Asynchronous Domino Logic Pipeline Design Based on Constructed Critical Data Path

Zhengfan Xia, Masanori Hariyama, and Michitaka Kameyama

***Abstract*— This paper presents a high-throughput and ultralow-power asynchronous domino logic pipeline design method, targeting to latch-free and extremely fine-grain or gate-level design. The data paths are composed of a mixture of dual-rail and single-rail domino gates. Dual-rail domino gates are limited to construct a stable critical data path. Based on this critical data path, the handshake circuits are greatly simplified, which offers the pipeline high throughput as well as low power consumption. Moreover, the stable critical data path enables the adoption of single-rail domino gates in the noncritical data paths. This further saves a lot of power by reducing the overhead of logic circuits. An 8 × 8 array style multiplier is used for evaluating the proposed pipeline method. Compared with a bundled-data asynchronous domino logic pipeline, the proposed pipeline, respectively, saves up to 60.2% and 24.5% of energy in the best case and the worst case when processing different data patterns.**

***Index Terms*— Asynchronous pipeline, critical data path, dual-rail domino gate, single-rail domino gate.**

I. Introduction

DURING the last decade, there has been a revival in research on asynchronous technology. Along with the continued CMOS technology scaling, VLSI systems become more and more complex. The physical design issues, such as global clock tree synthesis and top-level timing optimization, become serious problems. Even if technology scaling offers more integration possibilities, modularity and scalability are difficult to be realized at the physical level. Asynchronous design is considered as a promising solution for dealing with these issues that relate to the global clock, because it uses local handshake instead of externally supplied global clock [1]–[4]. The attractive properties are listed as follows:

1) low power consumption;
2) high operating speed;
3) no clock distribution and clock skew problems;
4) better composability and modularity;
5) less emission of electromagnetic noise;
6) robustness toward variations in supply voltage, temperature, and fabrication process parameters.

In asynchronous design, the choice of handshake protocols affects the circuit implementation (area, speed, power, robustness, etc.). The four-phase bundled-data protocol and the four-phase dual-rail protocol are two popular protocols that are used in most practical asynchronous circuits. The four-phase bundled-data protocol design most closely resembles the design of synchronous circuits. Handshake circuits generate local clock pulses and use delay matching to indicate valid signal. It normally leads to the most efficient circuits due to the extensive use of timing assumptions. On the other hand, the four-phase dual-rail protocol design is implemented in an elaborate way that the handshake signal is combined with the dual-rail encoding of data. Handshake circuits are aware of the arrival of valid data by detecting the encoded handshake signal, which allows correct operation in the presence of arbitrary data path delays. This feature is very useful for dealing with data path delay variations in advanced VLSI systems, such as asynchronous field-programmable gate arrays (FPGAs) [5]–[7] and system-on-chip [3], [4]. However, such attractive feature is realized at the expense of encoding and detection overheads. These overheads cause low circuit efficiency and restrict the application area of the four-phase dual-rail protocol design.

This paper presents a novel design method of asynchronous domino logic pipeline, which focuses on improving the circuit efficiency and making asynchronous domino logic pipeline design more practical for a wide range of applications. The novel design method combines the benefits of the four-phase dual-rail protocol and the four-phase bundled-data protocol, which achieves an area-efficient and ultralow-power asynchronous domino logic pipeline.

Asynchronous domino logic pipeline is an interesting pipeline style that can entirely avoid explicit storage elements between stages by exploiting the implicit latching functionality of domino logic gates. The latchless feature provides the benefits of reduced critical delays, smaller silicon area, and lower power consumption.

However, asynchronous domino logic pipeline has a common problem that dual-rail domino logic has to be used to compose the domino data path. Single-rail domino logic cannot be used because it would break the domino data path since only noninverting logic can be implemented [13]. As a

Manuscript received September 5, 2013; revised February 17, 2014; accepted March 25, 2014. Date of publication April 18, 2014; date of current version March 18, 2015. This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology KAKENHI under Grant 12020735, in part by the Ministry of Economy, Trade and Industry, through the Research and Development Subsidiary Program for Promotion of Academia-industry Cooperation, in part by VLSI Design and Education Center, and in part by the University of Tokyo through STARC, e-Shuttle Inc., Fujitsu Ltd., Cadence Design Systems Inc., Synopsys Inc., and Mentor Graphics Inc. This paper was presented at the dual-rail/single-rail hybrid logic design concept appeared in the 2012 IEEE International Symposium on Circuits and Systems, May 2012, pp. 3017–3020, as dual-rail/single-rail hybrid logic design for high-performance asynchronous circuit and the construction of the critical data path using synchronizing logic gates appeared in the 2012 IEICE Transaction on Electronics, Vol. E95-C, no. 8, pp. 1434–1443, Aug. 2012, as design of high-performance asynchronous pipeline using synchronizing logic gates.

The authors are with the Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan (e-mail: xiazhengfan@ecei.tohoku.ac.jp; hariyama@ecei.tohoku.ac.jp; kameyama@ecei.tohoku.ac.jp).

Digital Object Identifier 10.1109/TVLSI.2014.2314685

result, the domino data path has a dual-rail encoding overhead that consumes a lot of silicon area and power consumption. Such overhead almost cancels out the area and power benefits provided by the latchless feature.

Another problem is the overhead of handshake control logic. Conventional designs of asynchronous domino logic pipeline based on the four-phase dual-rail protocol rely on domino data path to transfer data and encoded handshake signal, and use completion detectors to detect and collect the handshake signal throughout the entire data paths [8]–[11]. Such design method is very robust for delay variations in data paths. However, it causes a serious detection overhead. The detection overhead is growing with the width of data paths, which impedes its application in the design of a large function block with a considerable data path width. On the other hand, asynchronous domino logic pipeline based on the four-phase bundled-data protocol avoids the detection overhead by implementing a single extra bundling signal, to match the worst case block delay, which serves as a completion signal. The problem is that this design method completely loses the good properties in the four-phase dual-rail protocol design. Besides, it does not solve the dual-rail encoding overhead problem in data paths [12].

In this paper, our proposed pipeline reduces both the dual-rail encoding overhead in data paths and the detection overhead in handshake control logic by designing based on a constructed critical data path. A stable critical data path is constructed using redesigned dual-rail domino gates. By detecting the stable critical data path, a 1-bit completion detector is enough to get the correct handshake signal regardless of the data path width. Such design does not only greatly reduce the detection overhead but also partially maintains the good properties in the four-phase dual-rail protocol design. Moreover, the stable critical data path serves as a matching delay to solve the dual-rail encoding overhead problem in data paths. With the help of the redesigned dual-rail domino gates, single-rail domino logic is successfully applied in noncritical data paths. As a result, the proposed asynchronous domino logic pipeline has a small overhead in both handshake control logic and function block logic, which greatly improves the circuit efficiency. According to the design feature, we name the proposed pipeline as asynchronous pipeline based on constructed critical data path (APCDP).

This paper is organized as follows. Section II introduces the background of asynchronous domino logic pipeline. PS0 is introduced to demonstrate the advantages and problems of asynchronous domino logic pipeline based on dual-rail protocol. Several related designs are also simply introduced. Section III focuses on the introduction of the proposed pipeline design method. Synchronizing logic gates (SLGs) and synchronizing logic gates with a latch function (SLGLs) are introduced to construct a stable critical data path. The robustness of the pipeline structure and the constructed critical data path is analyzed. Then, more complex pipeline structures are further discussed. Section IV presents the evaluation results that show the benefits of the proposed pipeline compared with a bundled-data asynchronous domino logic pipeline and a synchronous pipeline with a sequential clock gating (Sync-CG). Section V presents the conclusion.

