At-speed DfT Architecture for Bundled-data Design

Ricardo Aquino Guazzelli, Laurent Fesquet

At-speed DfT Architecture for Bundled-data Design

Ricardo Aquino Guazzelli, Laurent Fesquet
Univ. Grenoble Alpes, CNRS, Grenoble INP*, TIMA, 38000 Grenoble, France
Institute of Engineering Univ. Grenoble Alpes
{ricardo.guazzelli,laurent.fesquet}@univ-grenoble-alpes.fr

***Abstract*—At-speed testing for asynchronous circuits is still an open concern in the literature. Due to its timing constraints between control and data paths, Design for Testability (DfT) methodologies must test both control and data paths at the same time in order to guarantee the circuit correctness. As Process Voltage Temperature (PVT) variations significantly impact circuit design in newer CMOS technologies and low-power techniques such as voltage scaling, the timing constraints between control and data paths must be tested after fabrication not only under nominal conditions but through a range of operating conditions. However, this requirement demands modifications in the control and data paths, which are not straightforward and not desirable from a commercial standpoint due to its incompatibility with conventional testing tools. Even with the available testing methodologies for asynchronous circuits in the literature – by adapting the existing techniques for synchronous or creating new ones from scratch – those methodologies usually target the control or data path. This work explores an at-speed testing approach for bundled data circuits, targeting the micropipeline template. The main target of this test approach focuses on whether the sized delay lines in control paths respect the local timing assumptions of the data paths. By adding extra controllability points in the controllers and taking advantage of scan-chain structures, this work targets to generate/stall tokens in controllers, enabling circuit verification through available scan chains.**

I. Introduction

Asynchronous circuits are a promising solution for coping with aggressive process variations faced in the most advanced silicon technology nodes, as they are able to gracefully accommodate wide ranges in gate and wire delays. A similar phenomenon also takes place in more classical technologies when the performance is not an issue and the requirements are mainly driven by power consumption considerations. Indeed, in order to drastically reduce power consumption, power management strategies tend to minimize or aggressively shrink the power supply voltage. In such conditions, the process variations are exacerbated because the operating voltage is near-threshold or, worst, sub-threshold. Operating at very low-voltage makes sense, especially with the emerging Internet of Things (IoT), where the devices may have strong low-power requirements. Most of the ultra-low power circuits operate at very low supply voltages and sometimes in electrical harsh environments, implying less predictable propagation delays and noisy working environments [1]. Delay variations can compromise the circuit functionality, especially if the timing assumptions are strong, which is the case for the synchronous circuits. Indeed, their timing assumption is based on the worst circuit critical path, pushing the designers to over-design and take excessive timing margins. Hence, asynchronous circuits can help designers to avoid such excessive margins by providing an easier timing closure and improving robustness against unexpectedly large delays [2]. The most efficient circuit class to tackle this problem is certainly the Quasi-Delay Insensitive (QDI) circuit class because it only requires a very weak assumption on some circuit forks (known as isochronic forks) [3]. Nevertheless, this class suffers from a large circuit area and requires specific skills and dedicated tools, making its adoption difficult by the industry. In order to overcome these issues, designing bundle-data circuits seems more acceptable because they have a similar area compared to the synchronous circuits, while offering a better robustness to process and voltage variations. As they are really good candidate for playing an important role in low power and ultra-low power circuits, it is important to propose effective solutions for testing the imposed timing constraints of such circuits after fabrication or even in their working environment. Moreover, testing circuits is a mandatory requirement in digital circuits and at-speed becomes especially crucial when the supply voltage is low or changed during operations.

As bundled-data circuits are technically close to their synchronous counterparts, they are more understandable for the designers and can benefit from the commercial electronic design automation (EDA) tools usually used with synchronous design [4]. Based on the existing synchronous Design for Test (DfT) approaches, it is possible to develop a similar framework for asynchronous bundled-data circuits, which could also take advantage of the commercial tools. Thus, testing asynchronous bundle-data circuits requires a limited effort to make them compliant with the EDA tools. It is of course needed to adapt the DfT strategy to their specificity. This paper focuses on an at-speed DfT strategy for micropipeline circuits, which can successfully be used in low-power devices exploiting low or several supply voltages. The proposed technique tries as much as possible to stay in the traditional DfT framework in order to foster its adoption by industry.

II. Asynchronous Bundled-Data Circuits

Taking an asynchronous approach, the literature presents different template designs. These templates are defined according two main characteristics: (i) data encoding; and (ii) the selected handshake protocol. On top of that, those templates are usually organized in two main major design families: bundled-data (BD) and Quasi-Delay Insensitive (QDI). By employing Delay

Insensitive (DI) encoding into its handshake scheme, QDI design provides more relaxed timing constraints, at the cost of heavy area and power trade-off. Bundled-data design, in the other hand, takes an implementation approach close to standard synchronous design, employing single-rail encoding but replacing the global clock signal by local handshake schemes. Figure 1 illustrates a generic asynchronous bundled-data push channel, in which data flows through the data paths according to the handshaking protocol between sender/receiver controllers. The most common handshake protocols employed in bundled-data circuits are (a) four-phase and (b) two-phase protocols, as depicted in Figure 2. With four-phase handshaking, only the rising transition of the request signal req indicates the validity of data and a single transition of the acknowledgement signal ack signifies that data has been captured by the receiver. To end the communication cycle, both req and ack signals must be reset. Two-phase handshaking, however, considers both rising and falling transitions of req to indicate the presence of new data, and both transitions of ack to signifies the data capture by the receiver. Here, there is no necessity to reset the handshake signals as the transition in ack already ends the communication cycle.

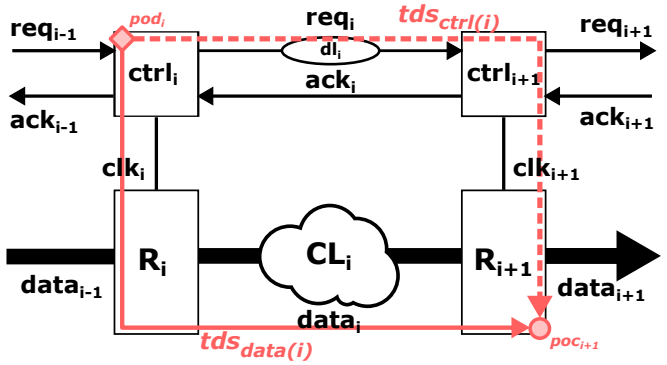

Fig. 1. Generic scheme of a asynchronous bundled-data push channel, highlighting the timing relationship between handshake and data signals.

