simulations use a 65nm bulk CMOS process at the typical-typical (TT) corner. Test vectors are injected into the SPICE simulation using a combined VCS/HSIM simulation, with Verilog models that implement the asynchronous handshake in the test environment. All simulations were carried out at the highest-precision setting.

Figure 22 shows the power-consumption breakdown of our proposed pipeline templates. In contrast to the PCeHB pipelines, which consume over 69% power in handshake overheads, the handshake and completion detection logic accounts for only 26%

of the total power in our proposed pipelines. The elimination of validity and neutrality detection logic for a large number of intermediate nodes in each pipeline is the main reason for the reduction of handshake related overheads.

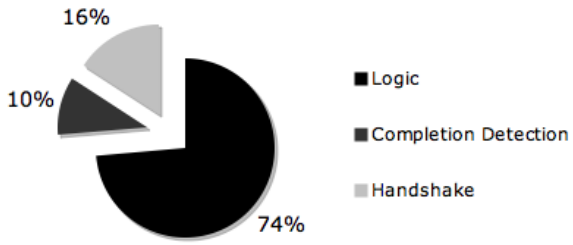

Fig. 22. Power consumption breakdown of N-P and N-Inverter pipelines

To fully quantify and evaluate our proposed pipeline templates, we simulated all three 8x8-bit array multiplier implementations across a wide range of voltages. All experimental results presented in this section include the explicit overhead of conversion templates. These templates convert input tokens from four phase protocol to single-track protocol and the outputs from single-track protocol back to four-phase protocol.

The throughput and energy consumption results for all three pipeline implementations with data points corresponding to 0.6V to 1.1V at 0.1V intervals plotted from left to right in Figure 23. As stated earlier, the N-Inverter and N-P implementations were designed from energy efficiency perspective while allowing throughput degradation of up to 25% compared to PCeHB design. To ensure fair comparison, the N-P implementation used a higher staticizer strength to yield similar noise margin as PCeHB and N-Inverter implementations. To minimize handshake circuitry, each N-Inverter and N-P pipeline block was packed with considerable logic computations and produced a large number of outputs, which reduced overall throughput. Hence, in terms of throughput, the PCeHB pipeline implementation yields the best results across all voltages. But this performance improvement comes at the cost of 45.4% and 59.5% higher energy per operation compared to the N-Inverter and N-P pipeline implementations respectively. Another key observation from Figure 23 is that for any single throughput target, be it in low throughput range such as 400-500 MHz or in high throughput range such as 1.3-1.5 GHz, our proposed templates consume far less energy per operation than the PCeHB implementation.

The fact that our proposed pipelines worked across a vast voltage range without requiring any transistor re-sizing highlights the robustness of our proposed templates. The experimental results include the power consumed in templates that are required to convert the inputs from four phase protocol to single-track protocol and the outputs from single-track protocol to four phase protocol.

The energy savings are largely due to:

— The massive reduction in the handshake circuitry because of the elimination of validity and neutrality detection gates for all internal nodes.

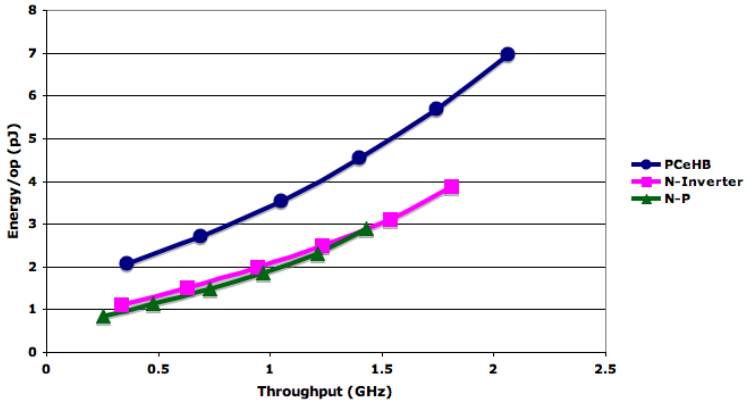

Fig. 23. 8x8-bit multiplier throughput vs energy for three different pipeline styles

— The sharing of inputs and intermediate outputs within a same pipeline block. In a PCeHB implementation, the inputs and outputs are copied from one stage to another and are subjected to separate validity and neutrality detection checks within each pipeline block.
— The use of a more energy-efficient completion detection scheme.

