POWER ANALYSIS OF CRITICAL PATH DELAY DESIGN USING DOMINO LOGIC

R.Yamini, V.Kavitha, S.Sarmila, Anila Ramachandran, 
Assistant Professor, ECE Dept, M.E Student, M.E. Student, M.E. Student 
Sri Eshwar College of Engineering, Coimbatore

ABSTRACT
This paper presents a high-throughput and ultralow-power asynchronous domino logic pipeline design method, targeting to latch-free pipeline 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 method saves power by reducing the overhead of logic circuits. Three phase Dual-rail precharge logic (TDPL) is used for evaluating the proposed pipeline method. Compared with asynchronous domino logic pipeline design, the proposed pipeline respectively saves up to 25% of energy 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 towards variations in supply voltage, temperature and fabrication process parameters.

In asynchronous design, the choice of handshake protocols affects the circuit implementation (area, speed, robustness, power, 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 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 latch
less 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 non-inverting logic can be implemented.

As a 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 latch less 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. 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 asynchronous domino logic pipeline has a small overhead in both handshake control logic and function block logic, which greatly improves the circuit efficiency.

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. 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 III focuses on comparison of different parameters of the pipeline structures. Section IV presents the conclusion.

**TABLE I**

<table>
<thead>
<tr>
<th>Code Word</th>
<th>(x_1)</th>
<th>(x_0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 0</td>
<td>(0, 1)</td>
<td></td>
</tr>
<tr>
<td>Data 1</td>
<td>(1, 0)</td>
<td></td>
</tr>
<tr>
<td>Spacer</td>
<td>(0, 0)</td>
<td></td>
</tr>
<tr>
<td>Not used</td>
<td>(1, 1)</td>
<td></td>
</tr>
</tbody>
</table>

![Fig. 1. Block diagram of PS0.](image)

**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 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 look-ahead 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 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 detection, and one precharge in the complete cycle for a pipeline stage. The pipeline cycle time \(T_{cycle}\) is

\[
T_{cycle} = 3 \times t_{Eval} + 2 \times t_{CD} + \epsilon_{Prech}
\]

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.
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 (Di) and one on the output side (Do). 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 (Di), a PCHB stage does not start evaluation until all input bits are valid. This design absorbs skew across individual bits in the stage: one on the input side (Di) and one on the output side (Do).

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 (Di), a PCHB stage does not start evaluation until all input bits are valid. This design absorbs skew across individual bits in the data paths. Although such 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.

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 over-head problem by implementing a single extra bundling signal. The bundling signal serves as 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].

3) ASYNCHRONOUS PIPELINE DESIGN BASED ON CONSTRUCTED CRITICAL DATA PATH

A. Overview

Fig.7 shows the block diagram of the asynchronous pipeline (APCDP) method. The pipeline is designed based on a stable critical data path that is constructed using 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.

APDCP is more familiar to bundled-data asynchronous pipeline because the critical data path essentially works as a matching delay, which controls the correct data transfer in noncritical data paths. This design has many advantages. First, the matching delay is accurate.

The matching delay in APDCP 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.

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\(_f\).

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 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.
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 APDCP
Fig. 8 shows the structure of APDCP. 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 Lin gate) that has the largest number of inputs in each pipeline stage;
2) Changing these Lin 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.

Linking each pipeline stage’s SLG together is partially done in the process of selecting Lin gate in each pipeline stage. When searching Lin gate, there might be more than one option. It is best to select the Lin gate that is originally linked to the Lin gate in the following pipeline stage. After changing these Lin gates to SLGs, SLGs are naturally linked. For example, the linkage between Stage1 and Stage2 in Fig.8. However, if we cannot find the linked Lin 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.

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: The pipeline structure of APCDP is quite robust since the hold time 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. 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 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 fan out by many CMOS designer. As a result, the delay of a logic gate is expressed as

\[
\text{Delay} = f + p = gh + p = g \left( \frac{C_{\text{out}}}{C_{\text{in}}} \right) + p
\]

Where \( C_{\text{out}} \) is the capacitance that loads the output of the logic gate and \( C_{\text{in}} \) 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 effort $g$. 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 fan out $C_{out}$, 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.

The drawback is the extra overhead of transistors. In addition, the use of domino logic introduces many design risks because it is very sensitive to noise, circuit, and layout topologies [20]. 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. On the other hand, larger stack design deteriorates the noise problems and the pipeline speed. The merit is the reduced number of pipeline stages. To reduce the power consumption and area further we have an idea to introduce a method called Three Phase Dual-rail Precharge Logic (TDPL) with large fan-in gates which include 3 phases such as Evaluation, Discharge and Pre-Charge. This logic will finish the evaluation phase before its previous stage starts precharge. The method will increase the processing speed. The large fan-in gates also consume less power.

III. COMPARISON OF DIFFERENT PARAMETERS OF THE PIPELINE STRUCTURES

Fig.9 shows the simulation result which is obtained using Tanner13. PCHB is the timing robust pipeline design. It works in parallel processing. It will start the evaluation only if all the inputs are valid. It doesn’t require any delay assumptions by the designer. The robustness comes at the expense of performance. This method consumes less power when compared to pipelined 4-bit ripple carry adder. Fig.10 shows the simulation result of Pipelined 4-bit ripple carry adder. This adder works with the concept called pipelining. Hence the power consumption is more.

Also the simulation results of LP2/2 and APCDP are given in Fig.11 and Fig.12. We have done the analysis of these pipeline structures with different parameters such as Average power consumption, Static power, Power Delay Product and Energy Delay Product. Table II shows the comparison of parameters between PCHB, LP2/2 and APCDP Structures. From this table we can conclude that the proposed pipeline method APCDP consumes less power when compared to the other...
methods. The analysis also shows that the energy which is required to construct the critical data path for APCDP is minimum.

![Fig.11. LP2/2 Design](image)

![Fig.12.APCDP-Waveform](image)

### TABLE II COMPARISON OF PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>PCHB</th>
<th>LP2/2</th>
<th>APCDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Power</td>
<td>6.8714 mw</td>
<td>2.913 mw</td>
<td>2.368 mw</td>
</tr>
<tr>
<td>Static power</td>
<td>22.60 mw</td>
<td>9.41 mw</td>
<td>7.88 mw</td>
</tr>
<tr>
<td>Power Delay Product(PDP)</td>
<td>0.452 µws</td>
<td>283.165 nws</td>
<td>39.447 nws</td>
</tr>
<tr>
<td>Power Delay Product(EDP)</td>
<td>9.041 pws</td>
<td>8.51195 pws</td>
<td>0.1973 pws</td>
</tr>
</tbody>
</table>

### IV. 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 result shows that the proposed design has better performance than a bundled-data asynchronous domino logic pipeline. It is even comparable with a synchronous pipeline with sequential clock gating.

### REFERENCES


www.ijaert.org