Journal of Electronic Testing (2022) 38:321–334
https://doi.org/10.1007/s10836-022-06007-w

Built-In Self-Test for Multi-Threshold NULL Convention Logic Asynchronous Circuits using Pipeline Stage Parallelism

Brett Sparkman[1] · Scott C. Smith[2] · Jia Di[3]

Received: 3 March 2022 / Accepted: 4 June 2022 / Published online: 17 June 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Although several synthesis methods for asynchronous circuits exist, only limited test methodologies have been developed. This paper presents a built-in self-test (BIST) architecture for Multi-Threshold NULL Convention Logic (MTNCL) asynchronous circuits that utilizes an automated, industry-standard tool-based flow. The software procedure for, and hardware components to implement, BIST functionality are explained. To improve testing performance, the MTNCL pipeline is separated into multiple parallel BIST circuits, with standard pipeline components doubling as BIST circuitry to reduce area overhead. Results of this BIST architecture and software performance is explored for three different test cases looking at area impact and effects of varying the number of input patterns and initial seeds. Further refinements to fault exclusions based upon operating principles of MTNCL are developed to better depict actual fault coverage; and additional hardware modifications are proposed to improve controllability and observability to further increase fault coverage.

Index Terms Asynchronous logic · Built-in self-test (BIST) · Multi-threshold NULL convention logic (MTNCL) · NULL convention logic (NCL) · Sleep convention logic (SCL)

1 Introduction

While synchronous circuits have been the dominant architecture in digital systems for decades, asynchronous designs exhibit several advantages that are becoming more enticing as fabrication process technology continues to shrink. Several of the primary advantages involve the lack of a global clock, resulting in reduced power, noise, and electromagnetic interference, and robustness to PVT (process, voltage, temperature) variations. However, there are several barriers to adoption of asynchronous design styles, including lack of designer familiarity with asynchronous architectures, synthesis methods to generate asynchronous circuits from register-transfer level (RTL) hardware description language (HDL) code, and testing methods to validate functionality of the resulting asynchronous circuit.

Two promising asynchronous circuit paradigms are NULL Convention Logic (NCL) [1] and Multi-Threshold NULL Convention Logic (MTNCL), also known as Sleep Convention Logic (SCL) [2]. Both NCL and MTNCL are supported by the UNCLE synthesis tool [3]; and Design-For-Test (DFT) methods have been developed for both [4, 5]. Built-In Self-Test (BIST) methods have been developed for NCL [6], and an initial approach proposed for MTNCL that tests the entire MTNCL circuit as a single BIST stage [7].

This paper expands upon the authors' previous work in [7] to develop a parallelized BIST architecture for MTNCL circuits to decrease testing time, along with an automated flow to insert the BIST functionality and obtain and validate desired fault coverage. The developed automated flow utilizes standard synchronous test software when possible, to minimize custom software, while also providing the user with a sense of familiarity. Additionally, further refinements

Responsible Editor: V. D. Agrawal

✉ Scott C. Smith
scott.smith@tamuk.edu

Brett Sparkman
bsparkma@uark.edu

Jia Di
jdi@uark.edu

[1] Electrical Engineering, University of Arkansas, Fayetteville, AR, USA

[2] Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville, TX, USA

[3] Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR, USA

are implemented to yield higher fault coverage by increasing controllability and observability, and excluding some reported faults based on the operating principles of MTNCL circuits.

2 Background

NCL is a quasi-delay insensitive (QDI) asynchronous design paradigm that utilizes multi-rail signals, such as dual-rail logic, to represent data, and utilizes completion detection and a 4-phase return-to-0 handshaking protocol for control [1]. In dual-rail logic, 2 wires, D^1 and D^0, referred to as rails, represent 1 bit of data; D^1 asserted represents logic 1, D^0 asserted represents logic 0, both rails de-asserted represents the NULL state (i.e., absence of DATA), and both rails simultaneously asserted is illegal. NCL circuits are designed using threshold gates with hysteresis, and circuits must be input-complete and observable for delay-insensitivity [8]. MTNCL is similar to NCL, but instead of flowing a NULL wavefront through the circuit to reset all gates to 0 for the NULL state, MTNCL threshold gates include a *sleep* input, connected to a handshaking control signal, which simultaneously forces gates to 0 for the NULL state. Hence, MTNCL gates do not require hysteresis, and MTNCL circuits do not require input-completeness or observability, resulting in faster, smaller, lower power circuits compared to NCL [2].

