Received 22 September 2023, accepted 18 October 2023, date of publication 23 October 2023, date of current version 1 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3326856

RESEARCH ARTICLE

HPSAP: A High-Performance and Synthesizable Asynchronous Pipeline With Quasi-2phase Conversion Method

XIQIN TANG[1,2], JINGYU WANG[1,2], (Graduate Student Member, IEEE), YANG LI[2], JINHUA HU[3], FEI XIA[4], AND DELONG SHANG[1,3]

[1]Institute of Microelectronics, Chinese Academy of Sciences (IMECAS), Chaoyang, Beijing 100029, China
[2]University of Chinese Academy of Sciences (UCAS), Shijingshan, Beijing 100049, China
[3]Nanjing Institute of Intelligent Technology, Jiangning, Nanjing 211100, China
[4]Newcastle University, Newcastle upon Tyne, NE1 7RU Tyne and Wear, U.K.

Corresponding author: Delong Shang (shangdelong@ime.ac.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2021ZD0200300.

ABSTRACT This paper presents a high-performance and synthesizable asynchronous pipeline (HPSAP). First, a 4-phase pipeline controlled by the relative-timing (RT) controller is designed. The controller is small (7 gates) and its handshake protocol is highly concurrent, resulting in fewer component delays in cycle time. However, the RT pipeline's throughput is limited by the inherent reset phase in the 4-phase protocol. Thus, the variant HPSAP pipeline is proposed with the quasi-2phase conversion method. Unlike other existing solutions which aimed at reducing the reset time of the 4-phase protocol, this method imitates the behavior of the 2-phase pipeline and re-activates the reset edge by two steps: 1) replacing all delay-matched units with the maximum delay; 2) adding a small pulse generator on each RT controller. The post-layout HSPICE simulation of a 4-bit 10-stage HPSAP first-in-first-out (FIFO) pipeline indicated a throughput of 5.382 giga data item per second (GDI/s) under SMIC 55nm CMOS technology, which was 77.5% and 14.65% higher than the Click pipeline and Mousetrap pipeline. In the pipeline with processing, a 32 $\times$ 32 bits multiplier was built, and the maximum working frequency of the HPSAP multiplier was faster than Mousetrap and the synchronous counterparts.

INDEX TERMS High-performance, asynchronous pipelines, relative-timing, asynchronous controller, 4-phase handshake protocol.

I. INTRODUCTION

Most of today's high-performance microprocessors adopt the pipelined structure to increase throughput, with registers inserted between two adjacent stages to store the intermediate values. However, with the rising complexity of systems, clock-based architectures may introduce much energy, redundant timing margins, and complex clock distribution. Whereas asynchronous circuits mainly rely on handshake protocols, which naturally eliminate the global clock. Moreover, it has some benefits like lower power dissipation, reduced electromagnetic emission, and better modularity, which are beneficial to a high-performance system in today's ASIC designs [1], [2].

Unfortunately, asynchronous circuits are not widely adopted due to the lack of mature Electronic Design Automation (EDA) tools. An effective solution is to use conventional synchronous EDA tools for asynchronous circuit synthesis such as Synopsys Design Compiler (DC) and IC Compiler (ICC), but this approach requires that all components in the asynchronous circuit are standard cells.

In general, asynchronous designs can be divided into two types: quasi-delay insensitive (QDI) circuits and bundled-data (BD) circuits. Normally, the QDI circuits require many completion detectors to ensure the correct operation in the presence of arbitrary data path delays, resulting in a large

The associate editor coordinating the review of this manuscript and approving it for publication was Dušan Grujić.

area cost and power consumption. Whereas asynchronous BD circuits bundle the separate request and acknowledge wires with data signals, and use timing to ensure the correct operation rather than completion detectors, thus saving the area and energy. Besides, many efforts have been made in the automatic design of asynchronous BD circuits [3], [4], [5], including synthesis, physical implementation, and static timing analysis (STA), making the asynchronous BD pipelines more welcomed by industries.

The primary asynchronous BD pipeline - *micropipelines* is pioneered by Sutherland [6], but it uses slow and complex capture-pass latches that limit performance. Then, some high-performance asynchronous BD pipelines are proposed, such as GasP [7], IPCMOS [8], High Capacity pipeline (HCpipe) [9], Lookahead pipeline (LApipe) [10] and so on. But they are built with dynamic logic and some custom units, which can not be synthesized by conventional EDA tools.

Later, the high-performance Mousetrap pipeline is developed in [11], which only uses standard cells to construct the pipeline stage. The Mousetrap controller is simple and latches are used in the data path for data storage, providing fast data access and small area cost. However, the latch is normally more susceptible to metastability than the register, thus more likely to cause timing issues and data hazards. Most recently, another asynchronous BD pipeline style called Click [12] is proposed which consists of standard cells and uses registers to store data. As data is latched at a specific time, the chances of race conditions are reduced. However, the area overhead and the power consumption of the Click pipeline are relatively large because of the large size of the Click controller (84 transistors in total).

This paper focuses on developing a high-performance asynchronous BD pipeline that is both simple in design and amenable to synthesis. In [13], a small, fast, and synthesizable 4-phase asynchronous controller called RT controller has been introduced and compared with other existing asynchronous controllers. And this work is an extension of the previous work, in this paper, we first built a 4-phase RT pipeline, and the HPSAP pipeline is obtained by converting the 4-phase RT pipeline into a quasi-2phase one with the proposed quasi-2phase conversion method. By adjusting the delay-matching strategy and adding a small pulse generator, the HPSAP pipeline's throughput is improved approximately 1.95 times compared to the original 4-phase RT pipeline, and the energy/item is also reduced by 6.69%, merely at the expense of a 27.58% increase in the number of transistors.

The high throughput of the HPSAP pipeline is mainly due to three factors: 1) simple control logic, 2) highly decoupled protocol, and 3) inherent relative-timing assumptions. The handshake logic in the HPSAP pipeline only consists of nine gates (46 transistors), which is 45.23% less than Click (84 transistors). And the HPSAP pipeline's throughput (5.382 GDI/s) is 14.65% higher than that of the high-performance Mousetrap pipeline in a 4-bit 10-stage liner pipeline without processing.

Moreover, the proposed HPSAP pipeline avoids the use of custom cells and thus can be synthesized by conventional EDA tools. For instance, several methods for Click-based asynchronous circuits design with conventional EDA tools [14], [15] are also feasible in our design.

The rest of the paper consists of five parts: Section II introduces the structure of the 4-phase RT pipeline, operation, and timing constraints; Section III presents the proposed quasi-2phase conversion method and the implementation of the HPSAP pipeline; Section IV describes the automatic asynchronous design flow for the HPSAP pipeline; Section V provides the experiment results and discussion; Section VI is the conclusion.

II. THE 4-PHASE RT PIPELINE

In asynchronous BD pipelines, the data transfer between stages is controlled by handshake protocols such as the 2-phase BD protocol and 4-phase BD protocol [16]. In general, the 2-phase circuits are more efficient in transferring data as both transition edges of the control signals carry functions, but the conventional data storage approach requires a 4-phase control. The 4-phase circuits are relatively simpler and have better robustness to delay variations [17], but the 4-phase protocol requires an RTZ phase which affects performance. There is no general answer on which protocol is the best, depending on the application scenarios of the design.