TABLE I
CODE TABLE OF THE FOUR-PHASE DUAL-RAIL ENCODING

	Codeword (w_t, w_f)
Data 0	(0, 1)
Data 1	(1, 0)
Spacer	(0, 0)
Not used	(1, 1)

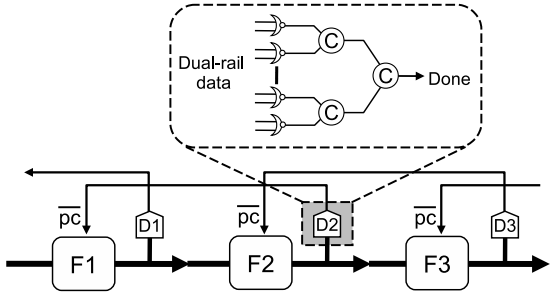


Fig. 1. Block diagram of PS0.

II. BACKGROUND

PS0 is a well-known implementation style of asynchronous domino logic pipeline based on dual-rail protocol [8]. It is an important foundation for most later proposed styles. Since our proposed pipeline is also based on PS0, we will begin by reviewing PS0 pipeline style, and then simply introducing two other advanced styles: 1) a timing-robust style called precharge half-buffer [9] and 2) a high-throughput style called lookahead pipeline [11]. Finally, we summary the delay assumptions of these pipelines and give our delay assumption in the proposed design.

A. PS0

1) Four-Phase Dual-Rail Protocol: PS0 is designed based on the four-phase dual-rail protocol. Fig. 3 shows an example of data transfer based on the four-phase dual-rail protocol, and Table I shows the code table of the four-phase dual-rail encoding. The four-phase dual-rail encoding encodes a request signal into the data signal using two wires, (w_t, w_f). The data value 0 is encoded as $(0, 1)$, and value 1 is encoded as $(1, 0)$; the spacer is encoded as $(0, 0)$; $(1, 1)$ is not used. When transferring the valid data, a spacer is inserted between them. A receiver can easily obtain the valid data by monitoring the two wires. This protocol is very robust since a sender and a receiver can communicate reliably regardless of delays in the combinational logic block and wires between them. The dual-rail encoded data path is known as the delay-insensitive data path.

2) Structure of PS0: Fig. 1 shows a block diagram of PS0. In PS0, each pipeline stage is composed of a function block and a completion detector. Each function block is implemented using dual-rail domino logic. Each completion detector generates a local handshake signal to control the flow of data through the pipeline. The handshake signal is transferred to

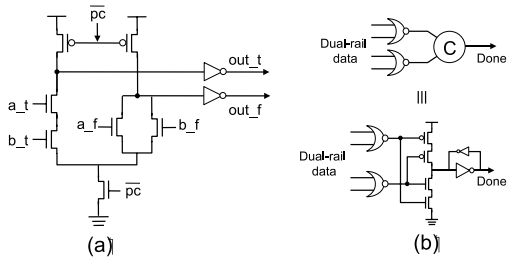


Fig. 2. (a) Dual-rail domino AND gate. (b) Two-bit completion detector.

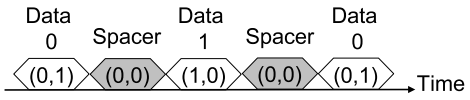


Fig. 3. Example of data transfer based on four-phase dual-rail protocol.

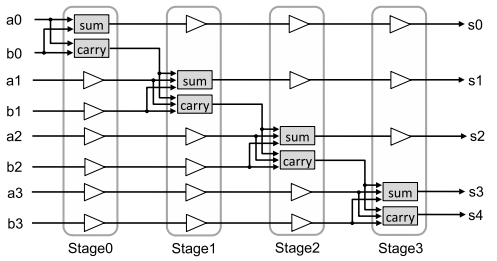


Fig. 4. Pipelined 4-bit ripple carry adder.

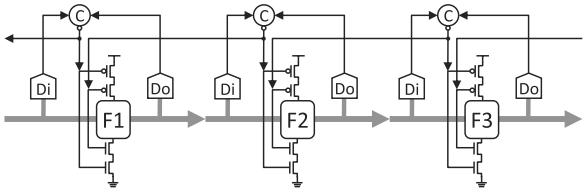


Fig. 5. Block diagram of PCHB.

the precharge/evaluation control port of the previous pipeline stage.

Fig. 2 shows an example of the dual-rail domino AND gate and the 2-bit completion detector. A two-input NOR gate serves as the 1-bit completion detector to generate a bit done signal by monitoring the outputs of dual-rail domino gate. To build a 2-bit completion detector, C-element is needed to combine the bit done signals. A full completion detector is formed by combining all bit done signals from the entire data paths with a tree of C-elements, as shown in Fig. 1.

3) Protocol of PS0: The protocol of PS0 is quite simple. F(N) is precharged when F($N + 1$) finishes evaluation. F(N) evaluates when F($N + 1$) finishes its reset, or precharge. In Fig. 1, if we observe a single data flow through an initially empty pipeline in which every pipeline stage is in evaluation phase, the complete cycle of events is as follows.

1) F1 evaluates and data flow to F2.
2) F2 evaluates and data flow to F3. F2's completion detector detects completion of evaluation and sends a precharge signal to F1.
3) F1 precharges and F3 evaluates. F3's completion detector detects completion of evaluation and sends a precharge signal to F2.
4) F2 precharges. F2's completion detector detects the completion of precharge and sends an evaluation signal (enable signal) to F1. The evaluation signal enables F1 to evaluate new data once again.

There are three evaluations, two completion detections, and one precharge in the complete cycle for a pipeline stage. The pipeline cycle time T_{cycle} is

$$T_{\text{cycle}} = 3t_{\text{Eval}} + 2t_{\text{CD}} + t_{\text{Prech}} \tag{1}$$

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.

4) Overhead Problems: There are mainly two overhead problems that prohibit the widespread use of PS0, the detection overhead in handshake control logic and the dual-rail encoding overhead in function block logic. A ripple carry adder, shown in Fig. 4, is used as an example to clarify these overhead problems.

The detection overhead is caused by the full completion detectors that are used to deal with data path delay variations by detecting the entire data paths. The overhead greatly affects the pipeline speed and power consumption. The most serious problem is that the detection overhead is growing the width of data paths. In a 4-bit ripple carry adder, the width of data paths is between 8 and 5 bits. The detection overheads of 8–5-bit completion detectors might be acceptable in practical design. However, in 32-bit ripple carry adder design, the width of data paths is at least 33 bits. The overhead of 33-bit completion detector is so large that PS0 is hardly applicable in such situation. Even the detection time can be reduced by partitioning wide data path into several data streams [11], the detection power is not reduced.

The dual-rail encoding overhead is caused by dual-rail domino logic that is used for not only implementing logic function but also storing data between pipeline stages. Because there are no explicit storage elements (latches or registers), a lot of dual-rail domino buffers have to be added to levelize each stage. The added dual-rail domino buffers consume a lot of silicon area and power. In a 4-bit ripple carry adder, 18 dual-rail domino buffer gates are added, which almost cancel out the benefit of removing explicit storage elements.

B. Other Advanced Pipelines

1) Precharge Half-Buffer Pipeline: Fig. 5 shows a block diagram of precharge half-buffer pipeline (PCHB). PCHB is a timing-robust pipeline style that uses quasi-delay-insensitive control circuits [9]. Two completion detectors in a PCHB stage: one on the input side (D_i) and one on the output side (D_o). The complete cycle of events for a PCHB stage is quite similar to that of PS0, except that a PCHB stage verifies its input bits. Because of the input completion detector (D_i), a PCHB stage does not start evaluation until all input bits are valid. This design absorbs skew across individual bits in the

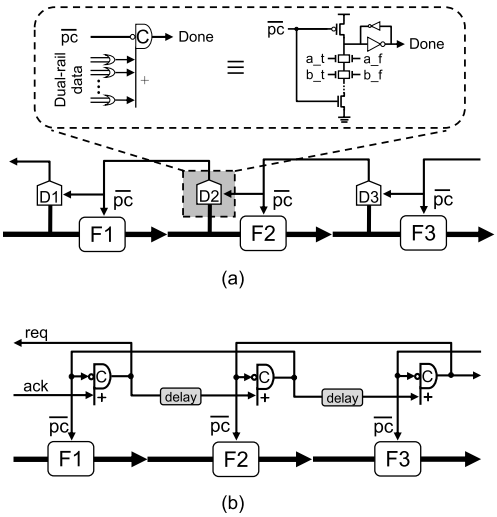


Fig. 6. Block diagrams of LP2/2. (a) LP2/2 based on dual-rail protocol. (b) LP2/2-SR.

data paths. Although this design makes PCHB more timing robust, it causes a two-time overhead in handshake control logic compared with PS0. Besides, PCHB has the same dual-rail encoding overhead as PS0.

