The Design of High-Performance Dynamic Asynchronous Pipelines: Lookahead Style

Montek Singh and Steven M. Nowick

***Abstract*—A new class of asynchronous pipelines is proposed, called *lookahead pipelines (LP),* which use dynamic logic and are capable of delivering multi-gigahertz throughputs. Since they are asynchronous, these pipelines avoid problems related to high-speed clock distribution, such as clock power, management of clock skew, and inflexibility in handling varied environments. The designs are based on the well-known PS0 style of Williams and Horowitz as a starting point, but achieve significant improvements through novel protocol optimizations: the pipeline communication is structured so that critical events can be detected and exploited earlier. A special focus of this work is to target extremely fine-grain or gate-level pipelines, where the datapath is sectioned into stages, each consisting of logic that is only a single level deep. Both dual-rail and single-rail pipeline implementations are proposed. All the implementations are characterized by low-cost control structures and the avoidance of explicit latches. Post-layout SPICE simulations, in 0.18-μm technology, indicate that the best dual-rail LP design has more than twice the throughput (1.04 giga data items/s) of Williams' PS0 design, while the best single-rail LP design achieves even higher throughput (1.55 giga data items/s).**

***Index Terms*—Asynchronous, dynamic logic, elastic pipelining, gate-level pipelines, latch controllers, micropipelines, pipeline processing, precharge logic.**

I. Introduction

IN THE CURRENT thrust towards multi-gigahertz systems, there are increasing design challenges to managing clock distribution [3]. Asynchronous design, which replaces global clocking with local handshaking, has the potential to make high-speed design more feasible.

This paper introduces a new class of asynchronous dynamic pipelines, called *lookahead pipelines (LP)*, which are capable of delivering multi-gigahertz throughputs. Since they are asynchronous, these pipelines avoid issues related to the distribution of a high-speed clock: e.g., wasteful clock power and management of clock skew. Moreover, the use of local handshaking, instead of global clocking, imparts the ability of localized flow control to the asynchronous pipeline: the number of data items in the pipeline is allowed to vary over time, depending on the operating speeds of the interfaces and stages. Thus, the proposed pipelines have a natural *elasticity* [37]. In contrast, except for some recent approaches [14], traditional synchronous pipelines typically do not provide elasticity. Finally, the inherent flexibility of asynchronous components allows an asynchronous pipeline to interface with varied environments operating at different rates; thus, asynchronous pipeline styles are useful for the design of system-on-a-chip (SoC) and reusable intellectual property (IP), and also have been shown to gracefully accommodate dynamic voltage scaling [17].

Asynchronous design has recently attracted industry due to its potential for low power consumption, low electromagnetic interference, high speed, and robust interfacing. Several asynchronous products are now commercially available. In particular, Philips has developed a fully asynchronous 80C51 microcontroller for use in its commercial pagers and smartcards [9], which exhibits significantly lower power consumption and electromagnetic noise emission. Handshake Solutions, a Philips subsidiary, in partnership with ARM, has developed the first commercial synthesizable clockless ARM family processor [1]. Intel's Pentium-4 microprocessor uses asynchronous self-resetting logic in the ALU pipeline, enabling it to run twice as fast as the rest of the CPU. The Sun UltraSPARC IIIi microprocessor employs asynchronous first-input first-output (FIFO) queues in its memory controller in order to deskew data received from external memory, and thereby provide a high-speed and flexible memory interface [38]. Fulcrum Microsystems, an asynchronous startup company, has developed a commercial largely-asynchronous crossbar switch for high-speed networking [5]. In addition, a number of substantial experimental chips have also been produced, both in industry and in academia, including several asynchronous microprocessors [10], [21]; a fast asynchronous Pentium instruction length decoder developed by Intel, exhibiting three times the throughput and only half the power consumption of a corresponding clocked implementation [27]; a fast asynchronous Huffman decoder for compressed-code embedded processors [2]; and a hybrid synchronous-asynchronous finite-impulse response (FIR) filter for disk-drive read channels developed in collaboration with IBM Research [33] that exhibited 50% lower latency and 15% higher throughput than a comparable clocked implementation.

The focus of this paper is on a key enabling technology to make high-speed asynchronous systems practical: the design of high-throughput asynchronous pipelines. While the lookahead pipelines introduced in this paper can be used for a wide range of applications, a special focus of this work is to target extremely high throughput. To this end, very fine-grain, or "gate-level," pipelining is used: the datapath is sectioned into narrow pipeline stages, each containing logic that is only a single level deep.

Manuscript received July 29, 2005; revised June 22, 2006. This work was supported in part by the National Science Foundation (NSF) under Award CCR-97-34803, NSF ITR Award CCR-00-86036, NSF ITR Award CCR-00-86007, a grant from the New York State Microelectronics Design Center (MDC), a gift from Sun Microsystems, Inc., a UNC Junior Faculty Development Award, a grant from UNC University Research Council, and an IBM Faculty Development Award. Preliminary versions of this paper appeared in the International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), 2000, and in a Ph.D. dissertation by M. Singh.

M. Singh is with the Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599 USA (e-mail: montek@cs.unc.edu).

S. M. Nowick is with the Department of Computer Science, Columbia University, New York, NY 10027 USA (e-mail: nowick@cs.columbia.edu).

Digital Object Identifier 10.1109/TVLSI.2007.902205

A further feature of this paper is that explicit latches are not needed: with careful sequencing of control, the dynamic gates themselves naturally provide an implicit latching functionality (e.g., immunity of data to precharging inputs). The removal of explicit latches also provides benefits of reduced critical delays, smaller chip area, and lower power consumption. Finally, the proposed control circuits are simple, small and fast, thus further minimizing the overhead of fine-grain pipelining.

A. Overview and Contribution

Lookahead pipelines are derived from the well-known PS0 asynchronous dynamic pipeline style of Williams *et al.* [39], but provide significantly higher throughput by using a variety of novel optimizations to reduce the impact of inter-stage handshaking delays. In particular, all of the proposed optimizations follow a strategy of *anticipation:* enabling critical pipeline events to be observed, and reacted to, earlier by using a more sophisticated pipeline structure. In particular, three optimizations over Williams' approach are proposed: 1) *early evaluation,* in which a stage receives control signals not only from the next stage, but also from stages *further down* the pipeline; 2) *early done,* in which a stage indicates to its previous stage that it is *about to* complete an action, instead of signaling after it has completed that action; and 3) a hybrid approach which combines these two schemes. The net result is a significant reduction in overall pipeline cycle time.

These optimizations were applied to both single-rail and dual-rail pipelines, resulting in several new lookahead pipeline structures. The proposed designs rely on the use of modest timing assumptions to obtain speed improvements. Thus, lookahead pipelines are not delay-insensitive or speed-independent, but are time-constrained, as are a number of existing asynchronous pipeline designs (including Williams' PS0). However, the timing constraints are one-sided and, in practice, easily satisfiable.

Post-layout SPICE simulations were performed on five new lookahead FIFO designs: three dual-rail and two single-rail. These designs were laid out in a 0.18-μm TSMC technology, and simulated with extracted wire and layout parasitics at 1.8-V nominal supply voltage, 300-K temperature, and a normal process corner. Simulation results are quite promising. The best dual-rail FIFO (LP2/1) achieved a throughput of 1.04 giga data items/s, which is *104% higher* than that of Williams' original PS0 design, in the same technology (0.18 μm), while maintaining comparable latency. The best single-rail FIFO ($\mathrm{LP_{sr}}2/1$) obtained even higher throughput: 1.55 giga data items/s.

While the single-rail designs obtained higher throughput than the dual-rail ones, the latter may hold a performance advantage in future fabrication technologies. In particular, dual-rail pipelines rely on delay-insensitive datapath encoding and robust completion detection [15], which allow for correct operation in the presence of arbitrary datapath delays. In contrast, single-rail pipelines use delay matching [15], which often requires additional timing margins to account for delay uncertainties. In future technologies [13], it is possible that some dual-rail designs may be faster than single-rail ones if the timing margins required in the latter exceed the cost of completion detection in the former. Therefore, this paper provides both single-rail and dual-rail styles, in order to support current as well as future technologies.

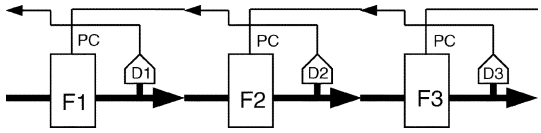


Fig. 1. Block diagram of a PS0 pipeline.

B. Paper Organization

The remainder of this paper is organized as follows.

Section II focuses on the design of *dual-rail* asynchronous lookahead pipelines. First, detailed background on Williams' classic PS0 dual-rail asynchronous pipeline style [39] is provided, followed by a review of several more recent dual-rail approaches. Then, three distinct dual-rail lookahead pipelines are introduced, LP3/1, LP2/2, and LP2/1, with details provided on their circuit structure, operation, performance analysis, and timing constraints. Finally, a targeted comparison with another pipeline style by Williams, PA0, highlights the new contributions.

Section III focuses on the design of *single-rail* asynchronous lookahead pipeliness. After comparison to related synchronous and single-rail asynchronous styles, two distinct single-rail lookahead pipelines are introduced, $\mathrm{LP_{sr}}2/2$ and $\mathrm{LP_{sr}}2/1$. For each, similar details are provided as with the dual-rail designs.

Finally, Section IV presents detailed post-layout simulation results and Section V presents conclusions.

Note: A companion paper [31] introduces an alternative dynamic asynchronous style called *high-capacity (HC)* pipelines. This latter paper also discusses some general issues in asynchronous pipeline design, such as the interfacing to varied environments and the handling of wide datapaths; it also includes an in-depth technical comparison of LP and HC, as well as their relationship to some of the fastest recent asynchronous pipelines [28], [35]. However, the current paper is fully self-contained: it includes a broad presentation of related work, and it can be understood without the advanced material. The interested reader is referred to [31] for the comparison of LP and HC, and other advanced asynchronous pipeline topics.

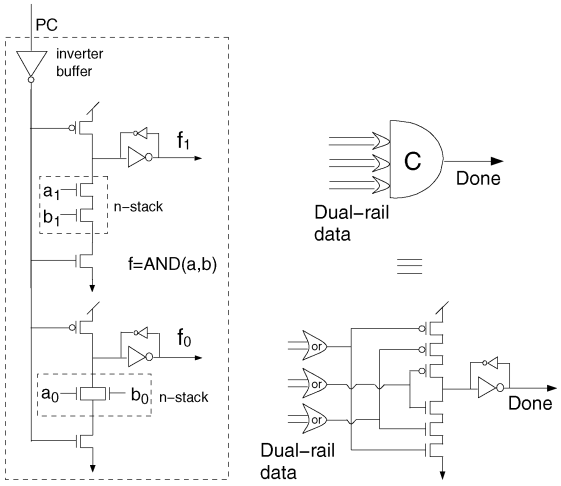
II. Dual-Rail Lookahead Pipelines

A. Background: Williams' PS0 Pipeline

This subsection gives detailed background on Williams' PS0 dual-rail pipeline [39], which is the starting point for the new lookahead pipeline style.

1) Structure and Implementation: Fig. 1 shows the structure of Williams' PS0 pipeline. Each pipeline stage is composed of a dual-rail function block and a completion detector. The completion detectors indicate validity or absence of data at the outputs of the associated function block.