The RT asynchronous controller (RTAC) proposed in [13] is designed based on the 4-phase protocol, which is characterized by its high speed and small size. Furthermore, it can be easily synthesized by synchronous EDA tools as it is built with standard cells. Therefore, it is chosen to be the basic handshake control unit in our design.

A 4-stage 4-phase RT pipeline is built in Figure 1, which involves a data path and a control path. The blue cycles symbolize pure wires (if there are no logic slices.) or combination logics, while the red rectangles symbolize the delay-matched units that were used to match each stage's worst-case propagation delay. Each register (or group of registers, if the data path has multi-bits) is controlled by a corresponding RTAC, which consists of four 2-input NAND gates, one 3-input NAND gate, one inverter gate, and one 2-input AND gate. When the latch signal Lt arrive, the new data can be stored in registers.

To improve the speed and save area, the correct operation of the 4-phase RT pipeline is guaranteed by relative-timing assumptions rather than completion detection. The highlighted paths (P1-P4) are used to illustrate these relative-timing assumptions, which will be discussed in section II-A.

A. RELATIVE-TIMING ASSUMPTIONS IN THE RT PIPELINE

Figure2 presents the signal transition graph (STG) specification of the RTAC, which is designed to be highly concurrent to improve the throughput. $\{Rin, Ain\}$ and $\{Rout, Aout\}$ are handshake signals on the left and right environment of each RTAC. The symbol "+ (-)" represents a signal transition from logic "0 (1)" to logic "1 (0)". Two types

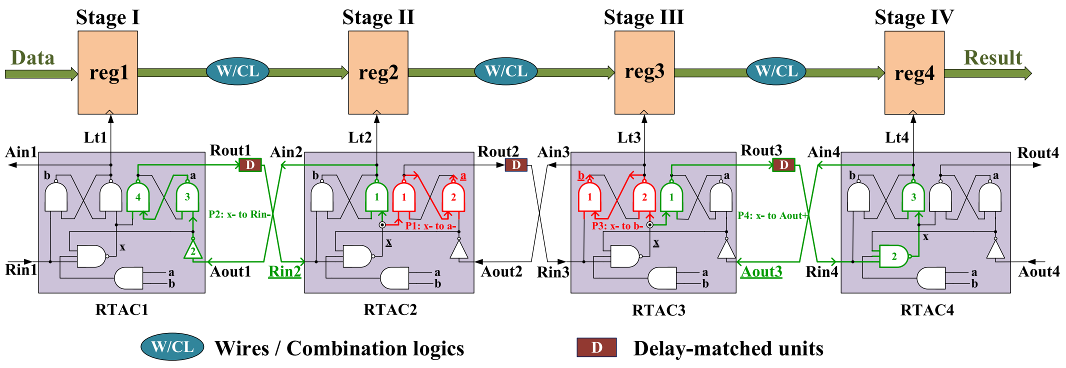


FIGURE 1. The general structure of the 4-phase RT pipeline.

of arrows are used to represent the timing relationships between two adjacent transitions. The solid arrow represents the flow relation ensured by the circuit itself, and the dashed arrow represents the flow relation that maintained by the environment. The red dot represents a token, and the arc marked with a token indicates the start point of the transition sequence.

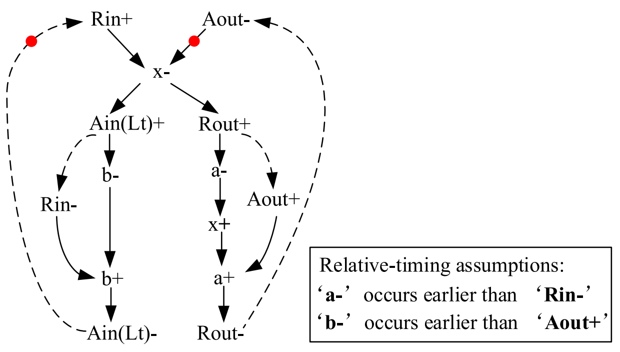
As shown in Figure 2, the STG is highly concurrent since the reset phases of the input and output handshake pairs happen independently. For instance, the transitions $Lt-$ and $Ain-$ can occur before the output request ($Rout+$) has been acknowledged by the next stage ($Aout+$), thus saving the handshake cycle time.

There are two relative-timing assumptions for assuring the appropriate operation of the RT pipeline, which are "**the transition a− should occur earlier than Rin−**" and "**the transition b− should occur earlier than Aout+**". Without the two relative-timing assumptions, the $Aout+$ may happen before the $Ain(Lt)+$, which is unreasonable since the data has not been stored by $Lt+$. And the $Rout+$ may happen after the $Rin-$, which is also incorrect since the request signal has not been propagated to the next stage.

In fact, the two relative-timing assumptions can be easily satisfied by the RT pipeline itself without adding extra circuits. As shown in Figure 1, for the first timing assumption, taking the RTAC2 as an example, the red path (P1) from $x-$ to $a-$ includes two gate delays, while the green path (P2) from $x-$ to $Rin-$ includes four gate delays plus an external delay-matched unit's delay. Normally, local wire delay is much smaller than gate delay in a real ASIC environment [18] and can be ignored for simplicity. Therefore, the $a-$ is ensured to be earlier than $Rin-$.

Similarly, for the second timing assumption, take the RTAC3 as an example, the red path (P3) from $x-$ to $b-$ is explicitly shorter than the green path (P4) from $x-$ to $Aout+$, so the $b-$ is ensured to be earlier than $Aout+$.

Note that the transition $x+$ can actually happen after any of the transitions $Rin-$, $b-$, $Aout+$, or $a-$. However, Figure 2 indicates that the $x+$ is only triggered by $a-$. In fact, this does not conflict with the actual circuit behavior: First, the relative-timing assumptions ensure that the



Relative-timing assumptions:
'**a-**' occurs earlier than '**Rin-**'
'**b-**' occurs earlier than '**Aout+**'

FIGURE 2. The STG of the RTAC with relative-timing assumptions.

transitions $b-$ and $a-$ happen earlier than $Rin-$ and $Aout+$, so the $x+$ would rather be triggered by $a-$ or $b-$ than $Rin-$ or $Aout+$; Second, although the $b-$ may happen earlier than $a-$, which would allow the $x+$ to occur before $a-$, the $x+$ still needs to wait for $a-$ so that the subsequent transition $a+$ can take place. Therefore, the $x+$ is assumed to be mainly decided by $a-$.

It is worth mentioning that Figure 2 is only used for illustrating the circuit function and not for synthesizing asynchronous circuits with asynchronous circuit synthesis tools like Petrify.

B. THE OPERATION OF THE 4-PHASE RT PIPELINE

A timing diagram is presented in Figure3 to illustrate the operation process of the 4-phase RT pipeline in Figure 1. Assume the rising edges of the 4-phase handshake signals are the active edges. The active flow is depicted by the dashed blue arrows and the reset flow is depicted by the dashed green arrows. Initially, all handshake signals $\{Rin, Ain, Rout, Aout\}$ are at a low level (t_0).