2) LP2/2: LP2/2 is a high-throughput pipeline style [11], which has both dual-rail protocol design and bundled-data protocol design.

Fig. 6(a) shows the block diagram of LP2/2 based on the dual-rail protocol. LP2/2 improves the throughput of PS0 by optimizing the sequential of handshake events. However, they do not solve the overhead problems in handshake control logic and function block logic. The handshake speed is accelerated by employing asymmetric completion detectors placed ahead of function blocks. Although this pipeline structure reduces the handshake cycle time, the asymmetric completion detectors still consume a lot of power since they have to detect the entire data paths.

Fig. 6(b) shows the block diagram of LP2/2 based on the bundled-data protocol (LP2/2-SR). LP2/2-SR avoids the detection overhead problem by implementing a single extra bundling signal. The bundling signal serves as a completion signal, which matches the worst case delay in function blocks. Although such design reduces the power consumption in handshake control logic, the overhead problem in function block logic remains unsolved since dual-rail domino logic still has to be used to compose the domino data path [12].

C. Delay Assumptions

PCHB is a very robust pipeline that requires no delay assumptions or calculations by designer. However, the robustness of circuits comes at the expense of performance. The complex handshake circuits slow down the handshake speed and consume more power.

PS0 is designed for fast handshake speed by seeking to design control circuits that are always correct for common conditions. It is based on a delay assumption that each pipeline stage's predecessor precharges no slower than the stage's successor evaluates.

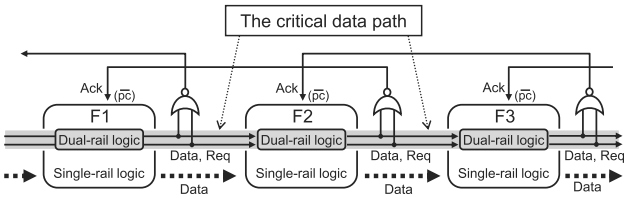


Fig. 7. Block diagram of APCDP.

Based on PS0, LP2/2 makes two more aggressive delay assumptions: first, each pipeline stage evaluates no slower than its completion detects plus the stage's successor precharges. Second, each pipeline stage completion detects plus its predecessor precharges no slower than the stage evaluates plus its successor completion detects. Although these delay assumptions improve the pipeline throughput, they sacrifice the pipeline robustness.

Our proposed pipeline is based on PS0, but makes a different delay assumption from LP2/2. We assume that, in the evaluation of domino gates, a n-stack pull-down path causes larger delay than a m-stack pull-down path (n is larger than m). Based on this delay assumption, we construct a stable critical data path for improving the circuit efficiency in both handshake control logic and function block logic. Because this delay assumption is made on gate level instead of handshake structure level, our proposed pipeline has, structurally, the same robustness as PS0.

III. Asynchronous Pipeline Based on Constructed Critical Data Path

A. Overview

Fig. 7 shows the block diagram of the proposed asynchronous pipeline (APCDP). The pipeline is designed based on a stable critical data path that is constructed using a special dual-rail logic. The critical data path transfers a data signal and an encoded handshake signal. Noncritical data paths, composed of single-rail logic, only transfer data signal. A static NOR gate detects the dual-rail critical data path and generates a total done signal for each pipeline stage. The outputs of NOR gates are connected to the precharge ports of their previous stages.

APCDP has the same protocol as PS0. The difference is that a total done signal is generated by detecting only the critical data path instead of the entire data paths. Such design method has two merits. First, the completion detector is simplified to a single NOR gate, and the detection overhead is not growing with the data path width. Second, the overhead of function block logic is reduced by applying single-rail logic in noncritical data paths. As a result, APDCP has a small overhead in both handshake control logic and function block logic, which greatly improves the throughput and power consumption.

APCDP is more familiar to bundled-data asynchronous pipeline because the critical data path essentially works as

TABLE II
STATES OF PULL-DOWN TRANSISTOR PATHS ON DIFFERENT DATA PATTERNS

Data patterns (a_t, a_f, b_t, b_f)	Pull-down transistor paths						
	Fig. 2(a) Conventional			Fig. 6 Synchronizing AND Gate			
	[a_t, b_t]	[a_f]	[b_f]	[a_t, b_t]	[a_t, b_f]	[a_f, b_t]	[a_f, b_f]
(0, 1, 0, 1)	OFF	ON	ON	OFF	OFF	OFF	ON
(0, 1, 1, 0)	OFF	ON	OFF	OFF	OFF	ON	OFF
(1, 0, 0, 1)	OFF	OFF	ON	OFF	ON	OFF	OFF
(1, 0, 1, 0)	ON	OFF	OFF	ON	OFF	OFF	OFF

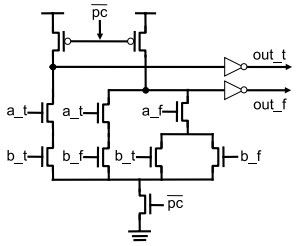
a matching delay, which controls the correct data transfer in noncritical data paths. Compared with conventional bundled-data design using a separate matching delay to match the worst case delay in function blocks, the proposed design method reuses the existing function block logic to provide the matching delay. Such design has many advantages. First, the matching delay is accurate. The matching delay in APCDP is exactly the same worst case delay in function blocks. Second, the matching delay is robust for delay variations. Dual-rail critical data path supplies delay-insensitive property, which can be self-adaptive to delay variations in function blocks. Third, the handshake control logic is efficient in area and power. The handshake control logic is implemented by reusing the existing function block logic.

Finding a stable critical data path in function blocks is very important in the proposed design. The problem is that it is difficult to get a stable critical data path using traditional logic gates. Traditional logic gates have the gate-delay data-dependence problem—the gate delay is dependent on input data patterns. For example, the ripple carry adder in Fig. 4. The ripple carry path seems to be the stable critical data path. However, actually, the critical data path varies according to different input data patterns. Because of the gate-delay data-dependence problem, the carry function gate can be triggered early by the input bits (a_n and b_n) regardless of the carry bit. Since the input bit travels faster in the buffer path than the carry bit in the ripple carry path, it cannot guarantee that the critical transition signal always presents on the ripple carry path.

Adding delay elements is an intuitive way to construct a stable critical data path. However, this method needs complex timing analysis and would cause huge overhead of delay elements. This paper introduces an efficient solution that uses SLGs to construct the critical data path. The SLGs solve the gate-delay data-dependence problem by making sure that SLGs cannot start evaluation until all valid data arrive [14]. This feature does not only help to construct a stable critical data path but also enable the adoption of single-rail domino logic in the noncritical data paths. As a result, the proposed design is significantly area and power efficient.

B. Logic Gates

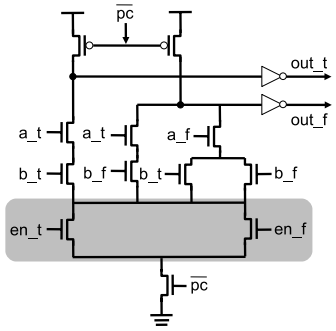
In VLSI circuits, it is difficult to get a stable critical data path using traditional logic gates due to the gate-delay data-dependence problem. Fig. 2(a) shows a traditional dual-rail domino AND gate. The true side of logic is implemented by out_t $=$ $a\_t \cdot b\_t$ and the false side by out_f $=$ $a\_f + b\_f$. Table II shows the states of pull-down transistor paths on different data patterns. In traditional dual-rail domino AND gate, there are three transistor paths: 1) $[a\_t, b\_t]$; 2) $[a\_f]$; and 3) $[b\_f]$. First of all, these paths have different number of transistors at the sequential position. When they turn ON, respectively, $[a\_f]$ and $[b\_f]$ cause less delays than $[a\_t, b\_t]$. Moreover, when the data pattern is $(0, 1, 0, 1)$, $[a\_f]$ and $[b\_f]$ will be both ON, which leads to a much quicker signal transfer. As a result, the gate delay has a large variation depending on different data patterns. To solve the gate-delay data-dependence problem, SLG and SLGL are introduced [14].