Dual-rail is a commonly-used scheme to implement an asynchronous datapath [15], [29]. In dual-rail, two wires (or rails) are used to implement each bit. The wires indicate both the *value* of the bit, and also its *validity.* The encodings of 01 and 10 correspond to valid data values 0 and 1, respectively. The encoding 00 indicates the reset or spacer state with no valid data and 11 is an unused (i.e., illegal) encoding. Encodings on the datapath



(a) A dual–rail AND gate in precharge logic

(b) A dual–rail completion detector

Fig. 2. Dual-rail logic and completion detection.

typically alternate between valid values and the reset state. Since the datapath itself indicates the validity of each bit, dual-rail is effective in designing asynchronous datapaths which are highly robust in the presence of arbitrary delays.

Each *function block* is implemented using dynamic logic. The precharge/evaluate control input PC of each stage is tied to the output of the next stage's completion detector. Since a precharge logic block can hold its data outputs even when its inputs are reset, it also provides some of the functionality of an implicit latch. Therefore, a PS0 stage has no explicit latch. Fig. 2(a) shows how a dual-rail two-input AND gate, for example, would be implemented in dynamic logic; the dual-rail pair, f_1 and f_0, implements the AND of two dual-rail inputs, a_1a_0 and b_1b_0.

Each *completion detector* verifies the completion of every computation and precharge of its associated function block. Validity, or nonvalidity, of data outputs is checked by ORing the two rails for each individual bit, and then combining all of the results using a Muller C-element [see Fig. 2(b)] [37]. A C-element [39] is a basic asynchronous state-holding element. The output of an n-input C-element is high when all inputs are high, is low when all inputs are low, and otherwise holds its previous value. It is typically implemented as a dynamic gate, with a series stack in both pull-up and pull-down, and an inverter on the output (with weak feedback inverter attached to maintain state). The latency of such a C-element, like typical dynamic gates, is two levels of logic.

2) Protocol: The coordination of pipeline control in a PS0 pipeline is quite simple, with two "backwards" synchronization events per cycle. Stage N is precharged when stage N+1 finishes evaluation, and stage N evaluates when stage $N+1$ finishes its reset. (More precisely, stage N enters the "precharge release" phase after stage $N+1$ finishes its reset; stage N will then only evaluate after it receives valid data inputs from stage $N-1$.) This protocol ensures that each pair of consecutive data tokens is always separated by a reset token (or "spacer"), where the data bits in a stage are reset to all 00 values.

The complete cycle of events for a PS0 pipeline stage is derived by observing how a single data token flows through an initially empty pipeline. The sequence of events from one evaluation by stage 1 to the next is as follows: (i) Stage 1 evaluates; then (ii) stage 2 evaluates; then (iii) stage 2's completion detector detects completion of evaluation; and then (iv) stage 1 precharges. At the same time, after completing step (ii), there is a step (iii)′ stage 3 evaluates; then step (iv)′ stage 3's completion detector detects completion of evaluation and initiates the precharge of stage 2; then step (v) stage 2 precharges; and finally (vi) stage 2's completion detector detects completion of precharge, thereby releasing the precharge of stage 1 and enabling stage 1 to evaluate once again. Thus, there are six events in a complete cycle for a stage from one evaluation to the next.

3) Analytical Cycle Time and Latency: The complete cycle for a pipeline stage consists of three evaluations, two completion detections, and one precharge. The analytical pipeline cycle time T_{PS0}, is therefore

$$T_{\mathrm{PS0}} = 3 \cdot t_{\mathrm{Eval}} + 2 \cdot t_{\mathrm{CD}} + t_{\mathrm{Prech}} \tag{1}$$

where t_{Eval} and t_{Prech} are the evaluation and precharge times for each stage, and t_{CD} is the delay through each completion detector.[1,2]

The per-stage *forward latency* [39] L is defined as the time it takes the first data token, in an initially empty pipeline, to travel from the output of one stage to the output of the next stage. For PS0, the forward latency is simply the evaluation delay of a stage

$$L_{\mathrm{PS0}} = t_{\mathrm{Eval}}. \tag{2}$$

4) Other Related Approaches: The classic work on dual-rail asynchronous pipelines is by Williams and Horowitz [39], as previously introduced. A number of variant implementation styles are introduced, such as PC0, PA0, PS1, etc., with a special focus on dynamic logic. An advantage of several of these designs is that they have *low forward latency*. This beneficial feature was exploited in the design of two commercial processors developed at HAL Computers, which employed floating-point dividers based on Williams' pipeline styles [40]. However, in spite of their advantages, all of these pipelines are *throughput-limited* [39]: the cycle time for each stage to process a data item has significant overhead.

Martin *et al.* [20] and Lines [18] present the design of a complete microprocessor using very fine-grain pipelining techniques similar to Williams'. The pipeline circuits are based on the conservative and robust quasi-delay-insensitive (QDI) model, yet the circuits have relatively good performance. While several distinct pipeline styles are introduced, the most commonly used ones have cycle times that are 18 gate delays or greater. Other QDI styles include those by Ozdag and Beerel [24], which improve upon the performance of [18], [20] through the use of concurrency reduction to simplify the design of the dynamic function blocks (i.e., reduced stack size). Finally, Ferretti and Beerel [7] combine the robustness of delay-insensitive encoding with the efficiency of the single-track handshaking of [35] and introduce single-track 1-of-N pipelines.

[1]To simplify the presentation, this paper will sometimes assume that all the pipeline stages have the same evaluation delay and the same precharge delay, and that all the completion detectors are equally fast. More realistic HSPICE simulations are presented in the Results section.

[2]Additionally, the analysis here assumes that the datapaths are narrow enough not to require additional buffers for the control signals. For wide datapaths, these buffers may be required, and their delays will impact the cycle time. The handling of wide datapaths is discussed in detail in a companion paper [31].

B. Overview of Lookahead Pipelines

The new lookahead pipelines use Williams' PS0 style as a starting point, but incorporate several optimizations to reduce the overall cycle time. The key optimization strategy is one of *anticipation*: anticipating the generation of certain critical events based on a richer set of observations of the state of the pipeline. Related strategies have been used recently in somewhat different contexts: from timing-optimal variants of Sutherland's basic micropipeline [6], to more aggressive timing optimizations based on *kiting* [22] and *relative timing* [34], [4].

In this paper, two specific optimizations are proposed: 1) *early evaluation* and 2) *early done*. In "early evaluation," a pipeline stage uses control information not only from the subsequent stage, but also from stages *further down* the pipeline. This information is used to give the stage a headstart on its evaluation phase. In the second optimization, "early done," a stage signals to its previous stage when it is *about to* precharge or evaluate, rather than after it has completed those actions. This information is used to give a pipeline stage a headstart both on its evaluation phase as well as its precharge phase. The net result of applying these two optimizations is a significant reduction in pipeline cycle time, and consequently, a dramatic increase in throughput, with no net increase in latency.

The remainder of this section presents three new dual-rail pipeline styles—LP3/1, LP2/2, and LP2/1—in detail. The LP3/1 pipeline uses early evaluation, the LP2/2 pipeline uses early done, and LP2/1 is a hybrid which combines both optimizations. In Section III, two new single-rail lookahead pipelines will be presented, using variants of these dual-rail approaches.

C. LP3/1 Pipeline

LP3/1 is the first of the new dual-rail lookahead pipelines. In this design style, an *early evaluation* protocol is used, in which a pipeline stage receives control information not only from the subsequent stage, but also *from its successor*. As a result, LP3/1 pipelines have shorter cycles than Williams' PS0 pipelines: a complete cycle for a stage of an LP3/1 pipeline consists of four events, as opposed to six events for a stage in a PS0 pipeline.

The new pipeline derives its name from the fact that three out of the four events in every stage's cycle fall in its evaluation phase, and one event falls in its precharge phase. Using this terminology, Williams' PS0 pipelines would be called 3/3.

It should be noted that an earlier pipeline style by Williams, called PA0 [39], has some similarities in the underlying protocol to the LP3/1 pipeline. However, PA0 uses a different implementation which is not able to fully take advantage of the new protocol; as a result, the critical path delays in PA0 are significantly longer than in the proposed LP3/1 pipeline. In addition, the two styles differ in the manner in which their stages interact. A detailed presentation of the LP3/1 design is given in this subsection, while key differences between PA0 and LP3/1 are highlighted in Section II-G.

1) Basic Structure: Fig. 3 shows the block diagram of an LP3/1 pipeline. Unlike a PS0 stage, an LP3/1 stage has two distinct control inputs. The first control input PC comes from the next stage, as in PS0. The second control input EVAL comes from the completion detector *two stages ahead.* This second input is the key to achieving a shorter cycle time.

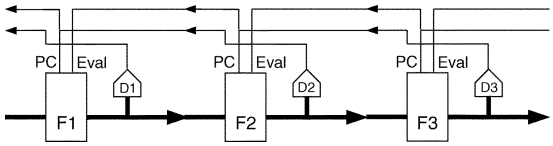


Fig. 3. Block diagram of an LP3/1 pipeline.

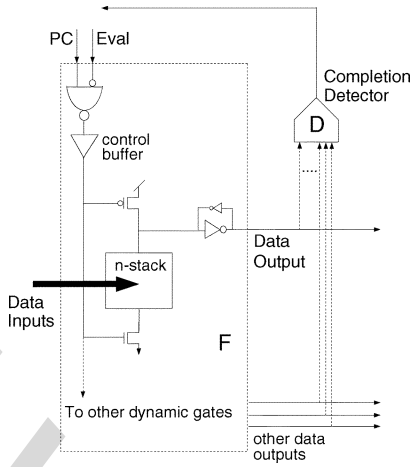


Fig. 4. Implementation of an LP3/1 stage.

2) Protocol and Implementation: The key idea of the new protocol is that stage N can evaluate as soon as stage $N+1$ has *started precharging,* instead of waiting until stage $N+1$ has *completed precharging*. This approach can safely be used because a dynamic logic stage undergoing precharge is insensitive to changes on its inputs.[3] Therefore, as soon as stage $N+1$ begins to precharge, stage N can immediately proceed with its next evaluation. Note that, since stage $N+1$ begins precharging only after stage $N+2$ completes its evaluation, the new condition for evaluation is, therefore: *Evaluate stage N when stage $N+2$ completes its evaluation*. The condition for precharge remains unchanged: *Precharge stage N when stage $N+1$ completes evaluation*. Therefore, stage N needs inputs from both the completion detector of $N+1$ as well as from that of $N+2$, to implement the new optimized protocol.

Fig. 4 shows the implementation of an LP3/1 stage. For simplicity, only one dynamic logic gate is shown inside the function block, but there will typically be several dynamic gates, one pair for each bit of dual-rail output data. The figure shows how the two control inputs are combined inside the implementation of one stage: a NAND gate, with an inversion (i.e., bubble) on one of its inputs, merges PC and EVAL into a single signal which controls the precharge and evaluation of the stage. Evaluation is enabled when either EVAL is asserted high or PC is deasserted low, or both. The former condition EVAL $=$ high corresponds to stage $N+2$ completing its computation (i.e., stage $N+1$ *starting* its precharge; see Fig. 3). The latter condition PC $=$ low is identical to the evaluation condition of PS0; its role in LP3/1 is explained in the following. Precharge is enabled when both PC is asserted high and EVAL is deasserted low.