Activation Flow: suppose that new data entered the pipeline at t_2, after a slight delay δ (to ensure the data was stable), the external environment sends a request signal $Rin1$ to RTAC1. If the reg1 is empty, the RTAC1 will respond to the sender by sending $Ain1(Lt1)+$, and data is appropriately stored into reg1. Simultaneously, the RTAC1

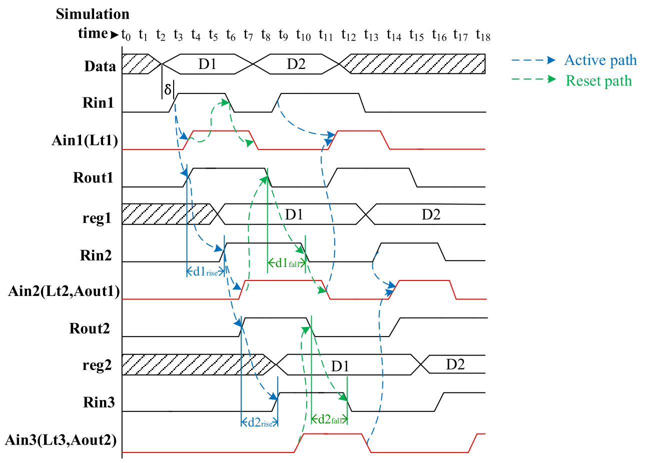


FIGURE 3. Operation timing diagram of the 4-phase RT pipeline.

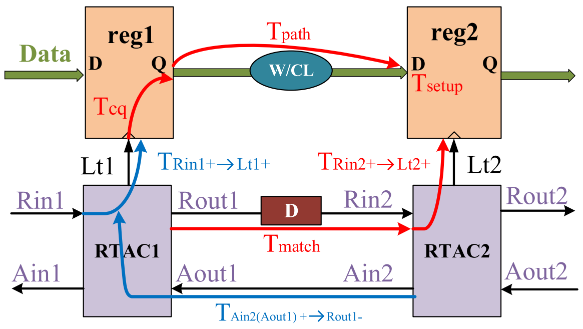


FIGURE 4. Setup and hold timing constraints.

sends the signal $Rout1+$ to RTAC2 to inform that new data has arrived.

Later, the rising transition $Rout1+$ will pass a delay-matched unit with the rising delay $(d1_{rise})$ to become $Rin2+$. The $Ain2+$ and $Rin2+$ will then be triggered by $Rin2+$ in the same way as the $Rin1+$.

Reset Flow: after receiving the acknowledge signal $Ain1+$, the request signal $(Rin1)$ and acknowledgment signal $(Ain1)$ are both reset. Following the $Aout1+$, $Rout1$ is reset. And the falling transition $Rout1-$ becomes $Rin2-$ after the falling delay $(d1_{fall})$. Later, the RTAC2 sends $Aout1-$ in response to $Rin2-$. All handshake signals eventually go back to the initial state.

C. SETUP AND HOLD TIMING CONSTRAINTS

To avoid hazards during data communication, setup and hold timing constraints should be satisfied in the RT pipeline. Figure 4 demonstrates two adjacent stages of the RT pipeline with marked timing paths.

Setup Time Constraint: Data must arrive at the input port of reg2 earlier than the $Lt2$ signal by a setup time T_{setup}, which means the red paths should satisfy the following timing constraint:

$$T_{cq} + T_{path} + T_{setup} < T_{match} + T_{Rin2+\to Lt2+} \tag{1}$$

Hold Time Constraint: The data stored by reg2 must be kept at least a hold time T_{hold} before the arrival of next $Lt1$ signal, as the following inequation 2:

$$T_{Ain2(Aout1)+\to Rout1-} + T_{Rin1+\to Lt1+} + T_{cq} + T_{path} > T_{hold} \tag{2}$$

where T_{path} is the worst-case propagation time in the data path, T_{cq} represents signal propagation delay from the register's clock port to the output port. T_{match} is the latency of the delay-matched unit.

The setup timing constraint could be met by setting a suitable T_{match}, while the hold timing constraint is naturally satisfied because the latency $T_{Ain2(Aout1)+\to Rout1-}$ is explicitly larger than the hold time. The T_{match} could be calculated as

$$T_{match} = T_{cq} + T_{setup} + T_{path} - T_{Rin2+\to Lt2+} \tag{3}$$

The expression 3 also shows that the inner latency of the RTAC $(T_{Rin2+\to Lt2+})$ can be overlapped with the latency of the delay-matched unit to achieve a shorter cycle time.

D. ANALYTIC CYCLE TIME

Cycle time is the time interval between successive data items emerging from the pipeline when the pipeline is operating at maximum speed [19]. The cycle time of the 4-phase RT pipeline can be measured by the interval between two adjacent active transitions of the $Ain(Lt)$ signal, as in the following expression:

$$\begin{aligned} RT_{cycle} = 2T_{match} + T_{Rin2^+\to Ain2^+} + T_{Aout1^+\to Ain1^-} \\ + T_{Rin2^-\to Ain2^-} + T_{Aout1^-\to Ain1^+} \end{aligned} \tag{4}$$

If expression 3 and the gates' delays are substituted into equation 4, a more intuitive expression for the RT pipeline's cycle time is obtained as follows:

$$RT_{cycle} = 2(T_{inv} + T_{nand3} + T_{nand2}) + 2T_{cq} + 2T_{setup} + 2T_{path} \tag{5}$$

III. THE QUASI-2PHASE HPSAP PIPELINE

As shown in expression 4, due to the reset phase, the cycle time of the 4-phase RT pipeline must contain a redundant RTZ process $(T_{Aout1^+\to Ain1^-} + T_{match} + T_{Rin2^-\to Ain2^-})$ which limits the performance. To solve the performance loss issue in 4-phase pipelines, Nowick proposed a "speculative completion" scheme [20]. For the activation phase, signals go through the path with matched delay, while for the reset phase, signals pass through another path without any delay element. However, complex circuitry is required to select among paths, resulting in a large area overhead and an increase in data propagation latency. Other researchers suggested replacing the normal delay-matched unit with the asymmetric delay unit [21]. The core idea is that only the latency of the active transition is matched with the data path, while the latency of the reset transition is set as short as possible. However, the design of the customed asymmetric delay units requires much manual effort in deciding the W/L ratios of the transistors and is not fitted into the automatic design flow.

The above methods focus on minimizing the reset time to improve the speed of a 4-phase circuit, but a better solution is that the reset phase can also be utilized. Therefore, **the quasi-2phase conversion method** is proposed in this article to re-active the reset phase by imitating the latching behavior of the 2-phase circuit. This method simply contains two parts: 1) the reset edge must be aligned with the next signal's active edge as in the 2-phase circuits. And this condition could be ensured by a modified delay-matching strategy. 2) an extra pulse-generation circuit should be added to transform reset edges into local control pulses.

A. DELAY-MATCHING STRATEGY

The delay-matching strategy used in the quasi-2phase conversion method was quite simple, just replacing all the inserted delay-matched units with the maximum delay-matched unit of the control path. The following are illustrations.

Figure 5 shows the control path of a 3-stage 4-phase RT pipeline with left and right environments. $D1 - D4$ are the delay units matched with the worst-case delay of each stage in the data path; $Lt1 - Lt3$ are local control signals; the whole control path forms a closed loop. Two terms (*forward-pass time* and *backward-pass time*) are defined as follows.

Forward-pass time is defined as the time interval of the current stage sending the acknowledge signal Ain to the former stage and receiving another request signal Rin again from the former stage.