Truth table of dual-rail AND logic

pc	a_t	a_f	b_t	b_f	out_t	out_f
0	–	–	–	–	0	0
1	0	1	0	1	0	1
1	0	1	1	0	0	1
1	1	0	0	1	0	1
1	1	0	1	0	1	0

Fig. 8. Synchronizing AND gate and the truth table of dual-rail AND logic.



The states of latch

pc	en_t	en_f	States
0	–	–	Opaque
1	0	0	Opaque
1	0	1	Transparent
1	1	0	Transparent
1	1	1	No used

Fig. 9. Synchronizing AND gate with a latch function and the table of latch states.

1) Synchronizing Logic Gates: SLGs are dual-rail domino gates that have no gate-delay data-dependence problem. Fig. 8 shows the synchronizing AND gate and the truth table of dual-rail AND logic. The principle is that, in the pull-down network, there is exactly one path activated according to one data pattern, and the stack of all possible paths is kept constant at the sequential position. Compared with the traditional design, the false side logic expression is changed to out_f $= a\_t \cdot b\_f + a\_f \cdot (b\_t + b\_f)$. Table II shows that there are four transistor paths: 1) $[a\_t, b\_t]$; 2) $[a\_t, b\_f]$; 3) $[a\_f, b\_t]$; and $[a\_f, b\_f]$. Every path has two transistors at the sequential position, and there is only one path turns ON corresponding to an input data pattern. As a result, the gate delay becomes independent on different data patterns. This kind of gates is named as SLGs because they can synchronize their inputs. The SLGs verify that all data signal transitions have arrived on their inputs before changing their outputs.

The characteristics of SLGs are listed as follows.

1) An SLG has a certain number, inputs' number, of transistors in pull-down transistor paths at the sequential position.

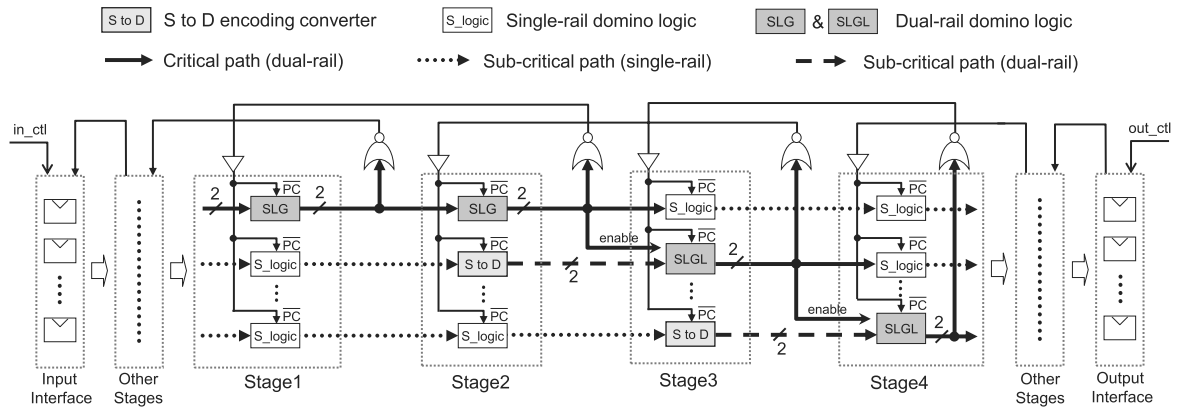


Fig. 10. Structure of APCDP.

2) An SLG has no gate-delay data-dependence problem. Its gate delay mainly relates to the inputs number.
3) An SLG can synchronize its inputs. The SLG cannot start evaluation until all valid data arrive.

2) Synchronizing Logic Gates With a Latch Function: Based on the characteristics of SLGs, SLGLs are extended. Fig. 9 shows synchronizing AND gate with a latch function and the table of latch states. An SLGL has an enable port (en_t, en_f), which controls the opaque and transparent state of the SLGL. The principle is that SLGLs cannot start evaluation without the presence of the enable signal.

Same as the dual-rail AND logic, all traditional dual-rail domino logic can be redesigned to become an SLG or an SLGL. The critical data path in dual-rail asynchronous pipeline can be easily constructed using SLGs and SLGLs.

C. Structure of APCDP

Fig. 10 shows the structure of APCDP. The solid arrow represents a constructed critical data path (dual-rail data path), the dotted arrow represents the noncritical data paths (single-rail data paths), and the dashed arrow represents the output of single-rail to dual-rail encoding converter.

In each pipeline stage, a static NOR gate is used as 1-bit completion detector to generate a total done signal for the entire data paths by detecting the constructed critical data path. Driving buffers deliver each total done signal to the precharge/evaluation control port of the previous stage. Since the completion detector only detects the constructed critical data path, the noncritical data paths do not have to transfer encoded handshake signal anymore. Therefore, single-rail domino gates are used in the noncritical data path to save logic overhead. Encoding converter is used to bridge the connection between single-rail domino gate and dual-rail domino gate.

1) Construction of the Critical Data Path: It is difficult to construct a stable critical data path using traditional logic gates for their gate-delay data-dependence problem. The critical signal transition varies from one data path to others according to different input data patterns. Since SLGs have solved the gate-delay data-dependence problem, a stable critical data path can be easily constructed by the following steps:

1) finding a gate (named as L_{in} gate) that has the largest number of inputs in each pipeline stage;
2) changing these L_{in} gates to SLGs;
3) linking SLGs together to form a stable critical data path.

The basic idea of finding the critical signal transition is that embedding an SLG in each pipeline stage and making the SLG to be the last gate to start and finish evaluation. First of all, the embedded SLG has the largest gate delay in a pipeline stage. The reasons are as follows.

1) The SLG has the largest stack in the pull-down network compared with other gates.
2) The SLG has only one pull-down transistor path activated for each input data pattern.

Then, if all gates evaluate at the same time or the SLG is the last gate to start evaluation in the pipeline stage, the critical signal transition would present on the output of the SLG.

In practice, making all gates evaluate at the same time is difficult, especially without the help of intermediate latches or registers. Therefore, we make the SLG become the last gate to start evaluation by linking each pipeline stage's SLG together. In the first pipeline stage, the critical signal transition is on the output of the SLG because all gates evaluate at the same time for the input control of latches or registers. After linking each pipeline stage's SLG together, the SLG in the following pipeline stage would be the last gate to start evaluation since it always waits for the critical signal transition from the previous SLG. As a result, the linked SLG data path becomes a stable critical data path.

Linking each pipeline stage's SLG together is partially done in the process of selecting L_{in} gate in each pipeline stage. When searching L_{in} gate, there might be more than one option. It is best to select the L_{in} gate that is originally linked to the L_{in} gate in the following pipeline stage. After changing these L_{in} gates to SLGs, SLGs are naturally linked. For example, the linkage between Stage1 and Stage2 in Fig. 10. However, if we cannot find the linked L_{in} gates in neighbor stages, SLGL needs to be used to solve the linking problem. The linkage between Stage2 and Stage3 is in such situation. The linkage is established by connecting the output of SLG in Stage2 and the enable port of SLGL in Stage3.