Due to the handshake schemes, timing constraints between handshake and data signals must be respected in order to guarantee circuit correctness. According to [5], the basic constraints in bundled-data design are: (i) The receiver must not see a request until its input data is stable; (ii) The receiver must not acknowledge to the sender until it has no further need for its input data and (iii) The sender must hold its output data steady between sending the req and receiving the ack. Taking Figure 1 again, consider that valid data is present at $data_{i-1}$ and we desire to propagate it to $data_{i+1}$. In order to do that, the launch controller $ctrl_i$ receives a flag through req_{i-1} (e.g. req_{i-1} rises), indicating valid data is present at the input of its respective registers – in this case, R_i. Usually, this request flag is called 'token', as it carries the information that valid data is present in the data path. Next, R_i stores and propagates data through the combinational block CL_i. At the same time, $ctrl_i$ also propagates the request signal req_i, indicating to the capture controller $ctrl_{i+1}$ in the receiving part that new data is available. If req_i is propagated to $ctrl_{i+1}$, which will pulse clk_{i+1}, before the new data had been completely computed by CL_i, the receiver register R_{i+1} will store invalid data, compromising the circuit operation. This implies that, from the point-of-divergence pod_i at the launch controller, the control path $tds_{ctrl(i)}$ must be tuned in order to match the propagation delay of the launch register and the combinational block $tds_{data(i)}$ at the point-of-convergence poc_{i+1} in the capture controller. In order to do that, a delay line dl_i is inserted into the request signal between the controllers, delaying the arrival of the request signal at the receiving end. This timing constraint between control and data paths is called forward (setup) constraint and can be represented by Equation 1.

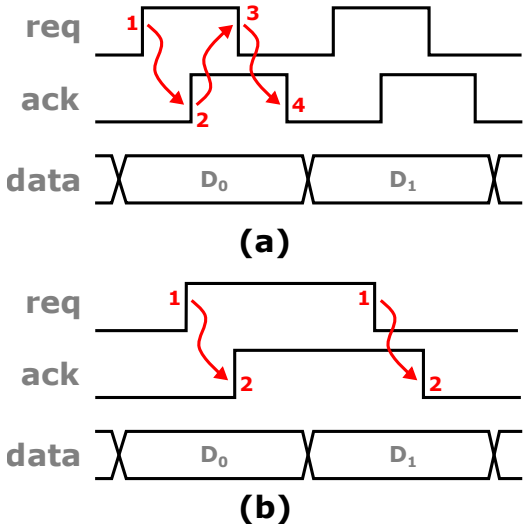

Fig. 2. Asynchronous bundled-data communication employing (a) four-phase and (b) two-phase handshake protocols.

$$tds_{data(i)} < tds_{ctrl(i)} \tag{1}$$

From a testing standpoint, the presence of the control part in bundled-data design infers lower controllability as it is not possible to control the registers directly through a global clock signal. This requires modifications in both control and data paths, specially when it is necessary to verify the local timing constraint between control and data paths.

III. Related Work

The testability of asynchronous circuits is already a topic in discussion in the literature. Several well-known authors in the asynchronous field have presented the issues and solutions while testing bundled-data and QDI circuits. This sections focuses on the test of bundled-data circuits, giving a historical view of how asynchronous testing has been evolving in the last decades.

In 1992, Pagey et al. [6] explores the basics of testing for the micropipeline template. In their work, they indicate that the control part of the circuit is concurrently testable during normal operation and ATPG for the data path can be reduced to that for the combinational circuits. In this case, the authors

suggest to set all latches in pass mode and treat the circuit as a single logic block, allowing the test pattern generation. The micropipeline template is also explored by Khoche and Brunvand in 1994 [7]. The authors propose stuck-at and delay testing in the data path by adding scan functionality to latches and C-elements. One year later, the same authors present a partial scan approach for self-timed circuits [8]. Here, C-elements are not included in the scan path and modified in order to test them as combinational logic. For the scan path, the latches employ two non-overlapping clocks input to provide race free operation during testing – basically, a level-sensitive scan-design (LSSD) structure.

In the same year, Petlin et Furber presents a fully-asynchronous scan test technique [9] for a micropipeline-based microprocessor called AMULET. According to the authors, the techniques allows both stuck-at and delay faults. Moreover, different testing control schemes are presented – called scan test control logic (STCL) – for two and four-phase protocols. Schöber et Kiel also explores a scan path design for micropipeline circuits [10]. They also propose a fully asynchronous scan approach, which in no clock is used during normal and test mode. In order to do that, latches are replaced by asynchronous scan latch (ASL) and an extra two-phase handshaking control permits the scan manipulation (scan in and out). As the test structure is also asynchronous, this implies in a relatively high area overhead due to the extra handshaking controls. Again, Petlin et Furber covers the subject of micropipeline testing in 1997 [11]. In their work, the authors present a built-in self-test (BIST) micropipeline design. The circuit employs an asynchronous built-in logic block observer (BILBO) for stuck-at fault detection. The authors also indicates that their BIST approach can check the timing constraint of the micropipeline template.

In 1996, Roncken et Bruls discuss that the autonomous handshake behavior of asynchronous circuits compromises the test quality [12]. Thus, the authors introduce the handshake component HOLD, which is inserted between handshake controls to add more controllability and better applicability of scan and quiescent current (IDDQ) testing. Roncken et al. [13] also addresses the use of LSSD-based testing design for asynchronous circuits. In their work, the authors propose an algorithm to implement near-optimal scan structures, avoiding the area overhead caused by LSSD. In the end of the 20th century, Roncken takes another step in the subject and presents a defect-oriented testing for asynchronous circuit [14]. This time, Roncken targets IDDQ testing for stuck-at and bridging fault detection. This is done through a HOLD DfT approach, which is responsible to stall the circuit at a desired state and, thus, allowing IDDQ testing in a quiescent state not normally available in an asynchronous design. This kind of testing approach is further addressed in [15], where it introduces the notion of naturalized communication and how to reused the handshake behavior to test two-phase bundled-data circuits, such as micropipeline, GasP, MOUSETRAP and Click templates. The authors present a proper stopper circuit, called MrGO, responsible to controlling actions, freezing or releasing specific handshake parts of the circuit. Details about the MrGO circuitry are presented, but the scan approach used for testing delay and stuck-at faults is not described, the same is true about fault coverage and the area overheads.