Backward-pass time is defined as the time interval of the current stage sending the request signal $Rout$ to the next stage and receiving an acknowledge signal $Aout$ from the next stage.

The two defined terms can also be expressed in the following equations:

$$\begin{aligned} T_{forward} &= T_{Ain_n \rightarrow Aout_{n-1}} + T_{Aout_{n-1} \rightarrow Rout_{n-1}} \\ &\quad + T_{Rout_{n-1} \rightarrow Rin_n} \\ T_{backward} &= T_{Rout_n \rightarrow Rin_{n+1}} + T_{Rin_{n+1} \rightarrow Ain_{n+1}} \\ &\quad + T_{Ain_{n+1} \rightarrow Aout_n} \end{aligned} \tag{6}$$

Assume the reset edge is enabled to trigger a data-latching behavior. In other words, Stage I is supposed to store the first data at $Lt1+$ and store the second data at $Lt1-$, similar to other stages. To ensure the correct data movement, the previous stage could not accept new data until the current stage successfully stores the current data from the previous stage, which means that the forward-pass time should be no shorter than the backward-pass time.

$$\mathrm{T}_{\mathrm{forward}} \geq \mathrm{T}_{\mathrm{backward}} \tag{7}$$

In an RT pipeline, the path delay from Ain to $Aout$ is regarded as zero, and the latency from $Aout$ to $Rout$ is close to the latency from Rin to Ain in the RTAC, so the above timing constraint is mainly decided by the inserted delay-matched units ($D1$ - $D4$), which means the following relationship should be satisfied:

$$D1 \geq D2 \geq D3 \geq D4 \tag{8}$$

Therefore, all inserted delays in the 4-phase RT pipeline must be replaced with the maximum delay ($\mathrm{D_{max}}$) to construct the quasi-2phase HPSAP pipeline:

$$D1 = D2 = D3 = D4 = D_{\max}\{D1, D2, D3, D4\} \tag{9}$$

Note that in an RT pipeline without processing, the condition 9 is naturally met since there is no difference in the delay-matched units.

With this delay-matching strategy, the latch signals $Ain(Lt)$ in the 4-phase RT pipeline can be converted into a more uniform style which is similar to the 2-phase handshake signals. As shown in Figure 6, in the 4-phase RT pipeline, the time interval between two adjacent active edges (i.e. the rising edge) of the Ain signals are each stage's matched delays. Whereas in the quasi-2phase HPSAP pipeline, the time interval between two adjacent edges of the Ain signals is the maximum delay of all delay-matched units.

B. THE PULSE GENERATOR CIRCUIT

Although the reset edge of the 4-phase handshake signal $Ain(Lt)$ of the RT pipeline is set to be valid again by the delay-matching strategy, it cannot be directly used to control the data path. An extra conversion circuit is required to convert the reset edges into control pulses so that they can interact with registers. Figure 7 presents the structure of a 4-stage quasi-2phase HPSAP pipeline, in which the "Pulse-gen" blocks are used to generate the latch signal Lt_pulse by differentiating the request signal Rin and the delayed $Ain(Lt)$ signal.

C. TIMING CONSTRAINTS IN THE HPSAP PIPELINE

To latch the data correctly in the HPSAP pipeline, the Lt_pulse signal must satisfy certain timing constraints. Figure 8 presents the detailed timing diagram of the Pulse-gen circuit and the necessary timing constraints, control pulses are produced at both edges of the Ain signal. The green waveform Ain_delay represents the delayed Ain signal, and the Lt_pulse signal is the result of the $Rin \oplus Ain\_delay$.

The setup and hold timing constraints are ensured by expressions 1 and 2. The other two timing constraints: the minimum pulse width constraint (C1) and the Lt consistency constraint (C2), are illustrated as follows:

C1: The pulse width of the Lt_pulse signal must be larger than the allowed minimum pulse width of the register T_{mpw}, which means:

$$T_{Rin \rightarrow Ain} + d_{buffer} + T_{XOR\downarrow} > T_{XOR\uparrow} + T_{mpw} \tag{10}$$

C2: Once the current Rin transition event is acknowledged, the next Rin transition event should arrive after the signal Lt_pulse- so that Lt_pulse can change consistently with Lt_pulse+ and Lt_pulse-. In other words, the delay from Ain

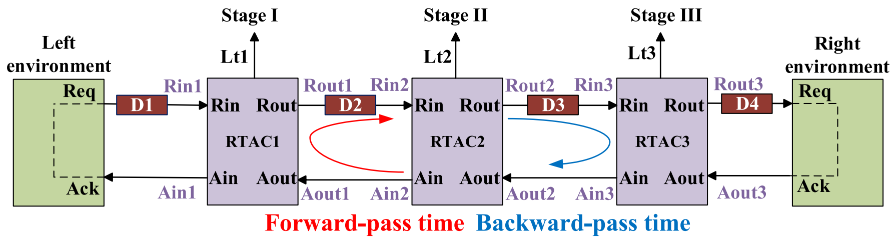


FIGURE 5. A 3-stage RT pipeline's control path with left and right environments.

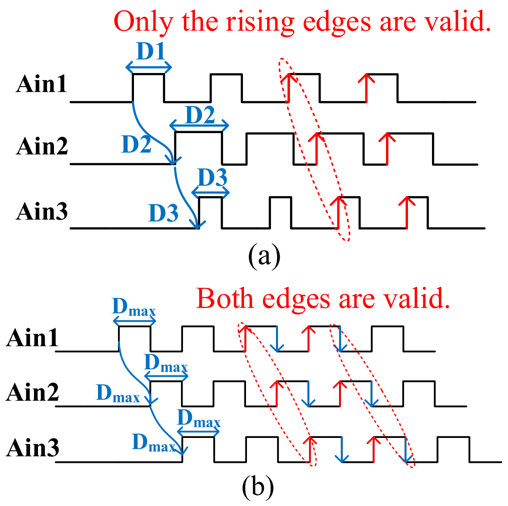


FIGURE 6. (a) The *Ain* signals in the 4-phase RT pipeline. (b) The *Ain* signals in the quasi-2phase HPSAP pipeline.

to *Rin* should be larger than a buffer's delay d_{buffer} plus the falling time of the XOR gate $T_{XOR\downarrow}$:

$$T_{Ain\rightarrow Aout} + T_{Aout\rightarrow Rout} + T_{Rout\rightarrow Rin} > d_{buffer} + T_{XOR\downarrow} \tag{11}$$

C1 is easily satisfied since the XOR gate's falling time is normally close to its rising time and the latency from *Rin* to *Ain* plus a buffer's delay is naturally larger than T_{mpw}. As for **C2**, the inner latency $T_{Ain\rightarrow Aout}$, $T_{Aout\rightarrow Rout}$, d_{buffer}, and $T_{XOR\downarrow}$ are fixed, while the path delay $T_{Rout\rightarrow Rin}$ equals to latency of the delay-matched unit T_{match}. As long as T_{match} satisfies the setup timing constraint in expression 1, **C2** can be satisfied.