3) Comparison of Operation: LP3/1 Versus PS0: In LP3/1, there are now two distinct control inputs for a stage N, which are outputs from stages $N+1$ and $N+2$. The *precharge phase*

[3]In general, this property is only true of *fully controlled* (or "footed") dynamic logic. All of the pipelines presented in this paper use fully controlled dynamic logic. Therefore, this property of precharge can indeed be exploited.

of stage N begins after stage $N+1$ is done evaluating (PC asserted high), much like PS0. However, *the precharge phase is now shortened:* precharge terminates when stage $N+2$ is done evaluating (EVAL asserted high). In contrast, in PS0, precharge terminates only once stage $N+1$ is done precharging. At this point, stage N enters its *evaluate phase.*

In LP3/1, the evaluate phase continues until two distinct conditions hold, which drive the stage into the next precharge: 1) stage $N+1$ has completed evaluation (as in PS0: PC asserted high) and 2) stage $N+2$ has completed precharging (EVAL deasserted low). The NAND gate in Fig. 4 (with a bubble on its EVAL input) directly implements these two conditions.

Interestingly, during LP3/1's evaluate phase, the early EVAL control signal from stage $N+2$ may be *nonpersistent:* it may be deasserted low even before stage N has had a chance to evaluate its new data! However, one-sided timing constraints of Section II-F are imposed to insure a correct evaluate phase: PC $=$ low will arrive in time to *take over* control of the evaluate phase, which will then be maintained until stage N has completed evaluating its inputs (as in PS0).

4) Analytical Cycle Time and Latency: The complete cycle of events for a single LP3/1 stage, say stage 1, from one evaluation until the next can be derived from a simulation of the design in Fig. 3: (i) Stage 1 evaluates; (ii) stage 2 evaluates; (iii) stage 2's completion detector detects completion of stage 2's evaluation; and then (iv) stage 1 precharges. At the same time, after completing step (ii), there is a step (iii)$'$ stage 3 evaluates; and step (iv)$'$ stage 3's completion detector detects completion of stage 3's evaluation, thereby enabling two subsequent events: both the precharge of stage 2 and the next evaluation of stage 1 ("early evaluation"). Thus, there are only four events in the complete cycle for an LP3/1 stage, from one evaluation to the next, down from the six events in PS0. This reduction by two events has been brought about by eliminating the two events of stage 2's precharge phase from stage 1's cycle.

The analytical cycle time of an LP3/1 pipeline is, therefore

$$T_{\mathrm{LP3/1}} = 3 \cdot t_{\mathrm{Eval}} + t_{\mathrm{CD}} + t_{\mathrm{NANDB}} \tag{3}$$

where t_{NANDB} is the delay through the NANDB gate for the early evaluation signal. Thus, the LP3/1 cycle time is $t_{\mathrm{Prech}} + t_{\mathrm{CD}} - t_{\mathrm{NANDB}}$ shorter than that of PS0.

The per-stage forward latency is simply the evaluation delay of a stage, as in PS0

$$L_{\mathrm{LP3/1}} = t_{\mathrm{Eval}}. \tag{4}$$

5) Miscellaneous: Loading Issues: The previous analysis does not take into account the fact that completion detectors in LP3/1 will be somewhat slower than those in PS0 due to greater capacitive loads. The increased loading in an LP3/1 pipeline is due to the need to fork off a stage's "done" signal to two preceding stages instead of one. Higher loading typically causes logarithmic overheads to the power, area, and latency of the completion detectors [36]. To quantify this impact, Section IV provides more refined results based on post-layout HSPICE simulations. These simulation results indicate that, in spite of the overhead due to increased loading, the LP3/1 pipeline in practice still has significantly higher throughput than Williams' PS0 pipeline.

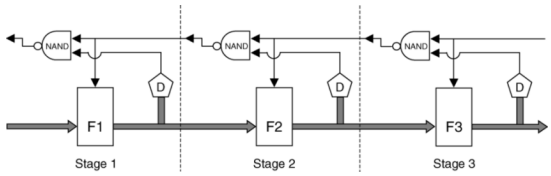


Fig. 5. Enhanced LP3/1 implementation.

The requirement to fork off the acknowledgment to two preceding stages is likely to cause greater overheads in future technologies because of the current trend in increasing interconnect delays [13]. To address this issue, an enhanced LP3/1 implementation is now introduced, which uses a simple restructuring of the pipeline to significantly reduce wiring length.

6) Enhanced Version of LP3/1: Simplifying the Stage Interfaces: As a final enhancement, an alternative version of the LP3/1 pipeline is now presented, with simpler interfaces between pipeline stages: the same protocol is used, but now there is direct communication only between *adjacent* stages. There are two benefits of the simpler stage interface: 1) greater ease of interfacing with the environment and 2) reduced wiring loads. This new version requires only minor circuit modifications.

Fig. 5 shows the structure of the enhanced LP3/1 pipeline. The implementation can be derived from the basic LP3/1 pipeline structure of Figs. 3–4 in two steps.

First, the basic LP3/1 pipeline is modified to simplify interstage communication. The only change is to the NAND gate. As shown in Fig. 5, the NAND's PC input is the same as before, but it now receives a different EVAL input: EVAL is no longer directly tapped off the completion detector of stage $N+2$; it now comes from the NAND gate output of stage $N+1$. Since EVAL now undergoes an inversion through stage $N+1$'s NAND gate, the inverter "bubble" on the NAND input is also eliminated. Thus, the function computed by the NAND gate remains unchanged. The net impact is that stage N now effectively communicates only with its immediate neighbor, stage $N+1$.

Second, through a simple redrawing of stage boundaries, stage N's NAND gate can be pushed into stage $N+1$, with no net change in functionality. Combined with the earlier optimization, the final result is that stage N communicates on *only a single wire* with its neighbor $N+1$, rather than on two wires with two neighbors (as in the original LP3/1).

Thus, through a small modification of the control circuitry, and a redrawing of stage boundaries, a cleaner interface and reduced wiring are obtained.

There are two advantages of the simplified stage interface. First, the resulting LP3/1 pipeline structure can now be more easily interfaced with the environment. In particular, the new pipeline expects only one acknowledgment from the right environment, and produces only one acknowledgment for the left environment. In contrast, the basic LP3/1 implementation of Fig. 3 requires each interface with the environment to carry two distinct acknowledgment signals. Second, wiring loads are reduced due to shorter overall wire lengths between stages, which can be a significant benefit in future fabrication technologies [13].

D. LP2/2 Pipeline

The second new dual-rail lookahead pipeline is called LP2/2. The key feature is that a pipeline stage is now allowed to signal its previous stage when it is "*about to* evaluate (or precharge)"

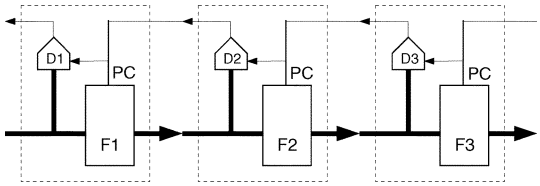


Fig. 6. Block diagram of an LP2/2 pipeline.

instead of after it has completed those actions. Thus, this pipeline uses an *early done* protocol.

An LP2/2 pipeline has a shorter analytical cycle time than that of a PS0 pipeline: like LP3/1, the cycle of an LP2/2 stage consists of four events. Moreover, LP2/2 pipelines have another desirable feature: unlike the basic LP3/1 design, the stages have only one control input as opposed to two, thereby reducing loading on the completion detectors.

1) Structure and Implementation: Fig. 6 shows a block diagram of an LP2/2 pipeline. The stages are similar to those used in PS0, but with one key difference: completion detectors are now placed *before* their functional blocks. The idea is to signal to the previous stage when the current stage is *about to* evaluate (or precharge).

A modified completion detector is needed in order to generate the "early done" signal. The completion detector now receives an extra input: the stage's PC control input. The functionality of the detector is as follows. The detector asserts *Done* (high) when the stage is about to evaluate: 1) the stage is enabled to evaluate (PC deasserted low) and 2) it has valid dual-rail inputs. The new completion detector deasserts *Done* (low) simply when the stage is about to precharge: PC is asserted (high). Thus, the done signals are produced in parallel with the actual precharge or evaluation by the associated function block, instead of after its completion. Note that these conditions are asymmetric: only a single condition (PC asserted high) enables the stage to precharge and its completion detector to indicate that precharge is complete, whereas two conditions must hold for the stage to evaluate and its completion detector to indicate completion of evaluation (PC deasserted low *and* valid data at stage inputs).

This new completion detector is implemented using an *asymmetric C-element* (see Fig. 7) [8]. An asymmetric C-element (abbreviated "aC") has three types of inputs: those that are marked "+", those marked "-", and a third type that is unmarked. The output of the aC is set high when all the unmarked inputs and all the "+" inputs go high. Conversely, the aC output is reset low when all the unmarked inputs and all the "-" inputs go low. For any other input combination, the aC holds its output value.

The particular asymmetric C-element of Fig. 7 is a degenerate special case: it has no "-" inputs, and has only one unmarked input. Clearly, it can be regarded as simply a precharged dynamic gate, which deasserts *Done* (low) whenever PC is asserted (high). For good performance, the depth of pull-down networks should typically not exceed four. Therefore, if the datapath consists of more than four bits of dual-rail data, the aC element is typically decomposed into multiple levels of logic.

2) Protocol and Performance Analysis: A complete cycle of events for stage 1 can be traced in Fig. 6. From one evaluation to the next, it consists of four events: (i) stage 1 evaluates; (ii) stage 2's completion detector detects "early done" of stage 2's evaluation (in parallel with stage 2's evaluation), thereby asserting the precharge control of stage 1; and then (iii) stage 1 precharges. At the same time, after completing step (i), there is a step (ii)′ stage 2 evaluates; then step (iii)′ stage 3's completion detector detects "early done" of stage 3's evaluation (in parallel with stage 3's evaluation), thereby asserting the precharge control of stage 2; and finally step (iv) stage 2's completion detector detects "early done" of stage 2's precharge (in parallel with stage 2's precharge), thereby enabling stage 1 to evaluate once again in the next step.

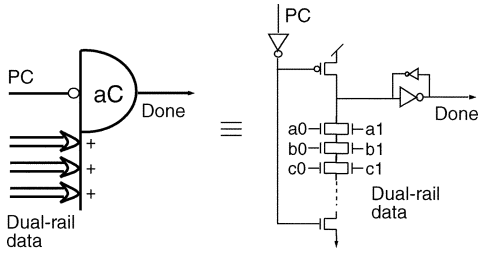


Fig. 7. LP2/2 completion detector.

Therefore, the analytical cycle time of an LP2/2 pipeline is

$$T_{\mathrm{LP2/2}} = 2 \cdot t_{\mathrm{Eval}} + 2 \cdot t_{\mathrm{CD}} \tag{5}$$

which is $t_{\mathrm{Eval}} + t_{\mathrm{Prech}}$ shorter than that of PS0. The latency of an LP2/2 pipeline is identical to that of PS0 and LP3/1

$$L_{\mathrm{LP2/2}} = t_{\mathrm{Eval}}. \tag{6}$$

E. LP2/1 Pipeline

LP2/1 is the final new dual-rail lookahead pipeline. This design is basically a hybrid: it combines both the "early evaluation" optimization of LP3/1 and the "early done" optimization of LP2/2. Consequently, an LP2/1 pipeline has the shortest analytical cycle time of the three LP design styles: a stage's cycle consists of only three events.