NCL and MTNCL systems consist of NCL/MTNCL registers, combinational logic (C/L), and completion detection, which can be grouped into stages that include 1 of each, as shown in Fig. 2 for an MTNCL pipeline. Each MTNCL register has dual-rail inputs and outputs, and a Boolean *sleep* input. MTNCL C/L implements a desired function by taking dual-rail logic inputs and producing dual-rail logic outputs, and is comprised of MTNCL gates, described above. MTNCL utilizes an early completion handshaking protocol, which allows DATA to flow through a stage after all inputs become DATA and the subsequent stage is requesting DATA, and sleeps the stage to NULL after all stage inputs are NULL and the subsequent stage is requesting NULL [2].

An example of an MTNCL slept early completion component for 8 dual-rail inputs is shown in Fig. 1, which is comprised of MTNCL TH12 gates to detect a DATA or NULL for each of the 8 register input signals, and a tree of MTNCL THnn gates to combine the multiple TH12 gate outputs into a single signal, which is then combined with the subsequent stage's early completion component's sleep output via a resettable inverted NCL TH22 gate, to generate the current stage's early completion component's sleep output.

Integrated circuits (ICs) require testing to detect faults that can occur during the fabrication process, such as a wire being shorted to ground (i.e., stuck-at-0) or shorted to V_{dd} (i.e., stuck-at-1), to ensure correct operation. There are 2 main testing methodologies, 1) utilizing external instrumentation to input test patterns to the device under test (DUT) and analyze the resulting outputs, or 2) incorporating additional logic within the DUT during its design, such that the DUT can internally generate its own test patterns and also validate the resulting outputs internally, without requiring external test hardware, which is referred to as BIST [9].

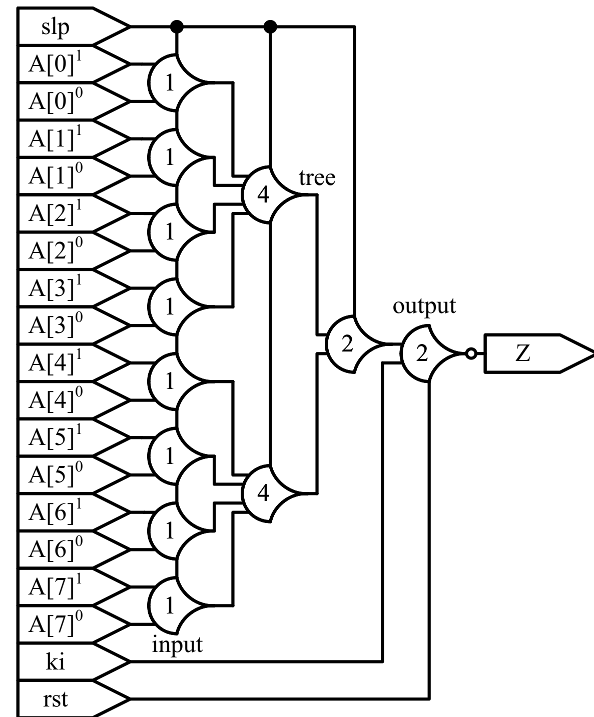

Fig. 1 MTNCL Slept Early Completion Component

3 MTNCL BIST Design with Pipeline Stage Parallelism

3.1 Components

In [7], a BIST implementation was designed that enables simple functional BIST by inserting the BIST circuitry as a wrapper around the entire MTNCL circuit. The approach presented herein improves upon the wrapper-based BIST design to increase observability and controllability of faults inside the MTNCL pipeline, by partitioning it so that each pipeline stage can be tested in parallel. This may yield shortened test times as the cycle time of the BIST design is reduced to single pipeline stages instead of the full design.

To implement MTNCL BIST, traditional synchronous BIST methods were adjusted for compatibility with MTNCL asynchronous systems. A common BIST method utilizes linear feedback shift registers (LFSRs), which are circular shift