D. ANALYTIC CYCLE TIME

In the HPSAP pipeline, the control signal *Lt_pluse* is generated from *Rin*, so the cycle time can be measured by the interval between adjacent rising and falling edges of *Rin*. As the forward path and the backward path of signal transition are highly synchronized, the cycle time of the HPSAP pipeline can be represented as:

$$HPSAP_{cycle} = T_{Rin^{+}\rightarrow Ain^{+}} + T_{Aout^{+}\rightarrow Rout^{-}} + T_{match} \tag{12}$$

Considering expression 3 and the gates' delays, the equation 12 can also be represented as:

$$HPSAP_{cycle} = T_{cq} + T_{setup} + T_{path} + T_{inv} + T_{nand3} + T_{nand2} \tag{13}$$

Compare to the analytic cycle time in equation 5, the quasi-2phase HPSAP pipeline's analytic cycle time is only half of the 4-phase RT pipeline.

IV. ASYNCHRONOUS DESIGN FLOW

As the control element - RTAC used in the RT and HPSAP asynchronous pipeline circuits is built based on standard cells, an asynchronous BD circuit design flow with popular commercial softwares like Verilog Compiler Simulator (VCS), Design Complier (DC), and IC Complier (ICC), was adopted in this work, which is compatible with synchronous sequential circuit design flow. The whole design flow is presented in Figure 9.

First, the asynchronous design architecture specification should be designed and described with HDL. In the next step, the circuit can be synthesized with adaptive delay matching (ADM) method. Then, the layout of the design is obtained with ICC, and the design rule check (DRC), layout versus schematic (LVS) check will be done. Finally, the timing of the circuits is checked with STA based on ADM method.

The data path has no difference as in synchronous circuit, so it can be described in behavior level. However, the elements in control path should be described in gate level with standard cells, and they should be set to not be touched in synthesis phase. Otherwise, the control path may be modified by DC tool, which may cause wrong operation. We used the command set_dont_touch to avoid modification.

The ADM method [14] was adopted to find out the length of delay units to be inserted in the control path to achieve the best performance. Assume the number of pipeline stages is *n*. First, since the exact delay vaslues are unknown before synthesis, *n* delay variables are set using command set_min_delay. The *n* variables are set to be the same initially, so as to the latency between each asynchronous control signals. Then, the circuit is synthesized and the slack value of each timing path could be obtained from the timing report cyclically by using the command $for_in_collection$. If all the slacks have a margin (marked as 'th') of +10% to the corresponding timing path's delay, the synthesis can be done.

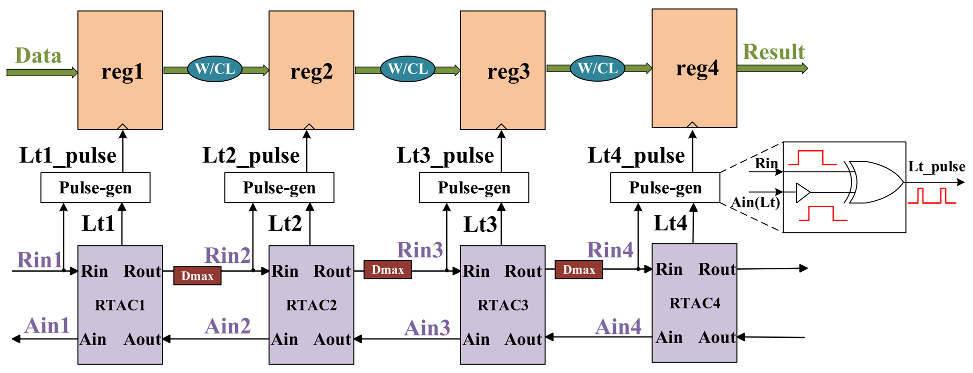


FIGURE 7. The implementation of a 4-stage quasi-2phase HPSAP pipeline.

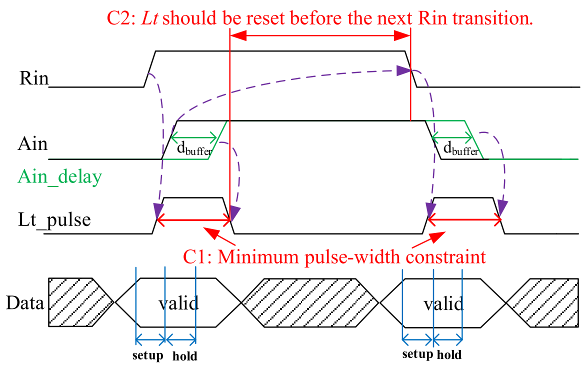

C2: Lt should be reset before the next Rin transition.
Rin
Ain
Ain_delay
Lt_pulse
dbuffer
C1: Minimum pulse-width constraint
Data
valid
setup hold

FIGURE 8. The local timing constraints in the pulse-gen circuit.

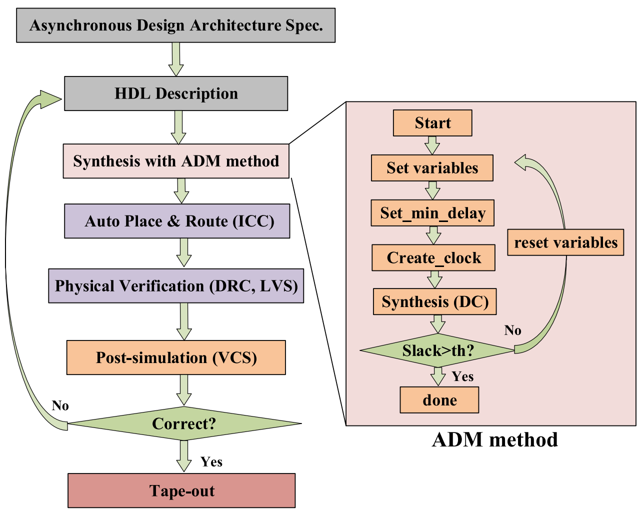


FIGURE 9. The automatic asynchronous BD circuits design flow with ADM method.

Otherwise, the delay variable of each stage will be reset to the value of the current timing path plus the threshold, and the circuit is resynthesized. Normally, if the initial calue is set bigger than zero and smaller than 5ns, only two iterations could complete the synthesis process.

Finally, design rule check (DRC) and layout versus schematic (LVS) check are carried on, and the parasitic parameter is extracted. The post-layout simulation is performed with VCS under all corners, to ensure there is no timing error after place-and route.

Based on the design flow above, we successfully designed a HPSAP pipelined asynchronous 32 $\times$ 32 bits multiplier, which will be presented in section V-C.

V. EXPERIMENTS AND RESULTS

A. EXPERIMENTAL SETTINGS

To verify the performance of the RT and HPSAP pipelines, two classic synthesizable asynchronous BD pipelines were chosen for comparison: the Click pipeline and the Mousetrap pipeline. The studies covered two possible operating conditions: asynchronous pipelines without processing (liner FIFOs) and asynchronous pipelines with processing.

Figure 10 depicts the test environment, which was a n-stage asynchronous liner pipeline with combination logic (CL) or simply wires (W) inserted between adjacent stages. Local control signals (Ctl) produced by different asynchronous controllers (AC) were used to manage each data storage unit (DSU). In the Click, RT, and HPSAP pipelines, the DSU represented the D flip-flop (DFF), while in the Mousetrap pipeline, it represented the D-latch (DL).