1) Structure and Implementation: Fig. 8 shows the structure of an LP2/1 pipeline. Each stage uses information from two succeeding stages (as in LP3/1), and also employs early completion detection (as in LP2/2). As in LP3/1, there are two control inputs for every stage, PC and EVAL, which are merged inside the stage in the same manner as was done for LP3/1, i.e., as shown in the basic implementation of Fig. 4 or, alternatively, as in the enhanced version of Fig. 5.

2) Protocol and Performance Analysis: A complete cycle of events for stage 1 can be traced in Fig. 8. From one evaluation to the next it consists of three events: (i) Stage 1 evaluates; (ii) stage 2's completion detector detects "early done" of stage 2's evaluation (in parallel with stage 2's evaluation), thereby asserting the precharge control of stage 1; and then 3) stage 1 precharges. At the same time, after completing step (i), there is a step (ii)′ stage 2 evaluates; and finally step (iii)′ stage 3's completion detector detects the "early done" of stage 3's evaluation, thus enabling the evaluation of stage 1 in the next step. Therefore, the analytical cycle time of an LP2/1 pipeline is

$$T_{\mathrm{LP2/1}} = 2 \cdot t_{\mathrm{Eval}} + t_{\mathrm{CD}} + t_{\mathrm{NANDB}} \tag{7}$$

which is $t_{\mathrm{Eval}} + t_{\mathrm{Prech}} + t_{\mathrm{CD}} - t_{\mathrm{NANDB}}$ shorter than that of PS0. Once again, the latency of an LP2/1 pipeline is identical to that of PS0

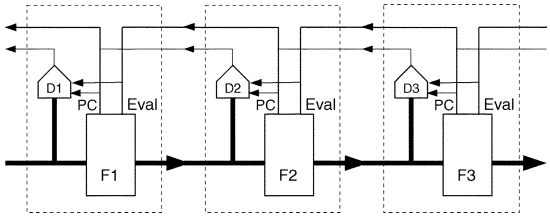


Fig. 8. Block diagram of an LP2/1 pipeline.

$$L_{\mathrm{LP2/1}} = t_{\mathrm{Eval}}. \tag{8}$$

F. Timing Constraints for Dual-Rail Lookahead Pipelines

Each of the proposed dual-rail LP designs requires certain one-sided timing constraints to be satisfied for correct operation. This subsection presents analytical equations for these timing constraints. Detailed HSPICE simulations appear in Section IV, which verify that all of these timing constraints can easily be satisfied in practice.

1) Precharge Width: LP3/1 and LP2/1 pipelines have a shorter precharge phase than PS0 pipelines, since the start of the evaluation phase is advanced by two time steps.[4] For correct precharge, the precharge of a stage should be complete before the stage receives the evaluation signal, EVAL $=$ high. That is, a *minimum precharge width* must be enforced.

The appropriate timing constraint for the LP3/1 pipeline is now formally derived. Using as reference the instant stage $N+1$ finishes evaluating, stage N receives the precharge signal at time $t_{\mathrm{CD}_{N+1}\uparrow}$, where $t_{\mathrm{CD}_{N+1}\uparrow}$ is the time it takes for stage $N+1$'s completion detector to switch high.[5] Also, from the same reference, the EVAL signal for stage N goes high at time $t_{\mathrm{Eval}_{N+2}} + t_{\mathrm{CD}_{N+2}\uparrow}$. Therefore, for correct precharge, the precharge width t_{Prech_N} must satisfy

$$t_{\mathrm{Prech}_N} \leq t_{\mathrm{Eval}_{N+2}} + (t_{\mathrm{CD}_{N+2}\uparrow} - t_{\mathrm{CD}_{N+1}\uparrow}).$$

This equation, while correct, is overly conservative. In practice, there are two factors that can result in a more relaxed constraint. First, note that the precharge phase includes additional margin, equal to t_{INV}, due to the extra inverter at the input of the NAND gate. Second, there is a further added margin, also equal to t_{INV}, due to the output inverter inside the dynamic gate in stage N. In particular, it is actually sufficient that the precharge pulse width be long enough to precharge the internal node of the dynamic gate; the delay of the output inverter need not be included in the minimum time required to complete precharge. As a result, a tighter precharge width constraint is

$$t_{\mathrm{Prech}_N} - t_{\mathrm{INV}} \leq t_{\mathrm{Eval}_{N+2}} + (t_{\mathrm{CD}_{N+2}\uparrow} - t_{\mathrm{CD}_{N+1}\uparrow}) + t_{\mathrm{INV}}. \tag{9}$$

Assuming all stages are similar and both transitions of a completion detector are equally fast, the constraint becomes

$$t_{\mathrm{Prech}} \leq t_{\mathrm{Eval}} + 2 \cdot t_{\mathrm{INV}}. \tag{10}$$

This condition is satisfied if a stage's precharge is not slower than, or only up to $2 \cdot t_{\mathrm{INV}}$ slower than, its evaluation. This constraint effectively means that "precharge should be fast enough" and, in practice, is usually easily satisfied. (Similar constraints on the precharge width for LP2/1 pipelines can be derived.)

2) Safe Takeover: For correct operation of the evaluation phase in LP3/1 and LP2/1, it is required that the "takeover" signal, PC $=$ low, arrive at the inputs of the NAND gate before the *nonpersistent* EVAL goes low. This requirement is needed to ensure that the control maintains a glitch-free evaluation phase whenever early evaluation is used (i.e., LP3/1 and LP2/1). While only LP3/1 is considered here, similar constraints can be derived for LP2/1 as well.

The following analysis calculates the time at which stage N's EVAL is deasserted low, and the time at which stage N's takeover signal appears. The reference time 0 is set at the point when stage $N+2$ has just completed evaluation, which will start the early evaluation of stage N. The time instant when EVAL for stage N is deasserted low (from stage $N+2$) is given by

$$t_{\mathrm{Eval}_{N+3}} + t_{\mathrm{CD}_{N+3}\uparrow} + t_{\mathrm{Prech}_{N+2}} + t_{\mathrm{CD}_{N+2}\downarrow}. \tag{11}$$

Similarly, the takeover signal PC of stage N is asserted low (from stage $N+1$) at time

$$t_{\mathrm{CD}_{N+2}\uparrow} + t_{\mathrm{Prech}_{N+1}} + t_{\mathrm{CD}_{N+1}\downarrow}. \tag{12}$$

Therefore, to maintain uninterrupted evaluation, the takeover should arrive at least a setup time, t_{setup}, before EVAL is deasserted. This timing constraint, however, once again benefits from the additional inverter delay t_{INV}, which the EVAL signal must go through at the inputs of the NAND gate before it is deasserted. Therefore, the safe takeover constraint is

$$\begin{aligned} &t_{\mathrm{CD}_{N+2}\uparrow} + t_{\mathrm{Prech}_{N+1}} + t_{\mathrm{CD}_{N+1}\downarrow} + t_{\mathrm{setup}} \\ &\quad \leq t_{\mathrm{Eval}_{N+3}} + t_{\mathrm{CD}_{N+3}\uparrow} + t_{\mathrm{Prech}_{N+2}} + t_{\mathrm{CD}_{N+2}\downarrow} + t_{\mathrm{INV}}. \end{aligned} \tag{13}$$

Assuming all stages are similar, this constraint becomes

$$t_{\mathrm{Eval}} + t_{\mathrm{INV}} \geq t_{\mathrm{setup}}. \tag{14}$$

This constraint is also easily satisfied since the setup time of a transistor is typically less than the evaluation time of an entire pipeline stage.

3) Input Hold Time: Finally, in LP2/2 and LP2/1, the data inputs to an evaluating stage must be held valid long enough for the stage to complete evaluation, before the inputs are reset. That is, the "early done" path through the completion detector must not reset the previous stage before the current stage has effectively absorbed its data inputs. If the time for a precharge-released dynamic gate to absorb its inputs is t_{hold}, then the input hold time constraint is

$$t_{\mathrm{hold}} \leq t_{\mathrm{CD}_N\uparrow} + t_{\mathrm{Prech}_{N-1}}. \tag{15}$$

Assuming all stages are identical, this constraint becomes

$$t_{\mathrm{CD}\uparrow} \geq t_{\mathrm{hold}} - t_{\mathrm{Prech}_{N-1}}. \tag{16}$$

[4]The "1" in their designation indicates precisely this fact: their precharge phase is only 1 "unit" long, where a "unit" is approximately the amount of time for one stage evaluation, or one stage reset, or one completion detection.

[5]Note that, here, for a more precise analysis, a distinction is made between $t_{\mathrm{CD}_{N+1}\uparrow}$ and $t_{\mathrm{CD}_{N+1}\downarrow}$, which are the delays associated with detection of stage $N+1$'s evaluation and reset, respectively.

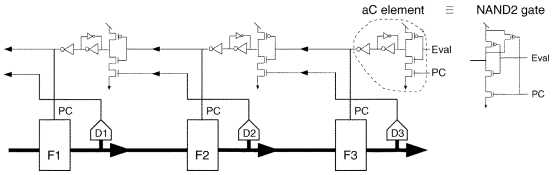


Fig. 9. Block diagram of a PA0 pipeline.

According to this constraint, the completion detectors cannot be "too fast." This constraint is also easily satisfied in practice.

G. Detailed Comparison: LP3/1 Versus Williams' PA0 Pipelines

In [39], Williams briefly introduces an optimized pipeline design, called PA0. Although so far only Williams' PS0 pipeline has been discussed, the PA0 pipeline style is also directly relevant to LP3/1: both PA0 and LP3/1 designs effectively use control inputs from two subsequent stages, instead of one. In what follows is an in-depth comparison between the two approaches highlighting the key differences.

The structure of Williams' PA0 pipeline is shown in Fig. 9. As in LP3/1, each pipeline stage receives two control inputs, PC and EVAL. The PC input of stage N is the completion signal from stage $N+1$. The EVAL input of N is derived from the completion detector of stage $N+2$.

The PA0 pipeline operates as follows. Stage N is driven into evaluation as soon as stage $N+1$ *starts* to precharge; thus, the pipeline allows early evaluation, much like LP3/1. The "trigger signal" which causes the start of evaluation is EVAL $=$ low. Stage N is precharged when $N+1$ is done evaluating (PC $=$ high) and $N+2$ is done precharging (EVAL $=$ high). This stage's control is implemented by an asymmetric C-element, shown in Fig. 9.