Kishinevsky et al. presents a path-delay fault testing algorithm for asynchronous circuits [16]. The algorithm is responsible to generate test sequences for a set of paths in the circuit that must respect a specific timing constraint. Despite the fact that their work covers delay testing, the authors consider only quasi-delay insensitive (QDI) circuits in their experiments.

In 1999, Kang et al. proposes a scan design for micropipeline circuits, focusing on delay testing [17]. The latches are replaced by their proposed scan latch that, according to the authors, allows full control during two-pattern delay testing. However, the proposed scan latch adds three latches (four in total), implying a relative high area overhead.

At the 21th century begin, Berkel et al. explores the insertion of synchronous and LSSD modes to C-elements [18], targeting a more general testing approach of any asynchronous implementation. Here, they consider the scan-path technique and level-sensitive operation of LSSD operation to add full controllability and observability in sequential gates. Consequently, asynchronous circuits could take advantage of traditional DfT and automatic test pattern generation (ATPG) tools. This statement is further explored in [19], [20], where it applies the DfT proposed in [18] and presents fault coverage and area results of five testable bundled-data circuits designed with Tangram, a Philips' design flow for asynchronous circuits. The authors were able to test both data and control paths with 100% and 99% fault coverage, respectivately, with area overhead ranging from 90% (full-scan) to 30% (partial-scan). In other to reduce the impact of LSSD, Beest et Peeters also explores a multiplexer-based testing technique to test C-elements [21]. The authors indicates that the proposed multiplexer-based approach achieves the same structural fault coverage of previous LSSD approaches, as well as profits the automatic pattern generation from conventional testing tools.

Others authors focus mainly on testing the handshaking controls in asynchronous circuits. For instance, King et Saluja [22] adds testability for the control part of micropipeline circuits. By inserting scan latches in the handshake controllers and restraining combinational loops, the proposed technique synchronously treats the existing asynchronous elements – like a finite state machine. This approach brings extra controllability and observability, allowing stuck-at fault detection in the control part at the cost of a signficant area overhead.

Delay testing for the MOUSETRAP [23] template is covered in [24]. The proposed DfT technique consists of controlling the acknowledgement signals between handshake controllers, permitting the user to pause or resume the pipeline at will. To allow this extra handshake control, the proposed technique requires the insertion of multiplexers in the acknowledgement signals and a shift register to configure each multiplexer. Another interesting approach to test the control part of asynchronous circuit is presented in [25]. In this work, a checker logic block is connected in the handshake signals in order to detect any violation in the handshake protocol. For each handshake pair – request and acknowledgement pair signal – the proposed checker logic requires four flip-flops, four David Cell (DC) circuits [26] and four delay elements, implying

that checking all handshake signal brings a significant area overhead.

In 2006, delay testing for asynchronous circuits is covered in [27], which in presents a non-intrusive delay testing technique for three different templates: MOUSETRAP [23], GasP [28] and high-capacity (HC) pipelines. However, the authors focus on the MOUSETRAP template most of the paper, also presenting how to generate test patterns for non-linear MOUSETRAP implementations – containing forks and joins, for example.

Finally, Kuentzer et al. explore the testing characteristics of the resilient bundled-data template called Blade [29], [30]. First, the authors analyze and classify the faults in the error detection logic (EDL) of the Blade template [29], proposing slight modification in the EDL design to increase controllability/observability. As the resilient architecture of the Blade template is capable to detect timing violations during operation, it also implies that the architecture also enables online delay testing of critical paths [30].

This work proposes a DfT architecture for at-speed testing, while taking in consideration the compatibility with stuck-at testing using traditional testing tools. Taking the same approach of [15], we desire to freeze and release specific control parts of the circuit and test their respective data path. This is done by adding controllability in the controllers, allowing to control directly the points-of-divergence of the forward timing constraint between control and data paths. Despite the proposed testing architecture imposes modifications in the controllers' design to add controllability, it requires only the use of logic gates already available in off-the-shelf standard cell libraries. The proposed architecture is also compatible with the Local Clock Set (LCS) flow [4] The LCS flow enables the implementation of bundled-data circuits with standard EDA tools, allowing the use of scan-chain insertion and static timing analysis (STA) features. Moreover, the STA enables better interface with ATPG tools for path-delay testing.

IV. Proposed Testing Architecture for Micropipeline Design

The main objective to the proposed architecture is to allow verifying whether the timing constraint imposed in Equation 1 has been respected after fabrication. Figure 3 illustrates the proposed testing structure with a push channel structure. Regarding testing, the circuit has multiple testing signals:

- Test mode (T_{mode}): when enabled, it bypasses the control logic and allows to control register with an external clock signal T_{clk}. This signal is usually added for stuck-at testing with scan chains;
- Scan enable (SC_en): when enabled, it bypasses the combinational logic presented in the circuit data paths, allowing to load and unload the scannable registers in the circuit. This signal is also presented for stuck-at testing with scan chains;
- Handshake breakers (HSB_i): this signal allows to disable or enable the left handshake signals of a given controller. When enabled, this signal disables the left handshake signals and permits external control of the left request signal of the controller through ext_req. When disabled, the left handshake signals are connected to the previous stage(s) and the controller operates normally;
- External request signal (ext_req): this signal is added specially for the at-speed testing methodology in this work. During at-speed testing, this signal is responsible to internally propagate tokens in control paths.