The *start* signal was used to activate the request line at the beginning. The data sender was controlled by the input acknowledge signal *Ain*, once the AC1 has acknowledged the current data, a new data could be sent immediately. D_{left} was used to meet the setup timing constraint of DSU1, which matched the time interval from data being produced to the AC1's input. The data receiver was controlled by a delayed *Rout* signal, and D_{right} was used to match the time interval from the data being output to the data receiver's input. Once data was output, it would be latched as soon as possible.

The *Rin* and *Ain* signals formed an input handshake loop ($Ain+ \rightarrow Rin- \rightarrow Ain- \rightarrow Rin+$), while *Rout* and *Aout* signals formed an output handshake loop ($Rout+ \rightarrow Aout+ \rightarrow Rout- \rightarrow Aout-$). Thus, the whole asynchronous FIFO formed a closed loop that allowed data to be transferred successively at the maximum speed. The

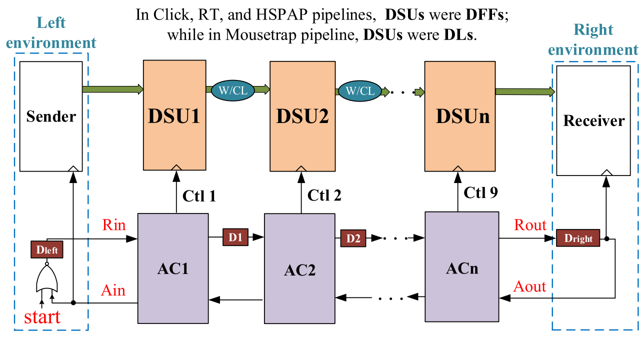


FIGURE 10. The test environment of asynchronous pipelines.

TABLE 1. Comparison in analytic cycle times of asynchronous FIFOs.

Pipelines	DSU	Analytic minimum cycle times	Values
2-ph Click	DFF	$T_{cq} + T_{XNOR\downarrow} + T_{XOR\uparrow} + 2T_{and2}$ 90+63.27+58.61+2×36.32	284.53ps
2-ph Mousetrap	DL	$2T_{latch} + T_{XNOR\uparrow}$ 2×57+59	173ps
4-ph RT	DFF	$2T_{cq} + 2T_{setup} + 2(T_{inv} + T_{nand3} + T_{nand2})$ 2×90+2×24.88+2×(11+24.24+16.83)	333.74ps
quasi-2ph HPSAP	DFF	$T_{cq} + T_{setup} + T_{inv} + T_{nand3} + T_{nand2}$ 90+24.88+11+24.24+16.83	166.71ps

"full" or "empty" state of each FIFO stage was recorded in the acknowledge wire.

All pipelines were implemented under the SMIC 55nm technology with the low voltage threshold (LVT) standard library to obtain a faster speed. The operating conditions were 1.2V nominal voltage supply, 25°C, and a typical process corner. To provide a fair comparison of all the circuits, the simulations were performed for 15ns with identical input data patterns.

B. COMPARISON IN PIPELINES WITHOUT PROCESSING

Before carrying out the experiments in asyncrhonous pipelines without processing (FIFOs), the analytical estimated cycle times of the four asynchronous FIFOs were summarized in Table 1 based on the standard cells' layout delay information recorded in the standard library file. Each standard gate' s latency was taken as the average of the maximum rise and maximum fall delays for simplicity, except for the XNOR and XOR gates. As shown in Table1, the analytic cycle times of different pipelines decrease in the following order: RT, Click, Mousetrap, and HPSAP.

Then, the practical cycle times were obtained from post-layout HSIPCE simulation. The results are listed in Table 2, which were larger than the estimate values in Table 1. This was reasonable since the parasitic elements and interconnect delays were not taken into account in the analytical equations.

As the analytic equations expected, the practical cycle time of the quasi-2phase HPSAP was the smallest and its maximum throughput could reach 5.382 (GDI/s), which is 77.5% higher than Click (3.032 GDI/s) and 14.65% higher than Mousetrap (4.694 GDI/s). The 4-phase RT pipeline gave the least throughput (2.759 GDI/s) mainly due to the extra RTZ phase.

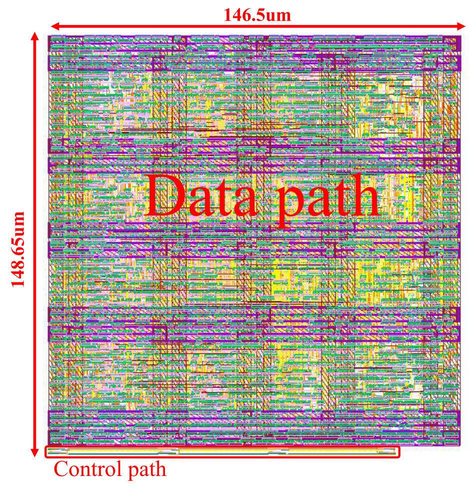


FIGURE 11. Layout of a 32 × 32 bits HPSAP asynchronous multiplier.

TABLE 2. Simulation results of synthesizable asynchronous FIFOs @1.2V, 25°C, typical.

Pipelines	T_{cycle} (ps)	TP (GDI/s)	Power (mW)	α (mW.s^2/GDI^2)	Tran.
Click	329.72	3.032	1.922	0.209	1120
Mousetrap	213	4.694	1.638	0.074	480
RT	362.4	2.759	1.484	0.195	580
HPSAP	**185.8**	**5.382**	**2.703**	**0.093**	**740**

To highlight the performance, the energy consumption was compared using the pipeline system's Figure of Merit (FOM) [22], which can be defined as:

$$\alpha = Power/Throughput^2 \tag{14}$$

A lower α for a pipeline means that the circuit consumes less energy per data item.

As shown in Table 2, the Mousetrap pipeline had the smallest FOM (0.074), followed by the HPSAP pipeline (0.093). The FOM of the HPSAP pipeline was 52.3% smaller than the RT pipeline (0.195).The Click had the largest FOM (0.208) due to its complex structure and more switching activities.

Regarding the area cost, the Mousetrap pipeline had the smallest number of transistors (480) since it used DL to store data, while the others used DFF. However, DL is usually not as stable as DFF because it is more sensitive to metastability. In an asynchronous pipeline, where timing is critical, metastability can cause severe consequences and disrupt the proper functioning of the system. Therefore, it is generally recommended to use DFF instead of DL in an asynchronous pipeline for precise latching data, even though DFF has a larger size.

Compared to the 4-phase RT pipeline, the number of transistors in the quasi-2phase HPSAP pipeline was increased by 27.58% due to the extra pulse generator circuits. The Click pipeline has the largest number of transistors, which was 1.51 times larger than the HPSAP pipeline.