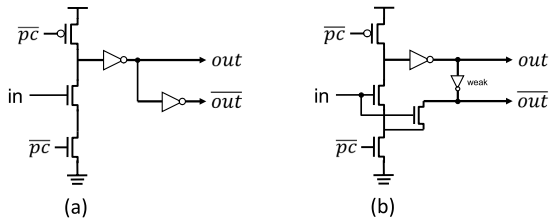


Fig. 11. Encoding converters. (a) Intuitive design. (b) Proposed design.

TABLE III
TRUTH TABLE OF ENCODING CONVERTER

$\overline{pc}$	in	out	$\overline{out}$
0	–	0	1
1	0	0	1
1	1	1	0

2) Encoding Conversion: Since the completion detector detects only the constructed critical data path, the noncritical data paths do not have to transfer encoded handshake signal anymore. The logic overhead in the noncritical data paths can be reduced using single-rail domino gates instead of dual-rail domino gates. However, single-rail domino gate and dual-rail domino gate use different encoding schemes. It has encoding compatibility problem when a single-rail domino gate connects to a dual-rail domino gate. Encoding converter needs to be designed to solve the problem.

Fig. 11 shows two implementations of encoding converter. Table III shows the truth table. In precharge phase $\overline{pc} = 0$, encoding converter outputs a dual-rail data0 (out, $\overline{out}$) = (0, 1). In evaluation phase $\overline{pc} = 1$, if the input is a single-rail data0 in = 0, the converter keeps the dual-rail data0. If the input is a single-rail data1 in = 1, the converter outputs a dual-rail data1 (out, $\overline{out}$) = (1, 0). Since single-rail encoding only has two states that, respectively, represent data0 and data1, there is no other state that can be converted to spacer (out, $\overline{out}$) = (0, 0). The disappearance of spacer violates the four-phase dual-rail protocol, which would cause data transfer error.

Fig. 10 shows two examples that encoding converters are used to bridge the connection between single-rail domino gate and dual-rail domino gate. Focusing on the encoding converter in Stage2, when Stage2 enters the precharge phase, the SLG outputs a spacer, but the converter outputs a invalid data0. This invalid data0 cannot be absorbed by the SLGL in Stage3 since the spacer impedes its evaluation. However, when Stage2 enters the evaluation phase, it has a risk that the invalid data0 might be erroneously absorbed if the output of the SLG becomes valid earlier than the output of the converter. The earlier arrived valid data from the SLG trigger the SLGL to start evaluation and absorb the invalid data0. To avoid this problem, the encoding converter needs to satisfy a timing constraint that the output of the converter should become valid earlier than the output of SLG. In other words, the constructed critical data path should be robust.

In addition to protect data transfer error by enhancing the robustness of the critical data path, we can also improve the conversion speed of the encoding converter. Interestingly, we do not have to care about the conversion from the single-rail data0 in = 0 to the dual-rail data0 (out, $\overline{out}$) = (0, 1). Because the converter initially outputs a dual-rail data0, this conversion can be considered infinite fast. We only have to focus on improving the conversion from the single-rail data1 in = 1 to the dual-rail data1 (out, $\overline{out}$) = (1, 0). Fig. 11(b) shows the proposed design of the encoding converter. When the converter enters the evaluation phase, the input in = 1 can immediately pull down $\overline{out}$. No matter the output (out, $\overline{out}$) is a instant spacer (0, 0) or the valid data1 (1, 0), it effectively protects the data transfer error. On the other hand, the intuitive design in Fig. 11(a) has a longer signal transition delay. $\overline{out}$ cannot be pulled down until out becomes 1. It has a higher possibility of causing data transfer error than the proposed encoding converter.

D. Robustness Analysis

APCDP has pipeline failure in the situation that a pipeline stage does not finish evaluating before its previous stage start precharge. In such situation, the pipeline stage cannot correctly finish evaluating because the precharge of its previous pipeline stage removes the valid data from the inputs. To avoid this pipeline failure, APCDP needs to satisfy an assumption that, in a pipeline stage, none of the other bits across the entire data paths is slower than the detected bit by more than the delay through a static NOR gate and the drive buffer chain following it. The robustness of APCDP is analyzed based on this assumption.

1) Robustness of the Pipeline Structure: According to the pipeline structure of APCDP, the hold time T_{hold} of valid data on the inputs of each pipeline stage is

$$T_{\text{hold}} = t_{\text{SLG\_Eval}} + t_{\text{NOR}} + t_{\text{Buf}} + t_{\text{SLG\_Prech}} \tag{2}$$

where $t_{\text{SLG\_Eval}}$ is the evaluation time for the SLG in a pipeline stage and $t_{\text{SLG\_Prech}}$ is the precharge time for the SLG in the previous pipeline stage. $t_{\text{NOR}} + t_{\text{Buf}}$ is the delay through the NOR gate and the drive buffer.

The pipeline structure of APCDP is quite robust since the hold time T_{hold} supplies sufficient time margins. In the construction of the critical data path, we introduced that the SLG is embedded as the last gate to finish evaluation in each pipeline stage. There are even some gates that are slower than the SLG because of delay variations in practice, T_{hold} supplies $t_{\text{NOR}} + t_{\text{Buf}} + t_{\text{SLG\_Prech}}$ time margins for pipeline failure protection. We believe that these time margins are sufficient for dealing with delay variations in practical design. However, for safety, we supply several enhance measurements for the constructed critical data path in the following section.

2) Robustness of the Critical Data Path: We first use the method of logical effort [19] to analyze the robustness of the constructed critical data path. Then, we discuss how to further enhance the robustness of the constructed critical path.

The method of logical effort is an easy way to estimate delay in CMOS circuit. In the method, modeling delay of a logic gate isolates the effects of a particular fabrication process by expressing all delays in terms of a basic delay unit particular to

that process. The delay incurred by a logic gate is comprised of two components: 1) a fixed part called the parasitic delay p and 2) a part that is proportional to the load on the gate's output, called the effort delay f. This effort delay depends on the load and the properties of the logic gate driving the load. There are two related terms for these effects: the logical effort g captures the effect of the logic gate's topology on its ability to produce output current, while the electrical effort h describes how the electrical environment of the logic gate affects performance and how the size of the transistors in the gate determines its load-driving capability. Electronic effort is also called fanout by many CMOS designer. As a result, the delay of a logic gate is expressed as

$$\text{delay} = f + p = gh + p = g\frac{C_{\text{out}}}{C_{\text{in}}} + p \tag{3}$$

where C_{out} is the capacitance that loads the output of the logic gate and C_{in} is the capacitance presented by the input terminal of the logic gate.

In each pipeline stage of APCDP, the SLG/SLGL has a larger gate delay than other gates according to the method of logic effort. First, the SLG/SLGL has more complicated topology than other gates in the pull-down network. It slightly increases the parasitic delay p and the logical effect g. Second, the output of SLG/SLGL is connected to a static NOR gate and the SLG/SLGL in the next stage. Compared with the outputs of other gates, the SLG/SLGL has a larger fanout C_{out}, which increases the electrical effort h. As a result, the SLG/SLGL has a larger gate delay than traditional logic gates even they have same number of inputs. When linking all SLGs/SLGLs together, these imposed delays increase the robustness of the constructed critical data path.

In practice, the robustness of the constructed critical path is affected by delay variations. As a matter of fact, it is a common problem in VLSI circuit design, same as the robustness of a clock signal in synchronous design and a match delay line in bundled-data asynchronous design [2]. As we all know, these designs all suffer from delay variations. To resist the influence of delay variations, synchronous design enlarges the cycle time of a clock signal to get some margin. On the other hand, bundled-data asynchronous design adds extra delay margin on the matching delay line to match the worst case delay in combinational logic block. Same like these solutions, the delay variations problem in the proposed design can be solved by enlarging delay margin on the constructed critical data path. We supply four measures to enlarge the delay margin, which are listed as follows:

1) sizing the pull-down transistors or the static inverters of SLGs and SLGLs to increase gate delays;
2) applying a low priority in circuit layout for the constructed critical path;
3) improving the noncritical paths delay;
4) adding delay elements on the critical path.