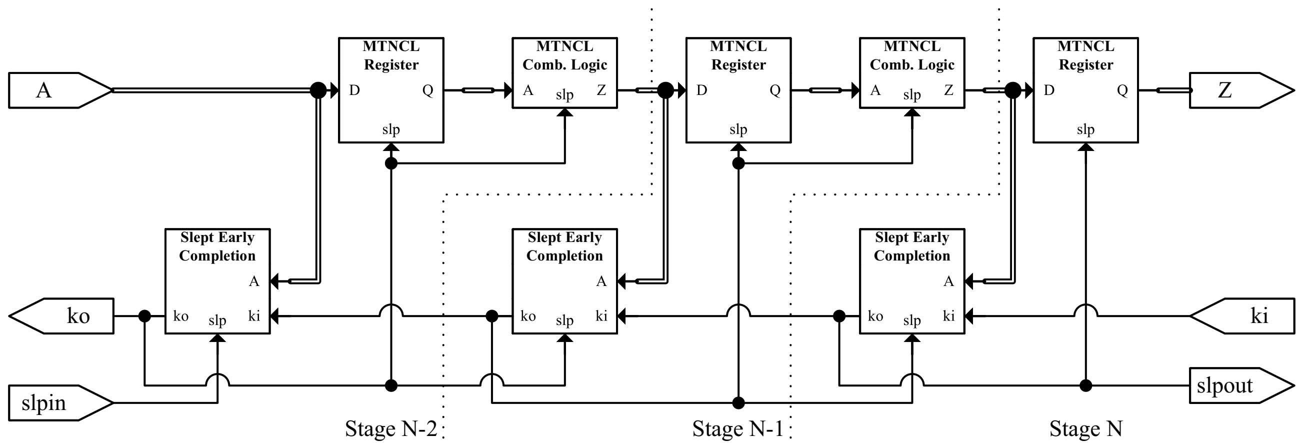

Fig. 2 MTNCL Pipeline Architecture [2]

registers with XOR elements in various feedback paths, to generate input patterns. An N-bit LFSR is reset to a non-zero seed, and then produces 2^{N}-1 pseudorandom outputs. As LFSRs are a simple and effective way to present a large number of input patterns, while requiring minimal additional logic, LFSRs were utilized to generate the BIST inputs at each of the pipeline stages. Since DFFs typically have both a Q and Q' output, with slightly different timing delays for these signals, dual-rail gating (DRG) components, shown in Fig. 3, were implemented to allow for proper flow of DATA and NULL wavefronts, by presenting a NULL wavefront when its *D/N'* signal is 0 and a DATA wavefront when its *D/N'* signal is 1. The output response of each pipeline stage of the DUT was measured with a Multiple Input Shift Register (MISR) by connecting both the D^0 and D^1 rails of the circuit to inputs of the MISR to enable checking both rails simultaneously. Multiplexers were used to control the flow of data between standard operation and BIST mode. Simple Boolean logical equivalence checkers, shown in Fig. 4, were utilized to control the number of input patterns presented to the BIST stages by gating off the LFSR clock once the final input pattern was presented, and then validating that the final MISR output was the expected value, meaning that the circuit is functioning correctly.

3.2 Software Procedure

An automated method was implemented to import an MTNCL DUT Verilog netlist, separate the design into multiple BIST stages, automatically insert the required MTNCL BIST logic, simulate digital functionality, evaluate fault coverage, and iterate, by first increasing the number of test patterns and then varying initial values for the LFSRs, until either the desired fault coverage is achieved or the maximum possible fault coverage is obtained with limits imposed upon the number of test patterns and seed adjustments. A flowchart outlining this high-level procedure is shown in Fig. 5. This automation tool was designed using Python for netlist parsing, separating the design into multiple BIST stages, implementing all BIST component netlists and testbenches, creating simulation macros, running both digital and fault simulations, evaluating simulation results, and iterating to improve fault coverage. Mentor Graphics ModelSim and Synopsys TetraMAX were utilized for digital simulation and fault simulation, respectively, as these are industry-standard software packages. Compatibility with other simulators may be possible with modifications.

A pipeline is separated into multiple BIST stages, as shown in Fig. 6. The first BIST stage consists of the input MTNCL register, C/L, and the first and subsequent registers' slept early completion components. Intermediate BIST stages include the stage's input MTNCL register, C/L, and the subsequent register's slept early completion component. The final BIST stage includes the last two MTNCL registers, C/L, and final slept early completion component. This was adjusted from the standard MTNCL pipeline shown in Fig. 1.