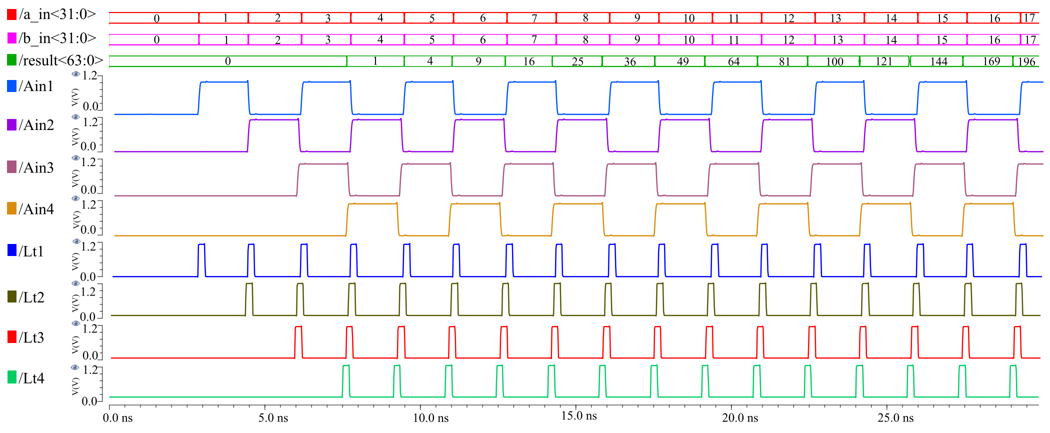


FIGURE 12. **Functional test of the 32 × 32 bits HPSAP multiplier.**

TABLE 3. **Comparison with other existing high-performance asynchronous FIFOs in references.**

Types	Syn.	Node (nm)	TP (GDI/s)	Energy/item (pJ)	Result
LApipe [10]	no	180	1.55	-	sim.
HCpipe [9]	no	180	1.75	24	meas.
APCCD_LP [23]	no	65	5.13	0.56	sim.
SP [24]	no	90	2.227	2.254	sim.
HC [25]	no	180	2.23	-	sim.
S_DELP [26]	yes	90	3.2	2.259	sim.
RT	yes	55	2.759	0.538	sim.
HPSAP	yes	55	**5.382**	**0.502**	sim.

* The listed results in references were all measured/simulated in a 4-bit FIFO at a normal condition, 300K.

1) COMPARISON WITH HIGH-PERFORMANCE ASYNCHRONOUS FIFOS IN REFERENCES

To further verify the performance of the proposed design, some existing high-performance asynchronous pipelines in references were listed for comparison. The listed results were found in references and were all measured/simulated in a 4-bit FIFO at a normal condition, 300K.

As shown in Table 3, the proposed HPSAP pipeline achieves the highest throughput and the lowest energy/item among others. Most importantly, the proposed HPSAP pipeline was synthesizable, while most high-performance asynchronous designs use custom units in their designs that were not supported by automatic design flow.

C. COMPARISON IN PIPELINES WITH PROCESSING

In this experiment, three four-stage 32 × 32 bits Booth multipliers were implemented with different control paths. The combiantion logics in multipler was divided into three parts, including Booth encoding, Wallace-tree reduction, and Carry-lookahead adder. The layouts of HPSAP multiplier, Mousetrap multiplier, and the synchronous multiplier were obtained under the SMIC 55nm CMOS technology. Here, only the asyncrhonous Mousetrap pipeline was choosed for comparison because it has been proved to be faster and smaller than the Click and RT pipelines in the asynchronous

TABLE 4. **Comparison on asynchronou control paths & synchronous clock tree.**

	T_{cycle} (ns)	control path/clock-tree area(um^2)	control path/clock-tree power(uW)	PDP (fJ)	F_{max} (MHz)
Mousetrap	3.152	166.913	**110.596**	348.59	317.258*
Synchronous	3	**83.719**	115.2	345.6	333.3
HPSAP	**2.961**	194.28	113.64	**336.488**	**337.723***

* The maximum frequency of asynchronous pipelines refers to the equivalent frequency, which is calcualted by ($1/T_{cycle}$).

FIFO experiment. The delay-matching units used in the HPSAP, Mousetrap, and synchronous pipelines were symmetric delays (i.e. buffer chains). Figure 11 presents the layout of the 32 × 32 bits HPSAP multiplier, The area of the control path was 194.28um^2, which only occupied 0.89% of the overall area (21777.225um^2).

Figure 12 shows the simulated waveforms of the quasi-2phase HPSAP pipelined multiplier. The local control signals $Lt1*$ (i.e. $Lt1$, $Lt2$, $Lt3$, $Lt4$) were generated based on both edges of the $Ain*$ signals as each transition of $Ain*$ (i.e. $Ain1$, $Ain2$, $Ain3$, $Ain4$) signals corresponded to a data movement. The multiplier produced a result at each rising edge of the $Lt4$.

Then, each asynchronous pipeline's control path and the synchronous pipeline's clock tree were compared in cycle time (T_{cycle}), area, power, power-delay product (PDP), and the maximum frequency (F_{max}). As there was no concept of frequency in asynchronous circuits, the maximum equivalent frequency was used for comparison, which was computed by ($1/T_{cycle}$). The post-simulation results are listed in Table 4.

1) DISCUSSION

As shown in Table 4, the HPSAP multiplier had a performance improvement of 6.45% to Mousetrap. This was reasonable since the cycle time is only decided by the inner latency of the controller and the maximum delay unit. The maximum delays were equal, while the RTAC controller had a smaller inner latency between $Aout$ and $Rout$ signals. The latency from $Aout$ to $Rout$ was only three simple gates' delay in RTAC controller, whereas in Mousetrap, the latency was an XNOR gate plus a latch's delay. However, the area of the HPSAP control path was 16.39% bigger than the Mousetrap control path mainly due to the maximum delay-matching strategy and the larger size of the RTAC controller.

In comparison with the synchronous pipeline, the HPSAP pipeline had a comparable F_{max} (337.723MHz) with F_{max} (333.3MHz) of the synchronous counterpart. And the PDP was reduced by 2.6% compared to synchronous one. However, the area of the HPSAP control path was about 2.32 times larger than the area of clock tree. This was reasinable because the clock network was relatively simple in such a small design, while the asynchronous control path required extra controllers and delay-matching elements to generate the control signals. In a large system, where the distribution of clock network is much complex, the area cost of the clock-tree will increase tremedously for a balanced clock control.

VI. CONCLUSION

This paper proposed a high-performance and synthesizable asynchronous pipeline - the HPSAP pipeline. The correct operation of the HPSAP pipelines was guaranteed by the relative-timing assumptions rather than completion detectors used in most conventional high-performance asynchronous designs. And the key advantage of the proposed HPSAP pipeline is that it can be supported by EDA tools while achieving great performance.

The HPSAP pipeline was compared with other high-performance aynchronous pipelines from two aspects: with and without processing. During the experiments without processing, the HPSAP pipeline achieved the highest throughput (5.382 GDI/s) among other existing high-performance asynchronous pipelines, which was 77.5% higher than the Click pipeline with 55.28% reduction in FOM, and 14.65% higher than the Mousetrap pipeline but with 25.67% increase in FOM. During the experiments with processing, a four-stage 32 $\times$ 32 bits HPSAP multiplier was implemented and compared with the Mousetrap and synchronous counterparts. The post-simulation results showed that the HPSAP multiplier had a performance improvement of 6.45% to Mousetrap but with a 16.39% bigger area cost. The speed of the HPSAP multiplier was comparable with synchronous one with a PDP reduction of 2.6%.

ACKNOWLEDGMENT

The authors are highly grateful to the Development and Construction Management Committee of Nanjing Chi-Lin Technology Innovation Park for hosting the research team. They also like to express thanks to the reviewers for their helpful suggestions.

REFERENCES

[1] S. Xiao, W. Liu, J. Lin, and Z. Yu, "A data-driven asynchronous neural network accelerator," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 9, pp. 1874–1886, Sep. 2021.