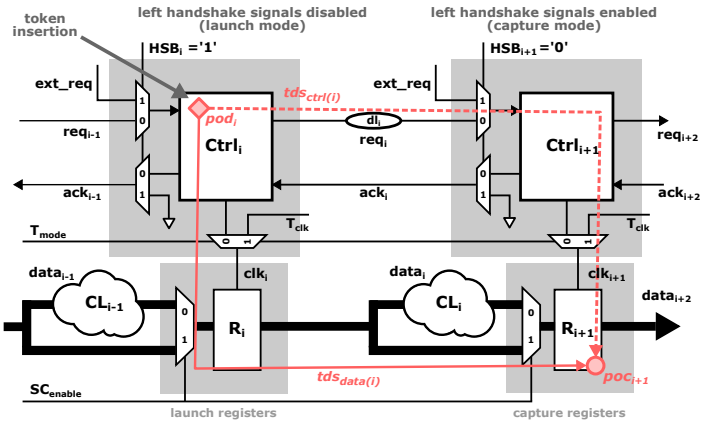

Fig. 3. Proposed testing structure. Lower MUX logic controlled by T_{mode} and SC_{en} are already presented for stuck-at testing. The proposed testing structure adds two more MUX gates that allows to disable the left handshake signals of the controller.

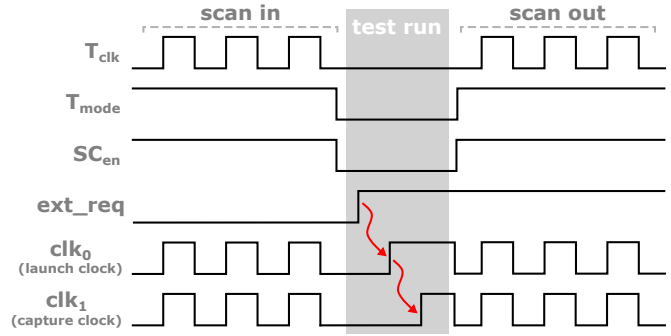

Fig. 4. Behavior of the testing signals during a test cycle.

Despite of the higher additional testing signals, note that the two first signals (T_{mode} and SC_{en}) are usually already presented into the design whether a scan-chain structure is used. Thus, the proposed structure only leverages the scan-chain structure for better controllability and observability of the data path. However, the proposed structure adds testing signals to the control part of the circuit: external request (ext_req) and handshake breaker (HSB_i) signals. As the name suggest, HSB_i signals are responsible to 'break' the left handshake communication of a given controller. Consequently, each controller in the circuit requires a reserved HSB_i signal. When enabled, the HSB_i sets the controller in **launch** mode during at-speed testing. In this mode, the left handshake signals of the controller are disconnected from the previous controller(s) and bypassed directly to ext_req. This isolates

the controller from its previous neighbour controller(s) and avoids any interference from another token inserted in previous controllers, which may create timing issues during at-speed testing. However, this also creates a testing limitation because no handshaking communication is possible and, consequently, makes it impossible to verify the timing constraint of the previous control/data path. This requires different HSB configurations in order to cover all existing timing constraint in the design. When HSB_i is disabled, the controller stays in **capture** mode and it will communicate normally with previous controller(s). To reduce the use of test input of each HSB_i signal, a shift register can be employed to load sequentially each HSB_i.

A. Test Cycle

The test cycle is composed of three main steps: (1) scan in, (2) test run and (3) scan out. Figure 4 illustrates the behavior of the main test signals. During the first step (1), both T_{mode} and SC_{en} signals are enabled. This will put the circuit in test mode and will allow loading the scan chain and configuring each HSB_i signal. Once the scan chain and the HSB shift register are loaded, T_{mode} and SC_{en} signals are disabled, the circuit goes back in normal mode. When ext_req is enabled, it launches all controllers in launch mode (HSB_i enabled) and propagates tokens to the remaining controllers set in capture mode (HSB_i disabled) – this is the test run step (2). Finally, the circuit is put in test mode again to scan out (3) the circuit and verify if the capturing registers contain the expected test vectors.

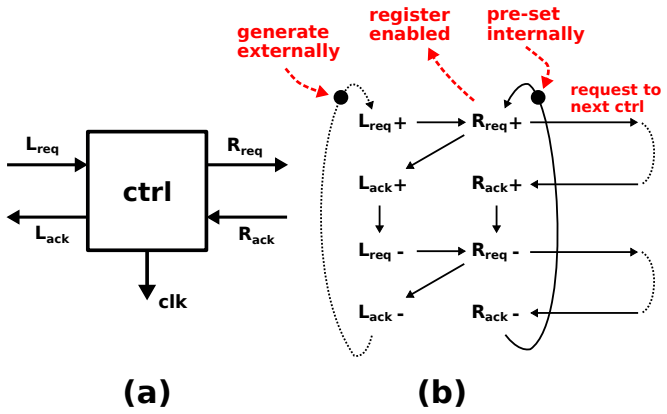

Fig. 5. Block representation (a) of a weak feedback half buffer (WCHB) controller with its handshake signals and STG representation (b) of the WCHB controller demonstrating the handshake behavior.

B. Initialization

For the proposed testing procedure, it is required to initialize both data and control paths.

For the data path, test vectors must be loaded into the design for two-pattern testing. Considering a launch register, which will be directly activated by an inserted token, a previous and successor data path is loaded with test vectors in such a way that, when the circuit is launched, the launch register will stimulate a target path while the token is also propagated through the control path. A straightforward technique to load the data path is a scan chain. To allow that, the controller clock clk_i is usually multiplexed with an external test clock T_{clk} and the registers are replaced by their equivalent scannable implementations. During test mode, the registers are externally controlled by T_{clk} and data paths are bypassed. During normal mode, registers are controlled internally by the their respective controllers and data paths are not longer bypassed.

Regarding the control path, all handshake signals must be set according the initial state of the employed handshake protocol. In that way, all the controllers must have known handshake signal values and be ready to process the inserted tokens during the test. This can be done by setting the primary control inputs and all the memories inside the control blocks – flip-flops, latches or C-element – according to the circuit initial state. On top of that, depending of the employed handshake protocol, multiple initial states must be considered to ensure that the circuit is fully tested. For the sake of illustration, Figure 5 illustrates block representation (a) of a weak-feedback half-buffer (WCHB) controller with its handshake signals and its State Transition Graph (STG) representation (b). The STG indicates the transition sequences of the left and right handshake signals of the WCHB controller. Note that we consider only a standard four-phrase handshake protocol. In this case, registers are only activated in the first phase of handshake protocol ($R_{req}+$ rising). As our technique focuses on externally generating the rising transition of the left request signal $L_{req}+$, all the remaining handshake signals and internal memories must have the logic values preceding $R_{req}+$. Taking into consideration the STG in Figure 5 (b), the left token is generated externally through ext_req. However, the right token requires that not only the controller itself would be properly reset, but also all its successor controllers – guaranteeing the desired logic value of R_{ack}.