For each simulation action, several simulations were performed. To create the various equivalence hardware, the

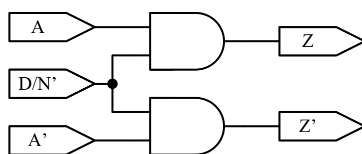

Fig. 3 Dual-Rail Gating (DRG) Component

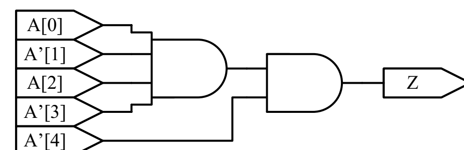

Fig. 4 Example Equivalence Component for an Input Pattern of 00,101

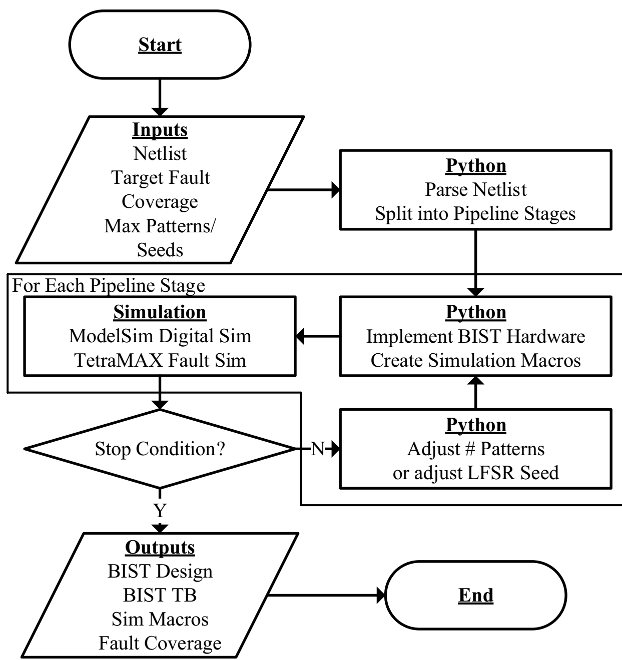

Fig. 5 Flowchart of High-Level MTNCL BIST Procedure with Parallelism

LFSRs' and MISRs' outputs of a golden simulation must be run for the desired number of patterns for each BIST stage. As digital simulation is an efficient manner of calculating these values, a base BIST block and testbench is initially created for this purpose, similar to Fig. 7 but excluding the two dashed blocks containing equivalence hardware and C/L. If the BIST stage currently being simulated is the final BIST stage, it will utilize the dashed completion tree component instead of the inverter; this completion tree includes the inputs and tree portions of the slept early completion component shown in Fig. 1. For all other BIST stages, the dashed single inverter is added instead of the completion tree component so that the stage's slept early completion component can be re-used; the inversion is required because the slept early completion component has an inverted TH22 gate at its output, as shown in Fig. 2.

A mechanism to detect when a valid DATA wavefront has arrived at each BIST stage output is required to clock the various MISRs for each output pattern. Thus, the internal MTNCL pipeline's slept early completion components are utilized to detect when the BIST stage output has transitioned to valid DATA, and then clock the MISR. An added completion tree component is utilized for the final BIST stage to ensure that DATA is stable when the MISR is clocked. Although the MISR outputs are not connected to any logic, they are monitored during simulation.

Since TetraMAX is a cyclic fault simulator, it is incapable of properly handling the asynchronous DATA and NULL wavefronts unless both the changing inputs and outputs occur in the same cycle; additionally, the DATA wavefront must be provided last so that the simulator uses the proper output value. To enable this, the base BIST block restricts both the input and output to transition only once during a cycle, through the use of the added TH22 threshold gate (i.e., in dashed box in Fig. 7) that conjoins the BIST block's completion tree component's inverted output with the *ko* output from the BIST stage, and feeds this back into the BIST stage's *ki* input. This ensures that the BIST block will not request a NULL wavefront until after a valid DATA wavefront has appeared at the output, propagating from the static DATA input. To further constrain the system with a static DATA input per cycle, the DRG *D/N'* control signal is taken from this *ki* as well, and then inverted to clock the LFSR. If multiple DATA transitions occur in a single cycle, as would be the case if inputs were provided as soon as requested, the TetraMAX functional simulator may produce invalid results compared to the golden simulation, resulting in an invalid fault grading result. The BIST stage's *ki* is utilized as a rising strobe during TetraMAX fault simulation each cycle.