These measures have different impacts on the performance of circuits. It is better to choose a proper measure or multiple measures according to the practical design requirements. In measure 1), reducing the transistor size of pull-down network or the static inverters can increase the gate delays of SLGs and SLGLs. The increased delay slightly slows down pipeline speed, but smaller transistor size helps in saving power and silicon area. Measure 2) is layout optimization. Although this measure slightly decreases pipeline speed, it does not impose the extra overhead of transistors. Measure 3) does not degrade pipeline speed. The problem is that improving the noncritical path delay would increase the power consumption. Measure 4) is an intuitive way to enhance the critical data path. The drawback is the extra overhead of transistors.

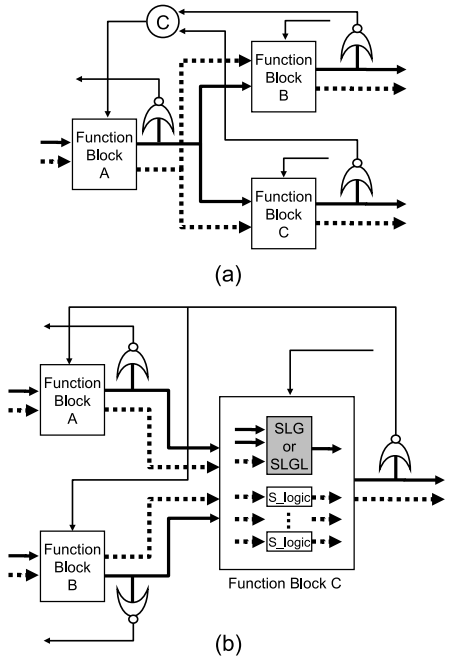


Fig. 12. (a) Fork structure. (b) Join structure.

In addition, the use of domino logic introduces many design risks because it is very sensitive to noise, circuit, and layout topologies [20]. The solutions to alleviate these problems are not in the scope of this paper. We only recommend to limit the largest stack of domino logic when designing APCDP. The limitation depends on processing technology, practical problems, and actual conditions. Smaller stack design helps in alleviating the noise problems and increasing the pipeline speed. The tradeoff is the increased number of pipeline stages. More pipeline stages means larger silicon area and higher power consumption. On the other hand, larger stack design deteriorates the noise problems and the pipeline speed. The merit is the reduced number of pipeline stages. Less pipeline stages helps in saving silicon area and power consumption.

E. Extension to Complex Structures

The previous sections just analyzed the linear pipeline structure. For more complex data paths, forks and joins are needed [2]. Fig. 12 shows fork structure and join structure in APCDP.

In fork structure, the outputs of function block A are split to connect with function blocks B and C. C-element is used to collect the handshake signal from A's successors. The

construction of the critical data path in fork structure is same as that of described in the linear structure. The problem is that the data paths from A to B and C are more complex than the linear structure. The complex data paths cause large delay variations, which affect the correctness of the critical data paths at the inputs of B and C. Pipeline failure happens when B and C do not finish their evaluations before A finishes its precharge. The delivery time of the precharge signal from B and C to A is that

$$T_{\text{del}} = T_{\text{NOR}} + T_C + T_{\text{buffer}} \tag{4}$$

where T_{NOR}, T_C, and T_{buffer} are delay time of a static NOR gate, a C-element, and the buffer gate. If the delay variations on the data paths are smaller than T_{del}, no pipeline failure happens.

In join structure, the outputs of function block A and B merge together at function block C, which requires sending an acknowledge signal from C to all its predecessors. In function block C, the critical data paths from function block A and B need to simultaneously connect to an SLG/SLGL. The design process is similar to that of described in the linear structure. The problem in join structure is that the acknowledge signal networks at function block B and C are more complex than the linear structure. Pipeline failure happens when A and B do not completely finish their precharge process before C enters the next evaluation phase, which means that C would mistakenly absorb old data from A or B. The delivery time of the precharge signal from C to A and B is that

$$T_{\text{del}} = T_{\text{NOR}} + T_{\text{buffer}}. \tag{5}$$

According to the handshake protocol, the time for C to enter the next evaluation phase is that

$$T_{\text{nextEval}} = 2T_{\text{Eval}} + 2T_{\text{NOR}} + 2T_{\text{buffer}} \tag{6}$$

where T_{Eval} is the evaluation time for a function block. Therefore, the margin time is that

$$T_{\text{margin}} = T_{\text{nextEval}} - T_{\text{del}} = 2T_{\text{Eval}} + T_{\text{NOR}} + T_{\text{buffer}}. \tag{7}$$

If the delay variations on the acknowledge signal networks are smaller than T_{margin}, no pipeline failure happens.

IV. Evaluation

This section presents the evaluation results of APCDP. An 8 × 8 array style multiplier is chosen as the test case, which is, respectively, designed using the proposed APCDP, LP2/2-SR [11], classic synchronous pipeline (Sync), and Sync-CG [15], [16]. Conventional dual-rail asynchronous pipelines are not selected as the evaluation counterparts because they are hardly applicable in the design of large function block (such as the 8 × 8 array style multiplier).

A. Experiment Setup

Four 8 × 8 array style multipliers are designed using HSPICE in a 65-nm design technology. All designs are simulated at 1.2 V normal supply voltage, 85 °C temperature, and a normal process corner.

TABLE IV
Evaluation Results of 8×8 Array Style Multipliers

	APCDP	LP22-SR (Bundled-data design)	Sync	Sync-CG
Function	8 × 8 multiplier			
Logic gate	Domino gate		Static gate	
Transistor counts	7375	9874	9010	9154
Total FET width	3075.45um	4726.35um	5065.07um	5171.48um
SLG & SLGL	56	NA	NA	NA
Storage element	0	0	278 (D flip-flop)	284 (D flip-flop)
Latency	0.44ns	0.37ns	1.3ns	1.3ns
Throughput (schematic)	4G data-set/s	5.2G data-set/s	4G data-set/s	4G data-set/s

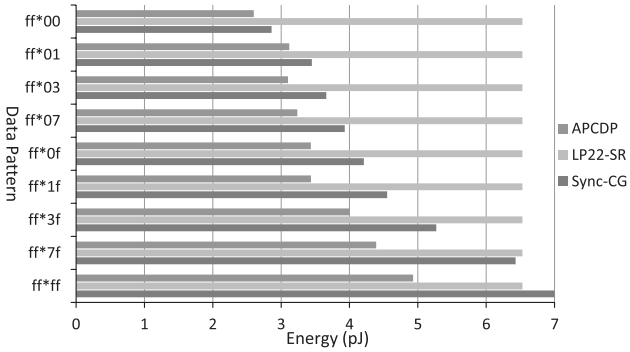


Fig. 13. Energy consumption per cycle for processing different data patterns.

LP2/2-SR is chosen as a representative of latchless pipeline for comparing with APCDP. LP2/2-SR was proposed along with LP2/2, which is significantly more area and energy efficient. It was reported that LP2/2-SR had about 60% smaller area and 55% lower energy consumption than LP2/2 in first-input first-output (FIFO) design [11]. Besides, LP2/2-SR resembles the design of latchless synchronous pipeline. The difference is that latchless synchronous pipeline uses a complex multiphase clocking instead of bundled-data handshaking. Because of their similarity, the performance of LP2/2-SR can be used as a reference for comparing APCDP with latchless synchronous pipeline.

An 8 × 8 array style multiplier is extremely fine-grain or gate-level pipelined using LP2/2-SR as well as APCDP. The depth of each pipeline stage is only one domino logic gate, and there are no explicit storage elements between stages. Their comparisons show not only the circuit efficiency of APCDP but also the merits of dual-rail asynchronous design.

To make a further comparison, Sync and Sync-CG are also designed. They are used as comparison references since they do not belong to latchless pipeline. The 8 × 8 array style multiplier is divided into five pipeline stages. The logic blocks are composed of static logic gates, and D flip-flops are used as intermediate storage elements between pipeline stages. The sequential clock gating in Sync-CG is designed using six D flip-flops, which realizes fine-grain clock gating that is similar to the handshake behaviors in APCDP and LP2/2-SR.