C. Verifying Circuit Correctness

After launching the circuit, the circuit must be checked to verify circuit correctness. This can only be done after the circuit had computed all tokens and no token is being propagated through the circuit. Then, the values in the capture registers can be extracted through the scan path and matched with the expected values. If the the capture registers contains expected values, it indicates that the timing constraints between the control and data paths was respected. Otherwise, the timing constraint between the launch/capture registers have been violated.

D. Retrieving Path-Delay Information with Local Clock Sets

As mentioned before, the LCS flow [4] enables the use of standard EDA tools to run synthesis and STA on bundled-data circuits. The basic idea of the LCS flow consists on using root clocks at each controller in order to 'break' the combinational loops in the control logic. Based on a given root clock, LCS derives launch and capture generated clocks, which allow to

verify hold and setup timing constraints, respectively, beyond neighbour root clocks. Because of this timing support created by the LCS flow, it is possible to check timing violations with standard EDA commands (e.g. $report\_timing$). Note that, during STA checking, the tool is capable to identify the required transitions to stimulate a given path, such as a critical path or any desired path. This very same information is also the required information that ATPG tools need to know how registers must be initialized for path-delay testing.

It is important to indicate that, in our case, the path-delay extraction is software dependent, specially because it must employ the same tools compatible with the LCS flow. In that way, giving a more pratical view of our approach, the path-delay information can be extracted through Synopsys' Primetime tool – currently supported by the LCS flow. On top of that, Primetime interfaces Synopsys' ATPG tool TetraMAX, which reads the given path-delay information and generates test vector for two-pattern testing. However, the behavior of the local clock signals are not identical to synchronous two-pattern testing as the root clock and the generated capture clock pulse only once. Consequently, the only valid expected register values generated by TetraMAX is in the capture registers and all values in the launch registers must be ignored (*don't care*).

E. Testing Non-linear Structures

Asynchronous circuits employs complex flow control scheme far different from a conventional linear scheme as illustrated in Figure 1. Thus, they can employ unconditional and conditional flow schemes such as forks (a), joins (b), splits (c) and merges (d), as indicated in Figure 6. According to the employed flow scheme, the proposed testing technique must be adapted in order to properly insert tokens in the circuit without stalling the circuit or violating the handshaking protocol.

With a fork scheme, the generated token propagates to all successor branches. This implies that the controller in the sending end must be configured in launch mode and all branch controllers in capture mode. As the fork employs a unconditional control flow, a single token in the receiving controller propagates to all branches, allowing to verify the operation correctness of all branches in parallel. The same idea applies to join schemes. However, in this case, the branches controllers are the sending end. Then, all branch controllers must be configured in launch mode. If the controller in the receiving end does not receive tokens from all branch controllers, no further token is generated and the circuit halts, registering no new data in the receiving end.

For conditional control flow, such as splits and merges, additional care is needed. In split schemes, the split selector is responsible to redirect the flow to a target branch. In order to test all branches, it is required to generate tokens that will propagate to a target branch, which is selected by the split selector controller. Then, for a target branch, the sending and split selector controller must be set in launch mode and the target branch controller in capture mode. This process is repeated until all branches were selected and tested.

Again, the merge scheme considers a similar logic as the split scheme. For a target branch, the target branch and merge selector controller must be set in launch mode and the receiving controller in capture mode.

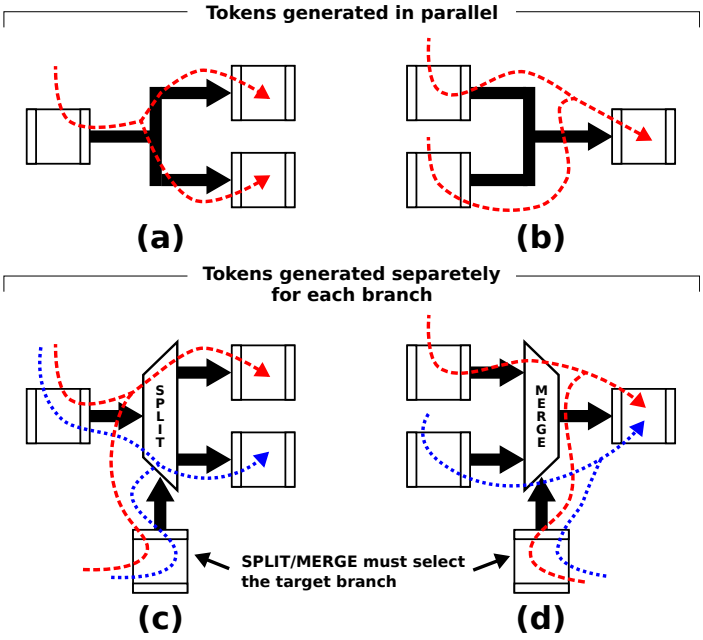

Fig. 6. Token generation according the control path structure: (a) fork, (b) join, (c) split and (d) merge. The arrows indicate where the token is generated (through ext_req) and where it is propagated.

F. Compatibility with Traditional Stuck-at Testing

Another point that the proposed architecture takes into consideration is to keep the compatibility with traditional testing, including the use of DfT and ATPG tools available in the industry. In this case, we consider scan insertion and traditional stuck-at testing. As the control part utilizes C-elements and combinational loops, the strategy focuses on isolate the control part. Taking as example Figure 7, this is done through enabling the T_{mode} signal, which bypasses all control logic and redirects all register control to T_{clk}. Consequently, the circuit operates synchronously and the scan test protocol remains the same.

V. Study-case Circuit

This section details the implementation of the proposed testing architecture in a study-case circuit. Moreover, it also includes stuck-at and at-speed testing.