There is a key difference in the controls of PA0 and LP3/1, although they seem to be functionally quite similar: Williams' PA0 control uses an asymmetric C-element, whereas the LP3/1 control uses a NAND2 gate. The basic idea in the LP3/1 approach is to simplify the control circuitry by imposing an additional *timing requirement* on the interaction between stages. The net result is not only a simpler control circuit in LP3/1, but also the removal of *two inverters* in series from the critical path.[6]

The replacement of PA0's C-element by LP3/1's NAND2 gate is now justified. A simple timing assumption must be made on the arrival of inputs to the NAND2 gate. In each design, an early evaluation of stage N is enabled by the trigger signal EVAL $=$ low, which is an input to the control. In PA0, the C-element holds this value, and evaluation persists, until the desired precharge phase begins. In contrast, in LP3/1, while EVAL $=$ low (input to the NAND2) correctly enables an early evaluation of stage N, this trigger signal is nonpersistent: the control output could incorrectly get deasserted. Therefore, for correct operation, a takeover signal (PC $=$ low) is required to arrive at the gate input, *before* EVAL is deasserted (see Section II-F).

Once this timing assumption on the arrival of PC is satisfied, the C-element can safely be replaced by the combinational gate. As shown in Fig. 9, the NAND2 gate is identical to the logic portion of the asymmetric C-element, but with one extra parallel pMOS transistor, controlled by PC. This modification makes the gate fully complementary; hence its staticizer (i.e., the feedback pair of inverters) can be deleted. In addition, the inverter after the C-element is eliminated to maintain correct logic polarity.

The net effect of this modification is a significantly shortened cycle time for the LP3/1 pipeline compared to PA0, because PA0's critical path for stage N goes through two C-elements: the C-element of stage $N+1$ and that of stage $N+2$. In particular, two levels of logic are saved because the C-elements are changed to combinational NAND gates, and a further two inverters are eliminated to maintain correct polarity. Therefore, compared to PA0, the cycle time of LP3/1 is *four gate delays shorter.*

Interestingly, while Williams points out that the throughput of PA0 is likely to be worse than that of PS0 [39], a significant throughput improvement has been observed in LP3/1 (see Section IV). This improvement is brought about partially because of the elimination of the extra inverters.

III. Single-Rail Lookahead Pipelines

While dual-rail datapaths allow variable-speed completion and have been effectively used in a number of applications, the area penalties are sometimes unacceptable in practice and the power overhead may be large [26]. Single-rail design has much wider applicability in the synchronous world, and several asynchronous groups have recently moved from dual- to single-rail design [26]. Therefore, the focus of this section is on single-rail lookahead pipelines. The target design style is a commonly-used approach called "bundled-data" [29] (described in detail in the following), where synchronous function blocks can be used along with attached matched delays.

Two new single-rail lookahead pipelines are introduced. The first design $\mathrm{LP_{sr}}2/2$, essentially adapts the dual-rail LP2/2 design of Section II-D to single-rail datapaths. As in the former design, an *early done* optimization is used. The second design, $\mathrm{LP_{sr}}2/1$, adds the further improvement of *early evaluate*, similar to the LP2/1 design of Section II-E. The two new pipelines operate correctly under simple and easily satisfiable one-sided timing constraints.

A. Previous Work

1) Synchronous Pipelines: Several novel synchronous pipelines have been proposed for high-throughput applications. In *wave pipelining*, multiple waves of data are propagated between two latches [41], [19]. However, this approach requires much design effort, from the architectural level down to the layout level, for accurate balancing of path delays (including data-dependent delays), and remains highly vulnerable to process, temperature and voltage variations. Therefore, wave pipelining is not widely used for general applications. Other aggressive approaches include *clock-delayed domino* [42], *skew-tolerant domino* [11], and *self-resetting circuits* [23]. These all require complex timing constraints which are difficult to verify; they also lack elasticity [37]—i.e., the ability to accommodate environments with dynamically varying speeds—and still require high-speed global clock distribution.

2) Single-Rail Asynchronous Pipelines: The classic single-rail asynchronous pipelines, introduced by Sutherland, are called *micropipelines* [37]. This style uses elegant transition-signaling (two-phase) control, but has slow and complex capture-pass latches which limit performance. Most recent

[6]Theoretically, for an inverting output from the C-element, the internal node of the C-element could be tapped. However, in our experiments, electrical simulations indicated this not to be very reliable.

designs—including our own—can be regarded as derived from this pipeline, using variations on both latches and protocol.

Several variants of micropipelines have been proposed using alternative latching or control structures. However, each has its limitations. A pipeline style of Yun *et al.* [43] uses dual-edge-triggered D-flipflops instead of capture-pass latches; while this design is simpler, it suffers from significant delays due to the overhead of the complex flipflops. The approaches by Day and Woods [6] and Furber and Liu [8] use four-phase control instead of two-phase control. While the new protocols allow use of simpler latches, the controllers themselves are quite complex. Finally, a design by Kol and Ginosar [16] uses double latches per stage to provide greater concurrency; however, the additional latches result in significant area and throughput overheads.

The fastest single-rail asynchronous designs reported so far are the IPCMOS pipeline from IBM [28] and the GasP pipeline from Sun [35]. Both approaches are targeted to single-rail bundled datapaths [29] that use static logic. Each style provides very high throughputs through use of novel complex control structures and aggressive circuit techniques. Throughputs of 3.3 GHz in 0.18 μm were reported for IPCMOS in an IBM proprietary silicon-on-insulator technology, at the normal process corner.[7] In contrast, the experimental results for the proposed lookahead pipelines are for a standard bulk-CMOS process, which is very likely significantly more conservative than the proprietary IBM process used for IPCMOS. For GasP, a throughput of 1.5 GHz has been reported in 0.35-μm technology.

The GasP and IPCMOS styles have several fundamental limitations when compared with the proposed lookahead style. In particular, the design methodologies of GasP and IPCMOS inherently require: 1) *non-standard signaling schemes;* (e.g., single-track and pulse-mode); 2) *special-purpose VLSI control circuit structures* (e.g., tri-stated wires, bidirectional communication on the same wire, and self-resetting gates); and 3) *two-sided timing constraints.* In contrast, lookahead pipelines have significantly simpler one-sided timing requirements, do not rely on fine-grain transistor sizing, and can be implemented using standard cells for control and standard dynamic gates for the datapath. A more in-depth comparison with GasP and IPCMOS styles appears in [31].

Finally, an asynchronous implementation of wave pipelining was proposed by Hauck *et al.* [12]. This approach makes two key contributions over synchronous wave pipelining. First is the addition of request signals (i.e., bundling), which helps closely track process, temperature, and voltage variations in the datapath. Second is the use of a special type of self-resetting dynamic logic that helps rein in path delay variations. As a result, quite high throughputs were reported: 1.5 GHz in 0.35-μm technology. However, this approach also has several drawbacks. Because no acknowledge signals are used, the pipeline only operates in a "feed forward" mode, without any backwards flow control, as is also true with synchronous wave pipelines (unlike lookahead pipelines): data items can be lost if the destination environment is momentarily stalled. Further, the approach still requires much design effort for balancing path delays: dummy transistors and capacitances are often required to be inserted at specific nodes, and the fanin and fanout of gates must be carefully controlled.

[7]Throughputs of up to 4.5 GHz have been reported, but these are only for extreme process cases ($L_{\text{eff}} = -2\sigma$ and low V_t).

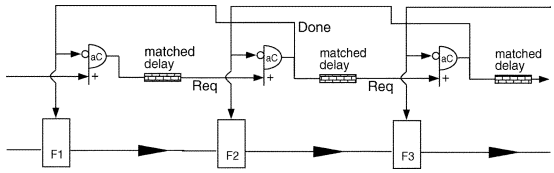


Fig. 10. Block diagram of an $\mathrm{LP_{sr}}2/2$ pipeline.

B. $\mathrm{LP_{sr}}2/2$ *Pipeline*

1) Structure and Implementation: Fig. 10 shows the structure of a single-rail $\mathrm{LP_{sr}}2/2$ pipeline. It has a number of similarities to the dual-rail LP2/2 pipeline of Section II-D. Each pipeline stage has a function block and a completion generator. The function block alternately evaluates and precharges. The completion generator produces a *Done* signal to indicate the evaluation or precharge by the function block. In turn, *Done* is communicated to two stages: 1) to the previous stage, as an "acknowledgment" and 2) to the next stage, as a "request."

However, unlike the earlier-presented lookahead pipelines, $\mathrm{LP_{sr}}2/2$ uses single-rail encoding [29] to implement the asynchronous datapath. In this scheme, a preexisting single-rail function block can be used, such as from a synchronous dynamic logic library, without the need for costly completion detectors. In addition, a single extra "bundling signal" is added, to match or exceed the worst-case block delay, and which serves as a completion signal.

In particular, the control signal *Req* indicates arrival of new data at a stage's inputs: a high value of *Req* indicates that the previous stage has finished evaluation; a low *Req* indicates that the previous stage has completed precharge. For correct operation, a simple timing constraint must be satisfied: *Req* must arrive *after* the data inputs to the stage have stabilized. This requirement is met by inserting a "matched delay" which is greater than or equal to the worst case delay through the function block. An advantage of this approach is that the datapath itself can use standard single-rail (synchronous style) blocks. A disadvantage is that adequate timing margins must often be added to ensure that the latency through the matched delay is sufficient to allow the datapath to settle before the request is generated. However, in practice, since the matched delay is localized to a given block, and tends to track with local temperature and voltage, tighter margins can be used than in synchronous design, and different stages can have substantially different delays (unlike in synchronous systems).

In practice, there are several ways to implement a matched delay. One technique is to simply use an inverter chain, or a chain of transmission gates; the number of gates and their transistor sizing determines the total delay. A more accurate technique duplicates the worst-case critical path of the logic block, and uses that as a delay line. By placing the matched delay in physical proximity to the associated datapath block, it becomes possible for the delay element to track process, temperature, and voltage variations quite well [26]. Bundled data has been widely used in asynchronous design, including in a commercial Philips 80C51 asynchronous microcontroller [9].

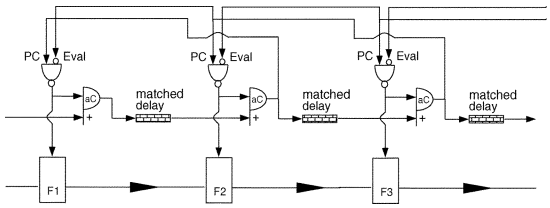


Fig. 11. Block diagram of an $\mathrm{LP_{sr}}2/1$ pipeline.

2) Protocol: The $\mathrm{LP_{sr}}2/2$ pipeline protocol is very similar to that of PS0. When a stage is done evaluating, it signals to the previous stage to initiate its precharge operation. Similarly, when a stage is done precharging, it signals to the previous stage to release precharge (i.e., enable the evaluate phase). In addition, the *Done* signal is passed forward to the next stage, indicating that the evaluation (or precharge) is complete.

Two new optimizations are used that take advantage of the innate property of dynamic logic, in this single-rail design. The first is aimed at reducing the cycle time; the second is aimed at decreasing latency.

The first optimization is to *tap off* the *Done* signal for the previous stage from *before* the matched delay, instead of after the matched delay. In spirit, this optimization is similar to the "early done" of LP2/2, and the same justification applies. For footed dynamic logic, it is safe to indicate *completion of precharge* as soon as the precharge cycle *begins:* during precharge, the stage is effectively isolated from changes at its inputs. Likewise, for a dynamic stage, it is safe to indicate *completion of evaluation* as soon as the stage begins to evaluate on valid inputs; once the stage has evaluated, its outputs are effectively isolated from a reset at the inputs.[8] This early tap-off optimization has a significant impact on the pipeline performance: the overall cycle time is reduced by an amount equal to *two* matched delays.