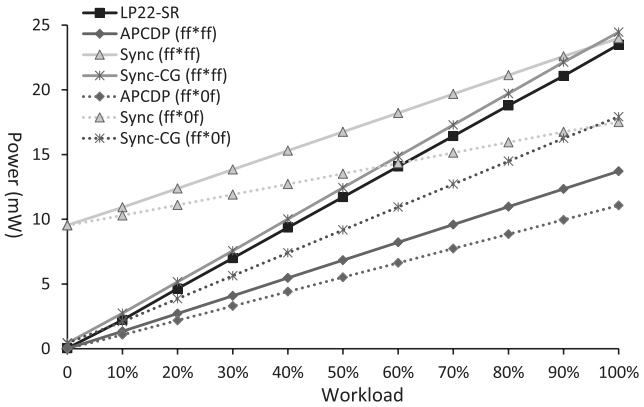


Fig. 14. Performances of power consumption (all designs operate at 3.6G dataset/s).

B. Results

Table IV shows the evaluation results of the 8×8 array style multiplier. The performances of throughput are evaluated without considering design margins, which are ideal results. The results show that APCDP has high throughput, the smallest transistor count, and the lowest forward latency in all designs. Fig. 13 shows the energy consumption for processing different data patterns. Fig. 14 shows the performances of power consumption under different workloads. The results show that APCDP is the most power efficient design.

1) Transistor Count: Table IV shows that APCDP, respectively, reduces the transistor count by 25.3% and 18.1% compared with LP2/2-SR and Sync. APCDP uses a mixture of dual-rail domino logic and single-rail domino logic. The single-rail domino logic gates in the noncritical data paths save a lot of the transistor count. Although the SLGs and SLGLs in APCDP consume more transistors than traditional dual-rail domino gates, they are in a small quantity (only 56, used in the critical data path), which have small impact on the transistor count.

The results also show that the transistor count of LP2/2-SR is larger than that of synchronous pipeline, which indicates that conventional latchless pipelines are difficult to realize the potential advantage of small silicon area. The main reason is the dual-rail encoding overhead in domino data path. Because of the latchless feature, a lot of implicit storage elements (dual-rail domino buffers) have to be added at each pipeline stage to store data. These added dual-rail domino buffers cause a large overhead. Although LP2/2-SR is significantly more area efficient than LP2/2, it still consumes more transistors than Sync and Sync-CG.

In addition to the transistor count, total FET width is a better metric to figure the relative difference of capacitance between designs. Table IV shows that APCDP, respectively, reduces the total FET width by 34.8% and 39.3% compared with LP2/2-SR and Sync. An interesting result is that LP2/2-SR has a smaller total FET width even it has a larger transistor count compared with Sync. This is because domino gates use keepers to protect charge leakage problem [20]. These keepers have a very small transistor size.

2) Latency: APCDP and LP22-SR have about one-third lower forward latency than Sync and Sync-CG. This is because latchless design has no sequential overhead (no registers or latches) on its forward path. Compared with LP22-SR, APCDP has a little larger latency. The latency is sacrificed for constructing the stable critical data path. Fortunately, this degradation is not serious.

3) Throughput: The performances of throughput are evaluated without considering design margins, which are all ideal results from the schematic simulations.

The results show that LP2/2-SR has the best throughput performance. This benefits from the bundled-data asynchronous design of LP2/2-SR. Traditional dual-rail domino data paths in LP2/2-SR actually have better signal transition speed than the data paths composed of SLGs/SLGLs. Bundled-data design can exploit this benefit to increase the pipeline throughput. However, delay margins need to be added in practical bundled-data design, which would decrease the performance of throughput.

Because of the dual-rail critical data path, APCDP does not have to add design margins in practical design. In Section III-D1, it shows that the pipeline structure of APCDP originally supplies some time margins. Although APCDP has a slower pipeline speed and a higher forward latency than LP2/2-SR in the ideal evaluation, it is possible that APCDP may have a faster pipeline speed and a lower latency than LP2/2-SR in practical design if the timing margins required in LP2/2-SR exceed the detection overhead in APCDP. Especially, the delay of function blocks is not known until the circuit layout has been generated. This can make the estimation of matching delay in LP2/2-SR overly conservative with a negative impact on performance.

The throughputs of Sync and Sync-CG relate to the pipeline granularity. Although the throughput performance can be improved using fine-grain design, the power consumption increases simultaneously. Therefore, Sync and Sync-CG are carefully designed considering the tradeoff between throughput and power. Although Sync and Sync-CG have the same throughput performance with APCDP, they hardly can win APCDP in practical design because synchronous design has to add design margins.

4) Power: The energy consumption of VLSI circuits relates to the toggling rate in data paths. In APCDP, the adoption of single-rail domino gates in the noncritical data paths saves not only silicon area by reducing transistor count but also energy consumption by reducing the toggling rate.

Fig. 13 shows the energy consumption per cycle for processing different data patterns. Each energy consumption is an average value calculated from 100 cycles. The results show that APCDP consumes much less energy than LP2/2-SR. The adoption of single-rail domino gates in APCDP reduces the toggle rate in data paths since single-rail domino logic does not toggle when transferring low-voltage signal. Besides, the toggling rate relates to the injected data patterns. Therefore, the energy consumption of APCDP varies a lot according to different data patterns. On the other hand, the energy consumption of LP2/2-SR remains almost constant. LP2/2-SR's full dual-rail domino data paths have almost a

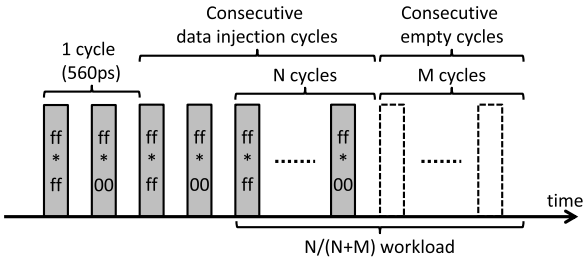


Fig. 15. Workload definition.

constant toggling rate regardless of the injected data patterns. Compared with LP2/2-SR, APCDP saves up to 60.2% of energy in the best case when processing data pattern ff*00 (hexadecimal digit) and 24.5% of energy in the worst case when processing data pattern ff*ff.

The energy consumption of Sync-CG is also evaluated. However, Sync-CG is designed using static logic. Different from domino logic, static logic does not have an initial state. To provide a relative fair comparison, we set the initial state of Sync-CG as low-voltage inputs. The injected data patterns are a certain data pattern (e.g., ff*ff) following with an initial pattern ff*00. The energy consumption per cycle is calculated from 100 cycles. The results show that Sync-CG has lower energy consumption than LP22-SR in most cases. However, Sync-CG also consumes more energy than LP2/2-SR when the toggling rate is high (processing ff*ff). Since the toggling rate in Sync-CG relates to the initial state, the practical energy consumption might be better or even worse than the evaluated results. In our evaluated cases, APCDP shows better energy consumption than Sync-CG.

Furthermore, the power performance with different workloads is evaluated. Fig. 14 shows the performance of power consumption when all designs operate at 3.6G dataset/s. The solid lines show the power consumption when the injected data patterns are recurring between ff*ff $\rightleftharpoons$ ff*00. The dotted lines shows the power consumption when the injected data patterns are recurring between ff*0f $\rightleftharpoons$ ff*00. The workload refers to the rate of the number of active-state cycles to the total number of cycles. In our case, the workload is calculated based on a period of consecutive data injection cycles (active-state cycles) following consecutive empty cycles. Fig. 15 shows the workload definition. The workload is calculated as $N/(N+M)$, where N is the number of consecutive data injection cycles and M is the number of consecutive empty cycles.