To apply the proposed testing architecture, a micropipeline-based 128-bit AES core is considered. The AES core was designed and synthesized in-house, using a 65-nm CMOS technology from STMicroelectronics. Figure 8 (a) depicts the register stages of the AES core, where each stage is controlled by a separated root clock. The original core comprises two main blocks: control and data path blocks. The control block employs a four-phase handshaking protocol and is responsible to interact with the external handshaking interfaces and control the registers in the data path. The AES data path block implements a FF-based design and its execution is controlled

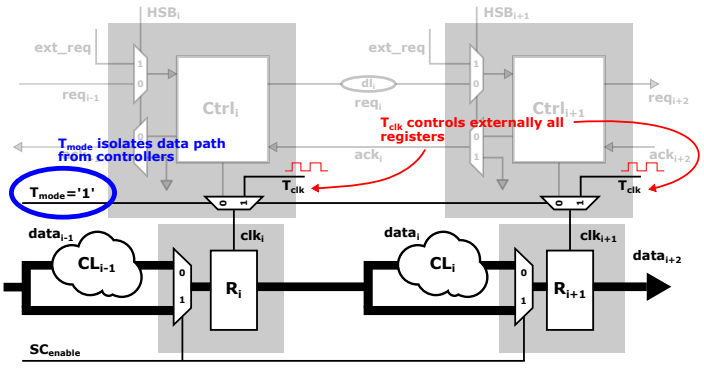

Fig. 7. Stuck-at testing with proposed approach. Data paths are isolated with T_{mode} signal, avoiding any interaction with controllers. Thus, circuit operates as a conventional synchronous circuit.

through 8 root clock signals (clk_{label}) and two one-hot control signals for the merge and split structure (sel_{merge} and sel_{split}), all generated by the control block.

TABLE I. AREA RESULTS OF THE ORIGINAL AND THE PROPOSED TESTABLE AES CORE.

Implementation	Original	Proposed
Combinational (um^2)	45280.80	54752.64
Buffers/Inverters (um^2)	8758.80	10154.64
Non-combinational (um^2)	11146.56	13138.80
Macro/black box (um^2)	86.16	86.16
Total (um^2)	56513.52	67977.60
Area Overhead (%)	-	20.28

The first main modification in the design is the employment of a testable version of the micropipeline controllers, illustrated in Figure 8 (b). The testable version adds the HSB logic (left MUX and AND gate) and a second multiplexer to implement the testing clock bypass. Moreover, a 8-bit shift register has been added into the design to enable the configuration of the HSB logic of each micropipeline controller. Each bit of the HSB shift register allows to access directly one of the eight root clocks. Note that the T_{clk} controls both AES control block and the HSB shift register. Thus, during scan manipulation, the HSB shift register can be loaded at the same time.

The synthesis step considers the Synopsys' Design Compiler tool with LCS flow support. Manually, all controllers are replaced by the proposed testable counterpart. The LCS flow has been modified to enable the DfT insertion through Synopsys' DFT Compiler. Taking a full-scan approach in this study case, the DFT Compiler replaces all registers for scannable FFs and only considers T_{clk}, SC_{en} and T_{mode} signals to control the scan path. The remaining testing signals (HSB_{in} and ext_req) are ignored. As the T_{mode} bypasses the control block when enabled, the DfT tool ignores the AES control block and HSB shift register during scan insertion. Table I compares the area results of the original and the proposed testable AES core. With

TABLE II. ATPG RESULT SUMMARY FOR STUCK-AT TESTING.

Detected	218323
Undetectable	17
ATPG Untestable	1550
Not Detected	0
Total Faults	219890
Test Coverage	99.30%
Scan Patterns	217

the addition of the testable controller, the 8-bit shift register and the scan path, the testable AES presents an area overhead of around 20%. This overhead comes mostly from the scan path, as the shift register and the AES control block only contribute to 0.3% of the area overhead. As the AES' control path represents 1.5% of the area consumption, it was expected that the extra circuitry in the control part would not inflict a significant impact.

As the control block is bypassed and the AES data path employs FF registers, it is possible to use a conventional ATPG tool, such as Synopsys' TetraMAX, to generate test patterns for traditional stuck-at testing. Table II gives a summary of the ATPG for the testable AES core. The ATPG achieves 99.30% of fault coverage considering the full-scan architecture considered in this study case. As the ATPG tool is not able to manage the asynchronous logic of the controllers, the tool is configure to not test the control path, which includes the handshake signals and the signals dedicated to at-speed testing (HSB_{in} and ext_req).

During at-speed testing, the proposed architecture verifies whether the delay lines between controllers match the critical paths between controllers – as previous detailed in Equation 1 – and also validates the controller operation.

The testing patterns were generated with an ad-hoc approach, targeting to stimulate the critical path of each pipeline stage. Here, the LCS flow provides essential information to assist with the pattern generation. As the LCS flow creates root clocks and generated launch/capture clocks to enable STA analysis between control and data paths, this same information is used to stimulate the desired critical path. In our case, all critical path information is obtained with Synopsys' PrimeTime, which is able to read all constraint files created by the LCS flow and indicate the required transitions to stimulate the critical path. For example, Listing 1 shows the PrimeTime's output regarding the critical path in the loop between MIX (clk_{stage3}) and ADD (clk_{stage1}) stages. Note that, however, the critical path information provided by PrimeTime contains only the stimuli at the data path – not the control part. Thus, the HSB configuration necessary to properly activate the correct launch/capture clocks was done manually according each case.