The second optimization is to allow an *early precharge release*. In dynamic logic, the function block can be precharge-released *before* new valid inputs arrive, provided at least that the stale inputs have precharged, and that new inputs will change only monotonically. Once data inputs arrive, the function block starts computing its data outputs. Similarly, once the matched bundling input arrives, the bundling output (*Req*) is also generated. Thus, in the $\mathrm{LP_{sr}}2/2$ design, precharge release of the function block is completely decoupled from the arrival of valid inputs. In contrast, in several other recent asynchronous pipeline designs, the function block is precharge-released only after the bundling input has been received [8]. This latter requirement typically adds extra gates to the critical forward path in the pipeline: the arrival of data now triggers the pipeline control to put the dynamic block into its evaluation phase, instead of the dynamic block's already being precharge released and ready to receive data. In contrast, the optimization used in $\mathrm{LP_{sr}}2/2$ results in a reduction in the forward latency.

3) Analytical Cycle Time and Latency: A complete cycle of events for a stage in $\mathrm{LP_{sr}}2/2$ is quite similar to that in PS0. From one evaluation of stage 1 to the next, the cycle consists of four events: (i) Stage 2 evaluates; (ii) stage 2 evaluates; (iii) stage 3 evaluates, asserting the precharge input for stage 2; and finally, (iv) stage 2 precharges, enabling stage 1 to evaluate once again.

The following notation is used for the various delays associated with this pipeline:

- t_{Eval} time for a stage evaluation;
- t_{gC} delay of the completion generator (asymmetric C-element);
- t_{delay} magnitude of the matched delay. For correct operation, $t_{\mathrm{delay}} \geq t_{\mathrm{Eval}} - t_{\mathrm{gC}}$. For ideal operation, assume that t_{delay} is no larger than necessary, $t_{\mathrm{delay}} = t_{\mathrm{Eval}} - t_{\mathrm{gC}}$. Often, for gate-level pipelines, the t_{gC} delay already matches the evaluation delay: $t_{\mathrm{gC}} \geq t_{\mathrm{Eval}}$; for such pipelines, the matched delay is unnecessary: $t_{\mathrm{delay}} = 0$.

In this notation, the delays of steps (i) and (ii) in the stage's traced cycle, are each t_{Eval}. The delays of steps (iii) and (iv) are each t_{gC}. Therefore, the analytical pipeline cycle time is

$$T_{\mathrm{LP_{sr}}2/2} = 2 \cdot t_{\mathrm{Eval}} + 2 \cdot t_{\mathrm{gC}}. \tag{17}$$

The per-stage forward latency of the pipeline is

$$L_{\mathrm{LP_{sr}}2/2} = t_{\mathrm{Eval}}. \tag{18}$$

C. $\mathrm{LP_{sr}}2/1$ Pipeline

$\mathrm{LP_{sr}}2/1$ is the final single-rail lookahead pipeline. This design can be thought of as a derivative of LP2/1 or LP3/1, but adapted to a single-rail bundled datapath.

1) Structure and Implementation: Fig. 11 shows the structure of an $\mathrm{LP_{sr}}2/1$ pipeline. Each stage has a function block and a completion generator identical to those of an $\mathrm{LP_{sr}}2/2$ single-rail pipeline. However, a stage receives control inputs not only from the subsequent stage (PC), but also from its successor (EVAL). Much like LP2/1 and LP3/1, the second control input enables "early evaluation."

2) Protocol and Performance Analysis: The sequencing of control is similar to that in LP2/1 or LP3/1. A complete cycle of events from one evaluation of stage 1 to the next, consists of three events: (i) Stage 1 evaluates; (ii) Stage 2 evaluates; and finally, (iii) stage 3 evaluates, triggering "early evaluation" of stage 1. Thus, the cycle time is

$$T_{\mathrm{LP_{sr}}2/1} = 2 \cdot t_{\mathrm{Eval}} + t_{\mathrm{gC}} + t_{\mathrm{NANDB}}. \tag{19}$$

The analytical cycle time is somewhat better than that of $\mathrm{LP_{sr}}2/2$, because $t_{\mathrm{NANDB}} < t_{\mathrm{gC}}$.

Once again, forward latency is simply t_{Eval}

$$L_{\mathrm{LP_{sr}}2/1} = t_{\mathrm{Eval}}. \tag{20}$$

D. Timing Constraints for Single-Rail Lookahead Pipelines

As with the dual-rail designs, each of the single-rail LP designs requires certain one-sided timing constraints to be satisfied for correct operation. The derivation of these constraints is very similar to that of Section II-F for the dual-rail designs. Once again, HSPICE simulations of Section IV verify that all of these timing constraints can be easily satisfied in practice.

1) Precharge Width: $\mathrm{LP_{sr}}2/1$ pipelines, much like LP2/1, require a timing constraint to enforce an adequate precharge

[8]More precisely, completion of evaluation can be safely indicated at time t_{hold} after the start of evaluation (cf. Section II-F3).

width. The precharge of stage N is started by the *Done* of stage $N+1$, and terminated by the *Done* of stage $N+2$. The former event occurs a time $t_{\mathrm{gC}_{N+1}}$ after stage N has finished its evaluation. The latter event occurs a time $t_{\mathrm{Eval}_{N+1}} + t_{\mathrm{gC}_{N+2}}$ from the same reference. Similar to dual-rail pipelines, the precharge of the single-rail variant has the benefit of two more inverters (see Section II-F1). Therefore, for correct precharge, the precharge width t_{Prech_N} must satisfy

$$t_{\mathrm{Prech}_N} \leq t_{\mathrm{Eval}_{N+1}} + (t_{\mathrm{gC}_{N+2}} - t_{\mathrm{gC}_{N+1}}) + 2 \cdot t_{\mathrm{INV}}. \quad (21)$$

2) Safe Takeover: For correct operation of the evaluation phase, $\mathrm{LP_{sr}}2/1$ pipelines require a timing constraint on the arrival of the "takeover" signal, much like their dual-rail counterparts (see Section II-F2). That is, the takeover signal PC = low must arrive at the inputs to the NAND gate before the nonpersistent EVAL goes low.

The analytical equation for the timing constraint is derived in much the same manner as was done for dual-rail. The reference time 0 is set at the point when stage $N+2$ has just asserted its *Done*, which starts the early evaluation of stage N. The time when EVAL for stage N is deasserted low (from stage $\mathrm{N}+2$) is

$$t_{\mathrm{delay}_{N+2}} + t_{\mathrm{gC}_{N+3}} + t_{\mathrm{NANDB}_{N+2}} + t_{\mathrm{gC}_{N+2}}. \quad (22)$$

Similarly, the takeover signal PC of stage N is asserted low (from stage $N+1$) at time

$$t_{\mathrm{NANDB}_{N+1}} + t_{\mathrm{gC}_{N+1}}. \quad (23)$$

Therefore, to maintain uninterrupted evaluation, the takeover should arrive at least a setup time t_{setup} before EVAL is deasserted. However, similar to the dual-rail pipelines, the safe takeover of the single-rail variant benefits from the inverter delay at the input of the NAND gate (see Section II-F2)

$$t_{\mathrm{NANDB}_{N+1}} + t_{\mathrm{gC}_{N+1}} + t_{\mathrm{setup}} \leq t_{\mathrm{delay}_{N+2}} + t_{\mathrm{gC}_{N+3}} + t_{\mathrm{NANDB}_{N+2}} + t_{\mathrm{gC}_{N+2}} + t_{\mathrm{INV}}. \quad (24)$$

Assuming all stages are similar, this constraint simplifies to

$$t_{\mathrm{Eval}} + t_{\mathrm{INV}} \geq t_{\mathrm{setup}}. \quad (25)$$

In the event, however, that neighboring stages have slightly different component delays, this constraint generalizes to

$$t_{\mathrm{Eval}} + t_{\mathrm{INV}} \geq t_{\mathrm{setup}} + (\delta_{\mathrm{NANDB}} + \delta_{\mathrm{gC}}). \quad (26)$$

Here, δ_{NANDB} and δ_{gC} are the differences in the latencies of NAND gates and gC elements, respectively, of adjacent stages.

3) Input Hold Time: Both $\mathrm{LP_{sr}}2/2$ and $\mathrm{LP_{sr}}2/1$ require a constraint to ensure that the data inputs to an evaluating stage are held stable long enough for the stage to complete evaluation. That is, the path through the "early tap-off" must not reset the previous stage before the current stage has effectively absorbed its data inputs. If the time for a precharge released dynamic gate to absorb its inputs is t_{hold}, then the input hold time constraint is

$$t_{\mathrm{hold}} \leq t_{\mathrm{gC}_N} + t_{\mathrm{Prech}_{N-1}}. \quad (27)$$

IV. Results

This section presents the results of post-layout SPICE simulations of lookahead pipelines. All of the five lookahead styles are considered: the three dual-rail (LP3/1, LP2/2, and LP2/1) and the two single-rail ($\mathrm{LP_{sr}}2/2$ and $\mathrm{LP_{sr}}2/1$). Results of simulations of Williams' PS0 pipeline are also presented to serve as the base case for comparison.

A. Experimental Setup

Six 10-stage FIFOs were designed and simulated (with no logic processing): one for each of the LP styles and one for PS0 for comparison. The FIFOs were laid out using the Cadence tool suite in a 0.18-μm TSMC process. Custom cells were designed for the dynamic function blocks, as well as for all C- and asymmetric C-elements. To serve as the base case for comparison, a 4-bit PS0 FIFO was also designed. All designs were simulated, with extracted wire and layout parasitics, at 1.8-V nominal voltage supply, 300-K temperature, and a normal process corner.

The width of the datapath was chosen to be 4 bits (i.e., four data rails for the single-rail designs and eight rails for the dual-rail designs). However, as discussed in a companion paper [31], wider datapaths are usually accommodated in practice without much performance degradation through either *datapath partitioning* (i.e., splitting into narrower data streams) [20], or through *control kiting* (i.e., moving control distribution delays off of the critical path) [22]. For example, in a recent fabricated FIR filter chip, using another asynchronous dynamic pipeline style, the control kiting approach was demonstrated to be successful in handling a highly varied datapath, ranging from 30 to 216 wires in width at varying points in the pipeline [31], [33].

The experiment was carefully designed to provide a fair comparison between the different pipeline styles. In particular, identical completion detectors were used in PS0 and LP3/1.[9] Furthermore, all of the pipeline implementations were subjected to the same transistor sizing approach. In particular, devices sizes were computed manually using the method of logical effort [36], a gain-based approach. Stage ratios, i.e., the ratio of the drive strengths of consecutive gates, varied between three and four. Finally, all designs included full initialization capability.

B. Simulation Results for Dual-Rail Pipelines

Table I summarizes the results of the simulation for the dual-rail pipelines. For each of the three dual-rail styles—LP3/1, LP2/2, and LP2/1—as well as for Williams' PS0, the table lists the overall pipeline cycle time T and a breakdown of the cycle time into the following components:

t_{Eval}	time for a stage evaluation;
t_{Prech}	time for a stage precharge;

[9]Similarly, the completion detectors of LP2/2 and LP2/1 were identical to each other, though they are inherently different from those of PS0 and LP3/1.