Once the LFSR and MISR pattern are known from a digital simulation of the BIST block, the equivalence modules are created when writing the testbench and BIST module for

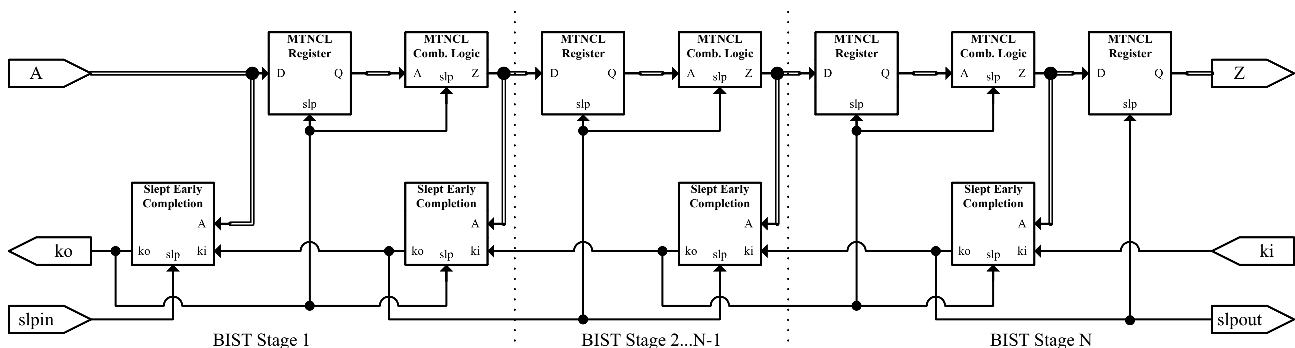

Fig. 6 MTNCL BIST Stage Pipeline Separation

Fig. 7 MTNCL BIST Block Architecture with Parallelism

fault simulation, and inserted into the architecture as shown in Fig. 7. The LFSR equivalence circuit is produced to match the input pattern following the desired final input. The LFSR and MISR are internally clocked using the BIST stage's handshaking signals. Once the final pattern is reached, the LFSR clock is gated off, and the DRG *D/N'* control signal remains low, providing a static NULL wavefront to the BIST stage input.

Since the fault simulation BIST architecture will be utilized by TetraMAX, it includes the constrained acknowledge signals, merged with the dashed TH22 gate, and utilizes the BIST stage's *ki* for the gating mechanism of the LFSR and DRG control signals. This ensures that only one DATA input is presented to the BIST stage during fault simulation every cycle, as required for the fault simulation.

Once the digital simulation is completed for the fault simulation BIST architecture, then a fault simulation is performed for stuck-at faults. If the fault simulation completes correctly, then all fault summary information is parsed and utilized to calculate fault coverage. This includes detected faults (DT), possibly detected faults (PT), undetectable faults (UD), ATPG untestable faults (AU), not detected faults (ND), and total faults. DT faults include faults that TetraMAX was able to completely evaluate as being detected with the present set of inputs. PT faults are evaluated when the good digital simulation values are known, but the faulty machine simulation resulted in an unknown value. UD faults are ones that cannot be tested, and may be due to unused outputs or pins that are statically tied to 0 or 1, may have controllability or observability limitations, or may have redundant logic that would mask an actual fault. AU faults are faults that cannot be controlled or observed due to constraints utilized during fault simulation, or faults regarding non-scan sequential devices. As these constraints and devices that would incur this type of fault are not utilized with the proposed flow or MTNCL, TetraMAX did not report any AU faults for any of the circuits reported in this paper. ND faults are ones that were not detected during fault simulation and may occur due to lack of controllability or observability. Total faults represent the total number of simulated faults.

Fault coverage is then calculated by dividing the sum of all DT and half-weighted sum of all PT from each BIST stage by the sum of the total faults of each BIST stage, as shown in Eq. (1). This value is not output from TetraMAX, although it does output the individual fault coverage of each BIST stage separately. This calculation is the same method that TetraMAX uses for a single design. As PT faults have a 50% chance of being detected in a binary system, they are assigned a weight of 0.5, which is the default TetraMAX value. None of the other fault types are utilized in this calculation, although they are recorded in the simulation log.

$$Fault\ Coverage = \frac{\sum DT + 0.5 \times \sum PT}{\sum Total\ Faults} \quad (1)$$

Once the fault simulation is complete, the final MTNCL BIST architecture is generated by removing the TH22 gate used to constrain the DATA/NULL cycles during the fault simulation, such that BIST stage input and output MTNCL registers once again function independently, as originally designed. A schematic of the final MTNCL BIST