The testing patterns were loaded in the scan chain and the HSB shift register was configured to set which controller would operate in launch or capture mode. Figure 9 presents the two testing cycles performed in the study-case circuit,

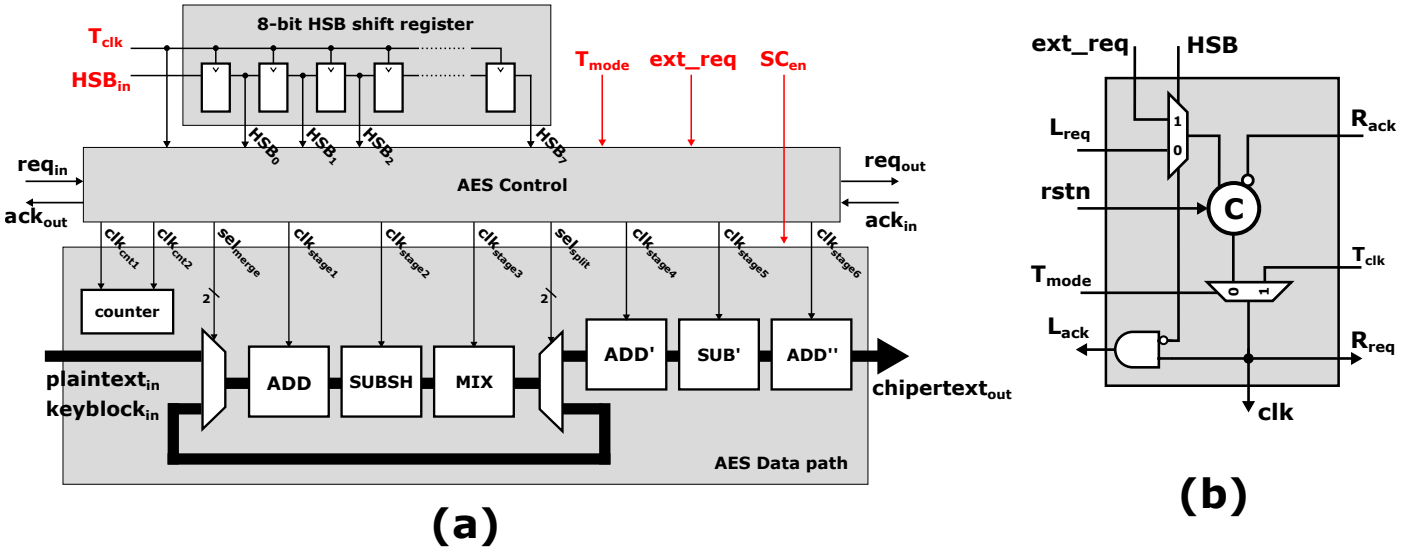

Fig. 8. Block diagram of the considered micropipeline-based AES core (a) and the testable micropipeline controller (b). Red arrows indicate the additional testing signals.

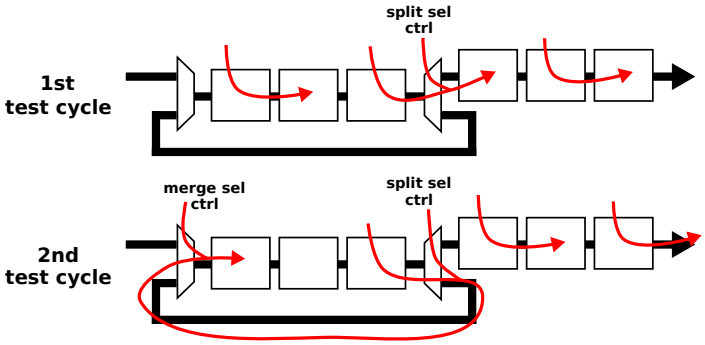

Fig. 9. HSB configuration for each test cycles during at-speed testing. Arrows represent where each token is inserted and the target path.

indicating where the test inserts tokens. The end of each arrow represents the last controller where the tokens was propagated and that their respective registers contains the resulting pattern to be checked. Moreover, as already discussed in subsection IV-E, the test of split and merge schemes requires that the selector controllers propagate the token to the target branch. The two cycles are required due to the fact that the HSB configuration disables the left handshake communication of the launch controllers. Thus, it is not possible to verify the timing constraints between the launch controllers and any previous controller.

Listing 1. Example of critical path between two root clocks of the AES circuit. This example considers the critical path in the loop between the third stage (MIX) and the first stage (ADD).

```
$path {
  // from: datapath/round/mix/ \
  //            colmix_reg/outrkey_reg_0__3__3_
  // to: datapath/round/add/ \
  //            addkey_reg/subst_d_reg_3__5_
```

```
  $name "aes_stage3-aes_stage1_setup_merge_c" ;
  $cycle 0.0 ;
  $slack 0.679212 ;
  $transition {
    "datapath/U355/D0" ^ ; // (HS65_LH_MUX21X27)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U286/A" ^ ; //(BFX27)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U34/A" ^ ;  //(NAND2X43)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U275/B" v ; //(NOR2X25)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U127/A" ^ ; //(IVX22)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U14/A" v ;  //(NOR2X3)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U47/B" ^ ;  //(NOR2X13)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U119/C" v ; //(OAI211X3)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U18/A" ^ ;  //(CBI4I1X3)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U62/D" v ;  //(CB4I1X18)
    "datapath/round/add/addkey_comb_in/ \
        keysched1_comb_in/sub0_comb/U147/D" v ; //(OAI211X5)
    "datapath/round/add/addkey_reg/ \
        subst_d_reg_3__5_/D" ^ ; //(SDFPRQX4)
    }
}
```

VI. Conclusions

This paper presents an at-speed DfT architecture for bundled-data circuits, applied to micropipeline-based designs. By modifying the micropipeline controller, it is possible to verify whether the forward timing constraints between the control and data paths had been respected. In addition to that, the architecture still enables traditional stuck-at testing, allowing the use of DfT and ATPG tools already available in the market. With a 128-bit AES core as study-case circuit, the modifications in the micropipeline controllers and the addition of the HSB shift register only contributes to a total area increase of 0.3%.

The architecture also leverages from the LCS flow to see exactly the stimuli required to activate the critical paths. This allowed us to properly load the scan chain and launch two-pattern testing to verify the timing constraints. However, it is important to highlight that the entire test setup is not yet fully automated and it will be addressed in the future. Another current limitation of the proposed architecture is that, after inserting tokens through the ext_req signal, it is not possible to evaluate whether the circuit had finished the test run. In nominal operation, the end of the test could be estimated according an expected worst-case delay, albeit this estimation is not trivial in a voltage scaling scenario, for example. Consequently, the architecture could employ any structure responsible to acknowledge the end of the test run. As a last point, the proposed DfT architecture can be extended to allow at-speed testing for any bundled-data template, covering templates such as Click, MOUSETRAP or GasP.