TABLE I
PERFORMANCE OF DUAL-RAIL LOOKAHEAD PIPELINES VERSUS WILLIAMS' PS0

Pipeline Design	t_{Eval} (ns)	t_{Prech} (ns)	t_{CD} (ns)	t_{NANDB} (ns)	Cycle Time, T Analytical Formula	(ns)	Throughput Giga items per sec.	% increase over PS0
LP3/1	0.21	0.21	0.60	0.22	$3 \cdot t_{\mathrm{Eval}} + t_{\mathrm{CD}} + t_{\mathrm{NANDB}}$	1.45	0.69	**35%**
LP2/2	0.18	0.22	0.38	-	$2 \cdot t_{\mathrm{Eval}} + 2 \cdot t_{\mathrm{CD}}$	1.11	0.90	**76%**
LP2/1	0.18	0.21	0.32	0.29	$2 \cdot t_{\mathrm{Eval}} + t_{\mathrm{CD}} + t_{\mathrm{NANDB}}$	0.96	1.04	**104%**
PS0	0.21	0.21	0.57	-	$3 \cdot t_{\mathrm{Eval}} + 2 \cdot t_{\mathrm{CD}} + t_{\mathrm{Prech}}$	1.98	0.51	**Base**

TABLE II
PERFORMANCE OF SINGLE-RAIL LOOKAHEAD PIPELINES

Pipeline Design	t_{Eval} (ns)	t_{Prech} (ns)	t_{gC} (ns)	t_{NANDB} (ns)	Cycle Time, T Analytical Formula	(ns)	Throughput Giga items per sec.
$\mathrm{LP_{sr}2/2}$	0.16	0.18	0.22	-	$2 \cdot t_{\mathrm{Eval}} + 2 \cdot t_{\mathrm{gC}}$	0.76	1.31
$\mathrm{LP_{sr}2/1}$	0.16	0.18	0.20	0.15	$2 \cdot t_{\mathrm{Eval}} + t_{\mathrm{gC}} + t_{\mathrm{NANDB}}$	0.65	1.55

t_{CD} delay through the completion detector (average of the up and down transitions). This includes the delay through the buffers that amplify this signal to provide sufficient drive;

t_{NANDB} for LP3/1 and LP2/1, this is the delay through the NAND2 gate that combines the two control inputs into one (see Fig. 4).

Finally, the tables list the throughput of each pipeline in billion (i.e., giga) data items/s, as well as expresses it as a percentage improvement over the throughput of PS0.

The results indicate that the throughput of each of the dual-rail LP FIFO's is significantly higher than that of PS0. As expected, LP2/1 delivers the highest throughput of all four designs, 1.04 giga data items/s: this rate is *104% faster* than that of Williams' PS0 design (0.51 giga data items per second). Our other two designs, LP3/1 and LP2/2, also exhibited higher throughputs: 0.69 and 0.90 giga data items/s, respectively, which represent 35% and 76% improvements. That is, in agreement with our analysis of Sections II-A–II-E, the throughputs increase in the following order: PS0, LP3/1, LP2/2, and LP2/1.

1) Discussion: The throughput improvements obtained in the lookahead pipelines are principally due to two factors: 1) protocol optimizations and 2) faster completion detectors. With each new protocol optimization, since there are fewer component delays, overall cycle time is reduced. The reductions in the number of critical components are summarized in the Analytical Formula column of Table I. The second factor is a circuit-level issue dealing with an individual component: the completion detector. Column t_{CD} indicates that in two of our designs—LP2/2 and LP2/1—the completion detector delay is significantly lower. The reason is that these two designs use an asymmetric C-element with a very short pull-up stack (see Fig. 7). In contrast, the completion detectors of PS0 and LP3/1 use a symmetric C-element which is slower.

It is also interesting to note that the evaluate and precharge latencies of LP stages (t_{Eval} and t_{Prech}) are essentially the same as in Williams' PS0. Hence, our throughput improvements are obtained without degrading latency.

The protocol optimizations that result in a significantly improved throughput for lookahead pipelines also introduce additional timing assumptions relative to those of the PS0 style. In particular, the following timing constraints of Section II-F must be satisfied: *precharge width* and *safe takeover* for LP3/1 and LP2/1 [(10) and (14)], and *input hold time* for LP2/2 and LP2/1 [(16)].

The simulations indicate that these timing constraints were met with adequate margins. There was at least a 0.21-ns safety margin for precharge pulse width (almost 100% margin), at least a 0.21-ns safety margin for safe takeover, and at least a 0.50-ns safety margin for input hold time.

2) Wider Datapaths: In many applications, a wide datapath can be partitioned into several narrower ones to keep the overhead of completion detection and control distribution low [20]. The overhead of partitioning is that forking and joining of data streams may be required in certain places; however, efficient fork and join structures have been proposed in [25], which only add two to four gate delays to the analytical cycle times of Table I. On the other hand, if the partitioning approach is not used, the throughput of all the dual-rail pipelines will decrease. This performance degradation is now quantified for the LP styles; a more general discussion appears in a companion paper [31].

Wider dual-rail datapaths require more complex completion detection, and greater buffering of the precharge control of each dynamic function block. In particular, for every doubling of the datapath width, completion detection requires two additional levels of logic. Therefore, for a 32-bit FIFO, the completion detection latency t_{CD} will increase by six gate delays for each of the designs. Similarly, for every eight-fold increase in the width of the datapath, two additional fanout-of-three inverters are added to amplify the precharge control of the logic block. These two extra inverters increase some, but not all, of the t_{Eval} and t_{Prech} latencies. In particular, the precharge and evaluation latencies from the control input to the data output increase; the latencies from the data inputs to the data output do not increase. Following the analysis of Section II, the critical cycle of the PS0 pipeline is lengthened by such control buffering in two places (one t_{Eval} and one t_{Prech} are affected). In contrast, the critical cycles for each of the LP pipelines are lengthened by only one buffer delay: only one t_{Eval} is affected, since there is no t_{Prech} in the cycle. Consequently, the analytical cycle time for each of the dual-rail pipelines will increase by the following amounts: 16 gate delays for PS0, 8 gate delays for LP3/1, 8 gate delays for LP2/2 (since the LP2/2 completion detector is asymmetric, only its up transition suffers the overhead), and 8 gate delays for LP2/1.

Intuitively, the overheads are lower in the LP designs as compared with PS0 because the analytical cycle times of the LP designs (see Table I) have either fewer precharge/evaluation latencies, or have fewer completion detection delays, or both. Thus, the LP styles have a significant advantage over PS0 when scaled to wider datapaths.

3) Area and Power/Energy Impact: In addition to throughput comparison, it is pertinent to also compare the area and power consumption of the dual-rail pipelines. In general, a significant fraction of the area and power consumption of dual-rail pipelines is consumed in completion detection. In particular, in our experiments, completion detectors accounted for 50%–70% of both total area and total power (or energy) consumption.

A brief comparison of the relative area and energy costs of the dual-rail pipelines is now provided. It is important to note that differences in power consumption could arise due to two factors: 1) differences in the amount of energy consumed (i.e.,

switched capacitance) per data item processed and 2) differences in throughput (i.e., number of data items processed per second). The following discussion factors out throughput differences and instead focuses on the total *energy* consumed by each pipeline per data item processed.

The improved throughput of the LP designs was obtained with little area or energy overheads. In fact, the LP2/2 and LP2/1 designs actually had *smaller* area than PS0 because their completion detectors were smaller. In particular, the LP2/2 and LP2/1 styles use an *asymmetric* completion detector (cf. Fig. 7) which was 40% smaller than the PS0 completion detector (cf. Fig. 2) because of a significantly smaller pull-up network. As a result, the LP2/2 and LP2/1 pipelines were overall 28% and 24% smaller, respectively, compared with the PS0 design. The LP3/1 pipeline was 4% larger in area than PS0 design: both use identical completion detectors, but LP3/1 needs an extra NAND gate in its control. The energy consumption results also showed a similar trend: the LP3/1 design exhibited 7% higher energy, but the LP2/2 and LP2/1 designs consumed 31% and 21% lower energy, respectively, when compared with the PS0 design.

C. Simulation Results for Single-Rail Pipelines

The operation of the 4-bit FIFO was simulated for both of the new single-rail bundled-datapath designs—$\mathrm{LP_{sr}}2/2$ and $\mathrm{LP_{sr}}2/1$.

Table II summarizes the results of our simulation. For each of the pipelines, the overall pipeline cycle time T is shown, as well as the delays of individual components: stage evaluation time (t_{Eval}), stage precharge time (t_{Prech}), the delay through the completion generator (t_{gC}), and in the case of $\mathrm{LP_{sr}}2/1$, the delay through the extra NAND gate (t_{NANDB}).

The two new designs $\mathrm{LP_{sr}}2/2$ and $\mathrm{LP_{sr}}2/1$, deliver quite high throughputs: 1.31 and 1.55 giga data items/s, respectively. As expected, the throughput of $\mathrm{LP_{sr}}2/1$, which combines both early evaluation and early done protocols, is better than the throughput of $\mathrm{LP_{sr}}2/2$.

As with the dual-rail designs, the simulations again indicate that the timing constraints of Section III-D [see (21), (26), and (27)] were adequately satisfied, with similar safety margins.

1) Wider Datapaths: In most applications, wide single-rail datapaths are handled with low performance overheads by using the approach of control kiting [22]. However, if the kiting approach is not used, the additional buffers required to amplify the precharge control of each stage will effectively increase the stage's evaluation and precharge latencies. As discussed in Section IV-B, for a 32-bit FIFO, t_{Eval} will increase by two gate delays. Therefore, for both $\mathrm{LP_{sr}}2/2$ and $\mathrm{LP_{sr}}2/1$, the overhead to cycle time is four gate delays.

2) Area and Power/Energy Impact: The single-rail pipelines were significantly more area- and energy-efficient than the dual-rail pipelines for two reasons: 1) the single-rail datapaths were only half as wide and 2) costly completion detectors were not required. In particular, the $\mathrm{LP_{sr}}2/2$ and $\mathrm{LP_{sr}}2/1$ designs were about 60% smaller and consumed 55% lower energy than their dual-rail counterparts.

V. Conclusion

This paper introduced several new asynchronous pipeline control structures for dynamic datapaths, called lookahead pipelines. These structures are especially suitable for very fine-grained, or gate-level, pipelines where each stage consists of a single gate.

The lookahead pipelines use the well-known PS0 pipeline style as a starting point, but achieve significant improvements by the use of novel protocol optimizations. In particular, we have introduced an *early done* scheme which moves completion detectors to the left of each function block, and an *early evaluation* scheme which taps off from the completion of two subsequent stages. Taken together, these schemes allow anticipation of critical events by a more refined sensing of the state of the pipeline.

Our best dual-rail lookahead pipeline offers more than twice the throughput (1.04 giga data items/s) of Williams' PS0, while maintaining the same forward latency. Our best single-rail lookahead pipeline achieves even higher throughput: 1.55 giga data items/s.

The single-rail designs are competitive with the leading recent approaches from Sun (GasP [35]) and IBM (IPCMOS [28]), while offering significant advantages in robustness and ease of design because of much simpler circuit structures and timing constraints.