The solid and dotted lines, respectively, show that APCDP reduces the power by 41.6% and 52.9% compared with LP2/2-SR. This evaluation also verifies that Sync and Sync-CG have a better performance of power than LP2/2-SR in most situations, except for processing data pattern ff*ff. However, Sync and Sync-CG can hardly win APCDP. APCDP saves up to 43.9% and 38.6% of power compared with Sync-CG when, respectively, processing ff*ff and ff*0f. In addition, the results also show that Sync-CG saves a lot of clock power compared with Sync. However, because of the clock-gating design, Sync-CG consumes a little more power than Sync when the circuits work at peak speed.

C. Further Discussion

The above evaluation is based on schematic simulation. To apply APCDP to large function modules, design automation is an important issue that has to be solved. Design automation for asynchronous circuits is a very large research domain. It is difficult to fully cover this topic. The interested reader is referred to [21]. Here, we just simply address several specific design automation issues in APCDP.

One issue is the design automation for constructing a stable critical data path. Based on the proposed construction method, design automation flow can be easily developed. First, the gate with the largest number of inputs in each stage and connection of gates between stages is identified. Then, the proper gates using SLGs or SLGLs are replaced according to the gate connection information and optimization methods.

Another issue is the design automation for layout. Standard placement and routing (P&R) tools tend to reduce the worst case delay in function blocks by optimizing circuit layout. Such optimization is not suitable for APCDP since it would degrade the robustness of the constructed critical data path. To avoid this problem, a specific P&R method has to be developed to increase the delay on the constructed path.

The last issue is timing verification. There are almost no EDA support for verifying domino circuits because charge sharing and uncertainty about worst case delay makes static timing analysis (STA) very complex [21]. In APCDP, the known critical data path and the dual-rail handshake control logic help in easing the timing analysis problem. After creating a high-quality domino gate library, it is possible to apply STA for verifying the timing constraints in APCDP.

V. Conclusion

This paper introduced a novel design method of asynchronous domino logic pipeline. The pipeline is realized based on a constructed critical data path. The design method greatly reduces the overhead of handshake control logic as well as function block logic, which not only increases the pipeline throughput but also decreases the power consumption. The evaluation results show that the proposed design has better performance than a bundled-data asynchronous domino logic pipeline (LP2/2-SR). It is even comparable with a synchronous pipeline with sequential clock gating.

References

[1] B. H. Calhoun, Y. Cao, X. Li, K. Mai, L. T. Pileggi, and R. A. Rutenbar, "Digital circuit design challenges and opportunities in the era of nanoscale CMOS," *Proc. IEEE*, vol. 96, no. 2, pp. 343–365, Feb. 2008.

[2] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*. Boston, MA, USA: Kluwer, 2001.

[3] M. Krstic, E. Grass, F. K. Gurkaynak, and P. Vivet, "Globally asynchronous, locally synchronous circuits: Overview and outlook," *IEEE Des. Test Comput.*, vol. 24, no. 5, pp. 430–441, Sep./Oct. 2007.

[4] A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proc. IEEE*, vol. 94, no. 6, pp. 1089–1120, Jun. 2006.

[5] J. Teifel and R. Manohar, "An asynchronous dataflow FPGA architecture," *IEEE Trans. Comput.*, vol. 53, no. 11, pp. 1376–1392, Nov. 2004.

[6] H. S. Low, D. Shang, F. Xia, and A. Yakovlev, "Variation tolerant AFPGA architecture," in *Proc. ASYNC*, 2011, pp. 77–86.

[7] M. Hariyama, S. Ishihara, and M. Kameyama, "Evaluation of a field-programmable VLSI based on an asynchronous bit-serial architecture," *IEICE Trans. Electron.*, vol. E91-C, no. 9, pp. 1419–1426, Sep. 2008.

[8] T. E. Williams, "Self-timed rings and their application to division," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Jun. 1991.

[9] A. M. Lines, "Pipelined asynchronous circuits," Dept. Comput. Sci., California Inst. Technol., Pasadena, CA, USA, Tech. Rep., 1998.

[10] S. M. Nowick and M. Singh, "High-performance asynchronous pipelines an overview," *IEEE Des. Test Comput.*, vol. 28, no. 5, pp. 8–22, Sep./Oct. 2011.

[11] M. Singh and S. M. Nowick, "The design of high-performance dynamic asynchronous pipelines: Lookahead style," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 11, pp. 1256–1269, Nov. 2007.

[12] 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," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 7, pp. 1043–1056, Jul. 2010.

[13] D. Harris, *Introduction to CMOS VLSI Design—Lecture 9: Circuit Families* (Lecture Notes on the Subject). Claremont, CA, USA: Harvey Mudd College, 2004.

[14] Z. Xia, S. Ishihara, M. Hariyama, and M. Kameyama, "Synchronising logic gates for wave-pipelining design," *IEE Electron. Lett.*, vol. 46, no. 16, pp. 1116–1117, Aug. 2010.

[15] S. Ahuja and S. Shukla, "MCBCG: Model checking based sequential clock-gating," in *Proc. High Level Des. Validation Test Workshop*, 2009, pp. 20–25.

[16] L. Li, W. Wang, K. Choi, S. Park, and M.-K. Chung, "SeSCG: Selective sequential clock gating for ultra-low-power multimedia mobile processor," in *Proc. IEEE Int. Conf. EIT*, May 2010, pp. 1–6.

[17] Z. Xia, S. Ishihara, M. Hariyama, and M. Kameyama, "Dual-rail/single-rail hybrid logic design for high-performance asynchronous circuit," in *Proc. IEEE ISCAS*, May 2012, pp. 3017–3020.

[18] Z. Xia, S. Ishihara, M. Hariyama, and M. Kameyama, "Design of high-performance asynchronous pipeline using synchronizing logic gates," *IEICE Trans. Electron.*, vol. E95-C, no. 8, pp. 1434–1443, Aug. 2012.

[19] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Mateo, CA, USA: Morgan Kaufmann, 1999.

[20] P. Srivastava, A. Pua, and L. Welch, "Issues in the design of domino logic circuits," in *Proc. 8th Great Lakes Symp. VLSI*, Feb. 1998, pp. 108–112.

[21] A. Taubin, J. Cortadella, L. Lavagno, A. Kondratyev, and A. Peeters, "Design automation of real-life asynchronous devices and systems," *Found. Trends Electron. Des. Autom.*, vol. 2, no. 1, pp. 1–133, 2007.

Zhengfan Xia received the B.E. degree in electronic and information engineering from the China University of Geosciences, Beijing, China, and the M.S. and Ph.D. degrees in information science from Tohoku University, Sendai, Japan, in 2008, 2011, and 2014, respectively.

He is currently with the Toshiba Research and Development Center, Japan. His current research interests include asynchronous circuits, computer architecture, and security.

Masanori Hariyama received the B.E. degree in electronic engineering and the M.S. and Ph.D. degrees in information science from Tohoku University, Sendai, Japan, in 1992, 1994, and 1997, respectively.

He is currently an Associate Professor with the Graduate School of Information Sciences, Tohoku University. His current research interests include VLSI computing for real-world application, such as robots, high-level design methodology for VLSIs, and reconfigurable computing.

Michitaka Kameyama received the B.E., M.E., and D.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1973, 1975, and 1978, respectively.

He has been a Professor with Tohoku University since 1991, where he was the Dean of the Graduate School of Information Sciences from 2010 to 2014. His current research interests include intelligent integrated systems for real-world applications, advanced VLSI architecture, and new-concept VLSI, including multiple-valued VLSI computing.

Prof. Kameyama was a recipient of the Outstanding Paper Awards at the IEEE International Symposiums on Multiple-Valued Logic in 1984, 1985, 1987, and 1989, the Technically Excellent Award from the Society of Instrument and Control Engineers of Japan in 1986, the Outstanding Transactions Paper Award from the Institute of Electronics, Information and Communication Engineers in 1989, the Technically Excellent Award from the Robotics Society of Japan in 1990, the Special Award at the 9th LSI Design of the Year in 2002, and the Outstanding Paper Award at the IADIS International Conference in 2013.