REFERENCES

[1] J. Kwong and A. P. Chandrakasan, "Variation-driven device sizing for minimum energy sub-threshold circuits," in *ISLPED'06 Proceedings of the International Symposium on Low Power Electronics and Design*, Oct 2006, pp. 8–13.

[2] R. D. Jorgenson, L. Sorensen, D. Leet, M. S. Hagedorn, D. R. Lamb, T. H. Friddell, and W. P. Snapp, "Ultralow-power operation in subthreshold regimes applying clockless logic," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 299–314, Feb 2010.

[3] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design – A Systems Perspective*. Springer, 2001.

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

[5] R. M. Davies and J. V. Woods, "Timing verification for asynchronous design," in *Proceedings of European Design Automation Conference (EURO-DAC)*, 1996, pp. 78–83.

[6] S. Pagey, S. D. Sherlekar, and G. Venkatesh, "Issues in fault modelling and testing of micropipelines," in *Proceedings First Asian Test Symposium (ATS '92)*, Nov 1992, pp. 107–111.

[7] A. Khoche and E. Brunvand, "Testing micropipelines," in *Proceedings of 1994 IEEE Symposium on Advanced Research in Asynchronous Circuits and Systems*, Nov 1994, pp. 239–246.

[8] ——, "A partial scan methodology for testing self-timed circuits," in *Proceedings 13th IEEE VLSI Test Symposium*, April 1995, pp. 283–289.

[9] O. A. Petlin and S. B. Furber, "Scan testing of asynchronous sequential circuits," in *Proceedings. Fifth Great Lakes Symposium on VLSI*, March 1995, pp. 224–229.

[10] V. Schober and T. Kiel, "An asynchronous scan path concept for micropipelines using the bundled data convention," in *Proceedings International Test Conference 1996. Test and Design Validity*, Oct 1996, pp. 225–231.

[11] O. A. Petlin and S. B. Furber, "Built-in self-testing of micropipelines," in *Proceedings Third International Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 1997, pp. 22–29.

[12] M. Roncken and E. Bruls, "Test quality of asynchronous circuits: a defect-oriented evaluation," in *Proceedings International Test Conference 1996. Test and Design Validity*, Oct 1996, pp. 205–214.

[13] M. Roncken, E. Aarts, and W. Verhaegh, "Optimal scan for pipelined testing: an asynchronous foundation," in *Proceedings International Test Conference 1996. Test and Design Validity*, Oct 1996, pp. 215–224.

[14] M. Roncken, "Defect-oriented testability for asynchronous ICs," *Proceedings of the IEEE*, vol. 87, no. 2, pp. 363–375, Feb 1999.

[15] M. Roncken, S. M. Gilla, H. Park, N. Jamadagni, C. Cowan, and I. Sutherland, "Naturalized communication and testing," in *2015 21st IEEE International Symposium on Asynchronous Circuits and Systems*, May 2015, pp. 77–84.

[16] M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Saldanha, and A. Taubin, "Partial-scan delay fault testing of asynchronous circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 11, pp. 1184–1199, Nov 1998.

[17] Yong-Seok Kang, Kyung-Hoi Huh, and Sungho Kang, "New scan design of asynchronous sequential circuits," in *AP-ASIC'99. First IEEE Asia Pacific Conference on ASICs (Cat. No.99EX360)*, Aug 1999, pp. 355–358.

[18] K. van Berkel, A. Peeters, and F. te Beest, "Adding synchronous and LSSD modes to asynchronous circuits," in *Proceedings Eighth International Symposium on Asynchronous Circuits and Systems*, April 2002, pp. 161–170.

[19] F. T. Beest, A. Peeters, M. Verra, K. van Berkel, and H. Kerkhoff, "Automatic scan insertion and test generation for asynchronous circuits," in *Proceedings. International Test Conference*, Oct 2002, pp. 804–813.

[20] F. Beest, A. Peeters, K. Berkel, and H. Kerkhoff, "Synchronous full-scan for asynchronous handshake circuits," *J. Electronic Testing*, vol. 19, pp. 397–406, 08 2003.

[21] F. Beest and A. Peeters, "A multiplexer based test method for self-timed circuits," in *11th IEEE International Symposium on Asynchronous Circuits and Systems*, March 2005, pp. 166–175.

[22] M. L. King and K. K. Saluja, "Testing micropipelined asynchronous circuits," in *2004 International Conference on Test*, Oct 2004, pp. 329–338.

[23] M. Singh and S. M. Nowick, "MOUSETRAP: High-speed transition-signaling asynchronous pipelines," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 6, pp. 684–698, June 2007.

[24] Feng Shi, Yiorgos Makris, S. M. Nowick, and M. Singh, "Test generation for ultra-high-speed asynchronous pipelines," in *IEEE International Conference on Test, 2005.*, Nov 2005, pp. 10 pp.–1018.

[25] D. Shang, A. Yakovlev, F. Burns, F. Xia, and A. Bystrov, "Low-cost online testing of asynchronous handshakes," in *Eleventh IEEE European Test Symposium (ETS'06)*, May 2006, pp. 225–232.

[26] V. Varshavsky, M. Kishinevsky, V. Marakhovsky, V. Peschansky, L. Rosenblum, A. Taubin, and B. Tzirlin, *Self-timed control of concurrent processes*, 1990.

[27] G. Gill, A. Agiwal, M. Singh, Feng Shi, and Y. Makris, "Low-overhead testing of delay faults in high-speed asynchronous pipelines," in *12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'06)*, March 2006, pp. 11 pp.–56.

[28] I. Sutherland and S. Fairbanks, "GasP: a minimal FIFO control," in *Proceedings Seventh International Symposium on Asynchronous Circuits and Systems. ASYNC 2001*, March 2001, pp. 46–53.

[29] F. A. Kuentzer and A. M. Amory, "Fault classification of the error detection logic in the Blade resilient template," in *2016 22nd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, May 2016, pp. 37–42.

[30] F. A. Kuentzer, L. R. Juracy, and A. M. Amory, "On the reuse of timing resilient architecture for testing path delay faults in critical paths," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 379–384.