In a companion paper [31], we introduce an alternative approach to high-speed dynamic pipelines, called high-capacity (HC) pipelines. We also discuss several practical issues that are relevant to both LP and HC pipelines: e.g., interfacing with disparate environments, and handling wide datapaths. Also, a detailed comparison of our approaches with other recent approaches is provided.

Acknowledgment

The authors would like to thank N. Abbasi of IBM T. J. Watson Research Center, NY, and Prof. K. Shepard of Columbia University for their help with simulations. They would also like to thank M. Theobald and T. Chelcea, both formerly of Columbia University, for their discussions, and Prof. M. Horowitz of Stanford University for helpful feedback.

References

[1] ARM, Cambridge, U.K., "ARM offers first clockless processor core," 2006. [Online]. Available: http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=179101800

[2] M. Benes, S. M. Nowick, and A. Wolfe, "A fast asynchronous Huffman decoder for compressed-code embedded processors," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 1998, pp. 43–56.

[3] B. Chappell, "The fine art of IC design," *IEEE Spectrum*, vol. 36, no. 7, pp. 30–34, Jul. 1999.

[4] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Taubin, and A. Yakovlev, "Lazy transition systems: Application to timing optimization of asynchronous circuits," in *Proc. ICCAD*, 1998, pp. 324–331.

[5] U. Cummings, "Terabit crossbar switch core for multi-clock-domain SoCs," in *Proc. Symp. Rec. Hot Chips*, 2003, pp. 102–112.

[6] P. Day and J. V. Woods, "Investigation into micropipeline latch design styles," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 3, no. 2, pp. 264–272, Jun. 1995.

[7] M. Ferretti and P. A. Beerel, "Single-track asynchronous pipeline templates using 1-of-N encoding," in *Proc. Des., Autom. Test Eur. (DATE)*, 2002, pp. 1008–1015.

[8] S. B. Furber and J. Liu, "Dynamic logic in four-phase micropipelines," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 1996, pp. 11–16.

[9] H. v. Gageldonk, D. Baumann, K. van Berkel, D. Gloor, A. Peeters, and G. Stegmann, "An asynchronous low-power 80c51 microcontroller," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 1998, pp. 96–107.

[10] J. D. Garside, W. J. Bainbridge, A. Bardsley, D. A. Edwards, S. B. Furber, J. Liu, D. W. Lloyd, S. Mohammadi, J. S. Pepper, O. Petlin, S. Temple, and J. V. Woods, "AMULET3i — An asynchronous system-on-chip," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 2000, pp. 162–175.
[11] D. Harris and M. Horowitz, "Skew-tolerant domino circuits," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, pp. 1702–1711, Nov. 1997.
[12] O. Hauck, A. Katoch, and S. A. Huss, "VLSI system design using asynchronous wave pipelines: A 0.35 μm CMOS 1.5 GHz elliptic curve public key cryptosystem chip," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 2000, pp. 188–197.
[13] International Technology Roadmap for Semiconductors, "Overall roadmap technology characteristics," 2005. [Online]. Available: http://www.itrs.net/Common/2005ITRS/Home2005.htm
[14] H. M. Jacobson, P. N. Kudva, P. Bose, P. W. Cook, S. E. Schuster, E. G. Mercer, and C. J. Myers, "Synchronous interlocked pipelines," in *Proc. Int. Symp. Asynch. Circuits Syst.*, 2002, pp. 3–12.
[15] M. B. Josephs, S. M. Nowick, and C. H. K. van Berkel, "Modeling and design of asynchronous circuits," *Proc. IEEE*, vol. 87, no. 2, pp. 234–242, Feb. 1999.
[16] R. Kol and R. Ginosar, "A doubly-latched asynchronous pipeline," in *Proc. Int. Conf. Comput. Des. (ICCD)*, 1996, pp. 706–711.
[17] Y. W. Li, G. Patounakis, A. Jose, K. L. Shepard, and S. M. Nowick, "Asynchronous datapath with software-controlled on-chip adaptive voltage scaling for multirate signal processing applications," in *Proc. Int. Symp. Asynch. Circuits Syst.*, 2003, pp. 216–225.
[18] A. M. Lines, "Pipelined asynchronous circuits," M.S. thesis, Dept. Comput. Sci., California Inst. Technol., Pasadena, 1998.
[19] W. Liu, C. T. Gray, D. Fan, W. J. Farlow, T. A. Hughes, and R. K. Cavin, "A 250-MHz wave pipelined adder in 2-μm CMOS," *IEEE J. Solid-State Circuits*, vol. 29, no. 9, pp. 1117–1128, Sep. 1994.
[20] A. J. Martin, A. Lines, R. Manohar, M. Nyström, P. Pénzes, R. Southworth, and U. Cummings, "The design of an asynchronous MIPS R3000 microprocessor," in *Proc. Adv. Res. VLSI*, 1997, pp. 164–181.
[21] A. J. Martin, M. Nyström, and C. G. Wong, "Three generations of asynchronous microprocessors," *IEEE Des. Test Comput.*, vol. 20, no. 6, pp. 9–17, Nov./Dec. 2003.
[22] C. E. Molnar, I. W. Jones, W. S. Coates, J. K. Lexau, S. M. Fairbanks, and I. E. Sutherland, "Two FIFO ring performance experiments," *Proc. IEEE*, vol. 87, no. 2, pp. 297–307, Feb. 1999.
[23] V. Narayanan, B. Chappell, and B. Fleischer, "Static timing analysis for self resetting circuits," in *Proc. ICCAD*, 1996, pp. 119–126.
[24] R. O. Ozdag and P. A. Beerel, "High-speed QDI asynchronous pipelines," in *Proc. Int. Symp. Asynch Circuits Syst.*, 2002, pp. 13–22.
[25] R. O. Ozdag, M. Singh, P. A. Beerel, and S. M. Nowick, "High-speed non-linear asynchronous pipelines," in *Proc. Des., Autom. Test Eur. (DATE)*, 2002, pp. 1000–1007.
[26] A. M. G. Peeters, "Single-rail handshake circuits," Ph.D. dissertation, Dept. Math. Comput. Sci., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 1996.
[27] S. Rotem, K. Stevens, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken, and B. Agapiev, "RAPPID: An asynchronous instruction length decoder," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 1999, pp. 60–70.
[28] S. Schuster, W. Reohr, P. Cook, D. Heidel, M. Immediato, and K. Jenkins, "Asynchronous interlocked pipelined CMOS circuits operating at 3.3–4.5 GHz," in *Proc. ISSCC*, 2000, pp. 292–293.
[29] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*, C. A. Mead and L. A. Conway, Eds. Reading, MA: Addison-Wesley, 1980, ch. 7.
[30] M. Singh, "The design of high-throughput asynchronous pipelines," Ph.D. dissertation, Dept. Comput. Sci., Columbia Univ., New York, 2001.
[31] M. Singh and S. M. Nowick, "The design of high-throughput asynchronous dynamic pipelines: High-capacity pipelines," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 11, pp. XXX–XXX, Nov. 2007.
[32] M. Singh and S. M. Nowick, "High-throughput asynchronous pipelines for fine-grain dynamic datapaths," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 2000, pp. 198–209.
[33] M. Singh, J. A. Tierno, A. Rylyakov, S. Rylov, and S. M. Nowick, "An adaptively-pipelined mixed synchronous-asynchronous digital FIR filter chip operating at 1.3 gigahertz," in *Proc. Int. Symp. Asynch. Circuits Syst.*, 2002, pp. 84–95.
[34] K. Stevens, R. Ginosar, and S. Rotem, "Relative timing," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 1999, pp. 208–218.
[35] I. Sutherland and S. Fairbanks, "GasP: A minimal FIFO control," in *Proc. Int. Symp. Asynch. Circuits Syst.*, 2001, pp. 46–53.
[36] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Mateo, CA: Morgan Kaufmann, 1999.
[37] I. E. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, no. 6, pp. 720–738, Jun. 1989.
[38] Sun Microsystems, Mountain View, CA, "The UltraSPARC IIIi processor architecture overview. Technical whitepaper," 2004 [Online]. Available: http://www.sun.com/processors/whitepapers/US3i_External.pdf
[39] T. E. Williams, "Self-timed rings and their application to division," Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., Stanford Univ., Stanford, CA, 1991.
[40] T. E. Williams and M. A. Horowitz, "A zero-overhead self-timed 160 ns 54 b CMOS divider," *IEEE J. Solid-State Circuits*, vol. 26, no. 11, pp. 1651–1661, Nov. 1991.
[41] D. Wong, G. De Micheli, and M. Flynn, "Designing high-performance digital circuits using wave-pipelining," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 12, no. 1, pp. 24–46, Jan. 1993.
[42] G. Yee and C. Sechen, "Clock-delayed domino for adder and combinational logic design," in *Proc. ICCD*, 1996, pp. 332–337.
[43] K. Y. Yun, P. A. Beerel, and J. Arceo, "High-performance asynchronous pipeline circuits," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, 1996, pp. 17–28.

Montek Singh received the B.Tech. degree in electrical engineering from IIT Delhi, Delhi, India, in 1993, and the Ph.D. degree in computer science from Columbia University, New York, NY, in 2002.

He is an Associate Professor with the Department of Computer Science, the University of North Carolina, Chapel Hill. His research interests include the area of asynchronous circuits and systems, especially circuits and synthesis tools for the design of high-speed pipelined systems. In 2005, he was brought onto the DARPA CLASS Program (led by Boeing), to develop, in collaboration with Handshake Solutions, an industrial-strength automated synthesis flow for designing high-speed pipelined asynchronous systems. His work has been transferred to industries, including IBM, Boeing, and Handshake Solutions (a Philips subsidiary).

Dr. Singh was a recipient of a Best Paper Award at the 2000 IEEE Async Symposium, a Best Paper Finalist Nomination at the 2002 Async Symposium, and an IBM Faculty Award in 2004. He was a Program Committee Co-Chair for the Async'07 Symposium and the FMGALS'05 Workshop.

Steven M. Nowick received the B.A. degree from Yale University, New Haven, CT, in 1976, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1993.

He is an Associate Professor with the Department of Computer Science and Electrical Engineering, Columbia University, New York, NY. His main research interest is on CAD tools, as well as design methods, for the synthesis, analysis, and optimization of asynchronous and mixed-timing digital systems. In 2005, was brought onto DARPA's CLASS project, headed by Boeing, with participation of a Philips-based startup (Handshake Solutions), to create a commercially-viable CAD tool flow for designing asynchronous systems. He was also a co-founder of the IEEE Async Symposia series.

Dr. Nowick was a recipient of a National Science Foundation (NSF) Faculty Early Career (CAREER) Award (1995), an Alfred P. Sloan Research Fellowship (1995), an NSF Research Initiation Award (RIA) (1993), two Best Paper Awards, one at the 1991 International Conference on Computer Design and the other at the 2000 IEEE Async Symposium, and two medium-scale NSF ITR Awards for asynchronous research in 2000. He was Program Committee Co-Chair of Async'94 and Async'99 and General Co-Chair of Async'05, and was Program Chair of the IWLS'02 Workshop. He is currently an Associate Editor of the IEEE Transactions on Computer-Aided Design and of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.