To consider performance and energy together, we use two metrics: energy-delay product and energy-delay^2 product as shown in Figure 24. The results are normalized to the PCeHB implementation. The N-Inverter and N-P pipelines reduce the energy-delay product by 38.5% and 44% respectively. For energy-delay^2 product, N-Inverter implementation yields a 30.3% reduction and N-P pipelines result in 22.2% reduction when compared to the PCeHB implementation.

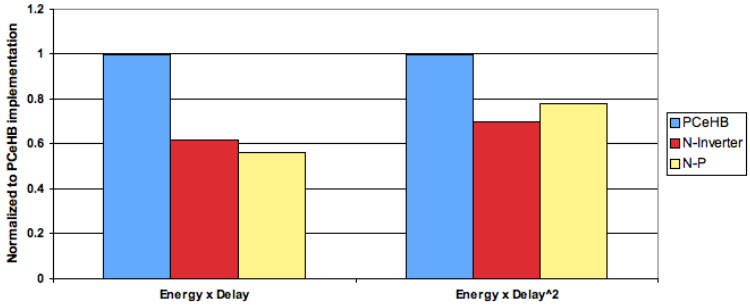

Fig. 24. 8x8-bit Multiplier energy-delay analysis for three different pipeline styles

The N-Inverter and N-P implementations reduce the overall multiplier latency by 20.2% and 18.7% respecitvely as shown in Table II. These two pipeline templates can pack significant amount of logic within a single pipeline block, which reduces the total number of pipeline stages required and hence results in latency reduction. Although, N-Inverter implementation requires twice as many pipeline stages as N-P implementation, it results in a 1.85% lower overall latency. This could be attributed to the use slower pull-up logic stacks in N-P templates.

Table II. 8x8-bit Array Multiplier Latency

Pipeline Style	Latency
PCeHB	663 ps
N-Inverter	529 ps
N-P	539 ps

In terms of the total transistor count, the N-Inverter and N-P implementations use 42.2% and 54.2% less transistors respectively than the PCeHB implementation as shown in Table III. The total transistor width in N-Inverter and N-P designs is 35.6% and 46% less respectively than that in the PCeHB implementation. This huge saving in the transistor count and width can be directly attributed to the packing of more logic stacks within a single pipeline block and the elimination of handshake logic for all intermediate nodes.

Table III. 8x8-bit Array Multiplier Transistor Count and Width

Pipeline Style	No. of Transistors	Width (μm)
PCeHB	17083	5290
N-Inverter	9864	3402
N-P	7819	2853

The choice of a particular pipeline implementation represents a design trade-off. Critical factors such as target throughput, logic complexity, power budget, latency range, total transistor count, noise margins, and timing robustness will have to be taken into account simultaneously before choosing a particular pipeline implementation. The N-P and N-Inverter templates represent a good energy efficient alternative to QDI templates, especially for logic computations which require a large number of inputs or outputs or those with multiple intermediate logic stages. We envision these circuits being used for large chunks of local logic (e.g. an array multiplier in a floating point unit) wrapped with QDI interfaces, rather than globally.

7. SUMMARY

We propose two energy-efficient pipeline templates for high throughput asynchronous circuits. These two templates, named N-P and N-Inverter pipelines, use single-track handshake protocol. Each pipeline contains multiple stages of logic. The handshake overhead is greatly minimized by eliminating validity and neutrality detection logic gates for all input tokens as well as for all intermediate logic nodes. Both of these templates can pack significant amount of logic within each pipeline block, while still maintaining a fast cycle time of only 18 transitions. Stalls on inputs and outputs do not impact correct operation. A comprehensive noise analysis of dynamic gates within our proposed templates shows sufficient noise margins across all process corners. Since our templates introduce multiple timing assumptions, we also analyzed the timing robustness of our pipelines. A completion detection scheme based on wide NOR gates is presented, which results in significant latency and energy savings especially as the number of outputs increase.

Three separate full transistor-level pipeline implementations of an 8x8-bit Booth-encoded array multiplier are presented. Compared to the PCeHB implementation, the N-Inverter and N-P pipeline implementations reduced the energy-delay product by 38.5% and 44% respectively. The overall multiplier latency was reduced by 20.2% and 18.7%, while the total transistor width was reduced by 35.6% and 46% with N-Inverter and N-P pipeline templates respectively.