[2] J. Zhang, M. Liang, J. Wei, S. Wei, and H. Chen, "A 28 nm configurable asynchronous SNN accelerator with energy-efficient learning," in *Proc. 27th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, Sep. 2021, pp. 34–39.

[3] G. Miorandi, M. Balboni, S. M. Nowick, and D. Bertozzi, "Accurate assessment of bundled-data asynchronous NoCs enabled by a predictable and efficient hierarchical synthesis flow," in *Proc. 23rd IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, May 2017, pp. 10–17.

[4] G. Gimenez, A. Cherkaoui, G. Cogniard, and L. Fesquet, "Static timing analysis of asynchronous bundled-data circuits," in *Proc. 24th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, May 2018, pp. 110–118.

[5] G. Gimenez, J. Simatic, and L. Fesquet, "From signal transition graphs to timing closure: Application to bundled-data circuits," in *Proc. 25th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, May 2019, pp. 86–95.

[6] I. E. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, no. 6, pp. 720–738, Jun. 1989.

[7] I. Sutherland and S. Fairbanks, "GasP: A minimal FIFO control," in *Proc. 7th Int. Symp. Asynchronous Circuits Syst.*, 2001, pp. 46–53.

[8] S. Schuster, W. Reohr, P. Cook, D. Heidel, M. Immediato, and K. Jenkins, "Asynchronous interlocked pipelined CMOS circuits operating at 3.3–4.5 GHz," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2000, pp. 292–293.

[9] M. Singh and S. M. Nowick, "The design of high-performance dynamic asynchronous pipelines: High-capacity style," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 11, pp. 1270–1283, Nov. 2007.

[10] 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.

[11] M. Singh and S. M. Nowick, "MOUSETRAP: High-speed transition-signaling asynchronous pipelines," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 6, pp. 684–698, Jun. 2007.

[12] A. Peeters, F. T. Beest, M. de Wit, and W. Mallon, "Click elements: An implementation style for data-driven compilation," in *Proc. IEEE Symp. Asynchronous Circuits Syst.*, May 2010, pp. 3–14.

[13] X. Tang, Y. Li, W. Liu, S. Qiao, Y. Zhou, and D. Shang, "Relative timing latch controller with significant improvement on power, performance, and robustness," in *Proc. 6th Int. Conf. Integr. Circuits Microsyst. (ICICM)*, Oct. 2021, pp. 95–100.

[14] H. Wu, Z. Su, J. Zhang, S. Wei, Z. Wang, and H. Chen, "A design flow for click-based asynchronous circuits design with conventional EDA tools," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2421–2425, Nov. 2021.

[15] M. Simoes, L. Bossuet, N. Bruneau, V. Grosso, P. Haddad, and T. Sarno, "Low-latency masking with arbitrary protection order based on click elements," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2023, pp. 36–47.

[16] J. Spars, "Asynchronous circuit design a tutorial," Tech. Rep., 2006.

[17] Z. Tabassam, S. R. Naqvi, T. Akram, M. Alhussein, K. Aurangzeb, and S. A. Haider, "Towards designing asynchronous microprocessors: From specification to tape-out, *IEEE Access*, vol. 7, pp. 33978–34003, 2019.

[18] J. V. Manoranjan and K. S. Stevens, "Qualifying relative timing constraints for asynchronous circuits," in *Proc. 22nd IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, May 2016, pp. 91–98.

[19] T. E. Williams, *Self-Timed Rings and Their Application to Division*. Stanford, U.K.: Stanford Univ., 1991.

[20] S. M. Nowick, "Design of a low-latency asynchronous adder using speculative completion," *IEE Proc.-Comput. Digit. Techn.*, vol. 143, no. 5, pp. 301–307, Sep. 1996.

[21] C. P. Sotiriou and L. Lavagno, "De-synchronization: Asynchronous circuits from synchronous specifications," in *Proc. IEEE Int. Syst.-Chip SOC Conf.*, Mar. 2003, pp. 165–168.

[22] M. Gholipour, K. Shojaee, A. Afzali-Kusha, A. Khademzadeh, and M. Nourani, "An efficient model for performance analysis of asynchronous pipeline design methods," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 5234–5237.

[23] Z. Xia, S. Ishihara, M. Hariyama, and M. Kameyama, "Design of high-performance asynchronous pipeline using synchronizing logic gates," *IEICE Trans. Electron.*, vol. 95, no. 8, pp. 1434–1443, 2012.

[24] C. K. Midhun, J. Joy, and R. K. Kavitha, "High-speed dynamic asynchronous pipeline: Self-precharging style," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 10, pp. 2235–2239, Oct. 2014.

[25] K. Sravani and R. Rao, "High throughput and high capacity asynchronous pipeline using hybrid logic," in *Proc. Int. Conf. Innov. Electron., Signal Process. Commun. (IESC)*, Apr. 2017, pp. 11–15.

[26] H. Rezaei, S. A. Moghaddam, and A. Rahmati, "High-performance dynamic elastic pipelines," *Microprocessors Microsyst.*, vol. 56, pp. 113–120, Feb. 2018.

XIQIN TANG was born in Hunan, China. She received the B.Eng. degree in microelectronics and sciences from the North China University of Technology, Beijing, China, in 2020. She is currently pursuing the Ph.D. degree in circuits and systems with the Institute of Microelectronics, Chinese Academy of Sciences, Beijing. Her current research interests include asynchronous design methodology, micro pipelines, and asynchronous microprocessors.

JINGYU WANG (Graduate Student Member, IEEE) received the B.Eng. degree from the School of Microelectronics, Xidian University, Xi'an, China, in 2019. He is currently pursuing the Ph.D. degree in microelectronics and solid-state electronics with the Institute of Microelectronics, Chinese Academy of Sciences, Beijing, China. His current research interests include asynchronous interface and asynchronous communication.

YANG LI was born in Shandong, China. She received the B.Eng. degree in integrated circuit design and systems integration from the Xi'an University of Posts and Telecommunications, China, in 2020, and the M.Sc. degree in electronic information from the University of the Chinese Academy of Sciences. Her current research interests include asynchronous circuits and the design of the high-speed interface between asynchronous chips.

JINHUA HU was born in Hubei, China. She received the B.Arts. degree in English and the M.Sc. degree in aeronautical engineering from Xi'an Jiaotong University, in 2014 and 2020, respectively. Her current research interests include electromagnetic nondestructive testing and chip packaging.

FEI XIA received the B.Eng. degree from Tsinghua University, Beijing, the M.Sc. degree from the University of Alberta, Edmonton, and the Ph.D. degree from King's College, London. He is currently a Senior Research Associate with the School of Engineering, Newcastle University. His current research interests include asynchronous and concurrent systems with an emphasis on power and energy.

DELONG SHANG received the B.Eng. degree from Nanjing University, the M.Eng. degree from the Chinese Academy of Sciences, and the Ph.D. degree from Newcastle University. He is currently a Full Professor with the Institute of Microelectronics, Chinese Academy of Sciences, and the Deputy Director of the Nanjing Institute of Intelligent Technology, IMECAS. His current research interests include computer architecture, brain-inspired computing, asynchronous circuit design, and low-power VLSI.