REFERENCES

BEEREL, P., LINES, A., AND DAVIES, M. 2009. Logic synthesis of multi-level domino asynchronous pipelines. Fulcrum Microsystems, U.S. patent 7,584,449 B2, 2009.

BOOTH, A. D. 1951. A signed binary multiplication technique. *Quarterly Journal of Mechanics and Applied Mathematics 4,* 2, 236–240.

CHENG, F. C. 1998. Practical design and performance evaluation of completion detection circuits. In *Proceedings of the International Conference on Computer Design, 1998*.

CUMMINGS, U. V., LINES, A. M., AND MARTIN, A. J. 1994. An asynchronous pipeline lattice-structure filter. In *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1994*.

D. FANG, J. T. AND MANOHAR, R. 2005. A high-performance asynchronous FPGA: Test results. In *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines, April 2005*.

DALLY, W. J. AND POULTON, J. 1998. *Digital Systems Engineering*. Cambridge University Press, Cambridge, UK.

FANG, D. AND MANOHAR, R. 2004. Non-uniform access asynchronous register files. In *Proceedings of IEEE International Symposium on Asynchronous Circuits and Systems, 2004*.

FERRETTI, M. AND BEEREL, P. 2002. Single-track asynchronous pipeline templates using 1-of-n encoding. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE), 2002*.

HOROWITZ, M. 2007. Scaling, power and the future of CMOS. In *Proceedings of the 20th International Conference on VLSI Design, 2007*.

LAFRIEDA, C. AND MANOHAR, R. 2009. Reducing power consumption with relaxed quasi delay-insensitive circuits. In *Proceedings of IEEE International Symposium on Asynchronous Circuits and Systems, 2009*.

LINES, A. 1995. Pipelined asynchronous circuits. M.S. thesis, California Institute of Technology.

MARTIN, A. J. 1990. *Programming in VLSI: from communicating processes to delay insensitive circuits*. Addison-Wesley.

MARTIN, A. J., LINES, A., MANOHAR, R., NYSTRÖM, M., PENZES, P., SOUTHWORTH, R., CUMMINGS, U. V., AND LEE, T.-K. 1997. The design of an asynchronous MIPS R3000. In *Proceedings of Conference on Advanced Research in VLSI, 1997*.

SCHMOOKLER, M., PUTRINO, M., MATHER, A., TYLER, J., NGUYEN, H., ROTH, C., PHAM, M., LENT, J., AND SHARMA, M. 1999. A low-power, high-speed implementation of a PowerPC microprocessor vector extension. In *Proceedings of the International Symposium on Computer Arithmetic, 1999*.

SCHUSTER, S. AND COOK, P. 2003. Low-power synchronous-to-asynhronous interlocked pipelined CMOS circuits operating at 3.3-4.5 GHz. *IEEE Journal of Solid-State Circuits 38,* 4, 622–630.

SEITZ, C. L. 1980. System timing. In *Introduction to VLSI Systems*, C. A. Mead and L. A. Conway, Eds. Addison-Wesley.

SHEIKH, B. R. AND MANOHAR, R. 2010. An operand-optimized asynchronous IEEE 754 double-precision floating-point adder. In *Proceedings of IEEE International Symposium on Asynchronous Circuits and Systems, 2010*.

SUTHERLAND, I. AND FAIRBANKS, S. 2001. GasP: A minimal FIFO control. In *Proceedings of IEEE International Symposium on Asynchronous Circuits and Systems, 2001*.

TIAN, Z., YU, D., AND QIU, Y. 2002. A high effective algorithm of 32-bit multiply and MAC instructions' VLSI implementation with 32x8 multiplier-accumulator in DSP applications. In *Proceedings of the International Conference on Signal Processing, 2002*.

TRONG, S. D., SCHMOOKLER, M., SCHWARZ, E. M., AND KROENER, M. 2007. P6 binary floating-point unit. In *Proceedings of the International Symposium on Computer Arithmetic, 2007*.

VAN BERKEL, K. AND BINK, A. 1996. Single-track handshake signalling with application to micropipelines and handshake circuits. In *Proceedings of the International Symposium on Asynchronous Circuits and Systems, 1996*.

WESTE, N. AND HARRIS, D. 2004. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley.

WILLIAMS, T. E. 1991. Self-timed rings and their application to division. Ph.D. thesis, Computer Systems Lab, Stanford University.