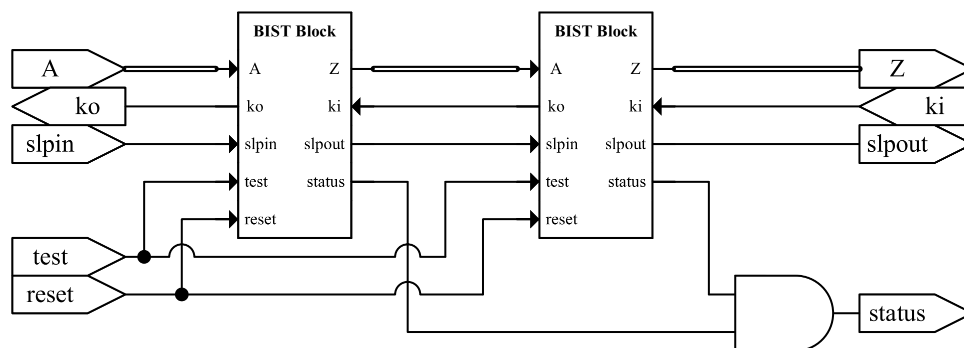

Fig. 8 Example 2-Stage Top-Level MTNCL BIST Design with Parallelism

architecture is shown in Fig. 7, excluding the dashed TH22 gate and using its inverted input as its following MUX's *B* input.

The interface for the MTNCL BIST final circuit is the same as the original MTNCL circuit, shown in Fig. 1, with the addition of a *test* input and *status* output. *Test* controls when the circuit functions in BIST mode vs. standard operating mode; and *status* indicates if the DUT generated the correct MISR output after a self-test, to show whether or not it is functioning correctly. Since this BIST implementation utilizes parallelism by separating pipeline stages of the MTNCL DUT, each pipeline stage is included as its own BIST block in the top-level design, then the status of each is merged using an AND tree to output the final self-test status of the overall DUT, as shown in Fig. 8. As each BIST block controls itself using its own BIST circuitry, the controllability and observability of the system is increased because each of the BIST stages' data inputs and outputs are independently provided by an LFSR and measured using an MISR, respectively.

4 MTNCL BIST with Parallelism Results

4.1 Design Preparation

A number of circuits were used to evaluate the developed automation flow, including many ISCAS '85 C/L benchmarks, which are available as structural Verilog netlists [10]. UNCLE [3] was utilized to synthesize MTNCL circuits from synchronous RTL. To integrate the linear synchronous pipeline functionality required by UNCLE into these purely C/L circuits, an input register was added before the C/L, two registers connected in series were added following the output of the C/L, and a *clk* input was added to control all three registers. Note that UNCLE cannot currently synthesize an MTNCL circuit from a synchronous circuit that includes feedback, which is why the ISCAS'89 sequential benchmarks were not used. Following UNCLE synthesis, the resulting pipelines were balanced using Synopsys Design Vision. Additionally, an 8-bit adder and a 32-bit multiplier were also evaluated. These two designs were implemented as 2-stage pipelines, using behavioral RTL, and synthesized into MTNCL circuits using UNCLE.

Three test cases were performed for each design. The initial test case had a target fault coverage of 75%, no maximum pattern count (the default), and no additional seeds besides the initial default binary 1 seed. The number of input patterns was initialized to 5. This would enable the circuit to utilize the maximum number of patterns for each BIST stage, 2^n-2 patterns, where n is the number of input bits to the BIST stage, provided the target fault coverage was not reached first. This run was primarily performed to ensure that the fault simulation would complete in a short amount of time.

For the second test case, a target fault coverage of 100% was utilized. The number of patterns was initialized to 5, while the maximum number of input patterns was set to 1E6 patterns. For any BIST stage with fewer than 20 input bits, this would utilize all possible input patterns, with some patterns repeated. For a BIST stage with 20 input bits or more, only 1E6 patterns would be tested. Similar to the previous test case, the number of seeds was limited to 1, so that only the initial default binary 1 seed was used. This test case was utilized to determine the maximal fault coverage obtainable by each design within 1E6 patterns and using only the initial seed. This was used to evaluate performance as the number of input patterns increased. The actual simulation time increases significantly as the design size and pattern count increases; a maximum pattern count of 1E6 was set to limit the actual time required for the larger designs.

The third test case had a target fault coverage of 100%, a pattern count set to exactly 1000 patterns (i.e., 1000 starting pattern count and 1000 maximum pattern count), and a maximum seed count set to 1000 seeds. The initial seed was the default seed pattern of binary 1, with additional seeds of the same-length as the LFSR hardware randomly generated. The 1000 static pattern count was selected after reviewing results from the other test cases, as most designs converged to a moderately high fault coverage within 1000 patterns. Although this does not run an exhaustive simulation for all possible LFSR seeds, it does provide insight into how fault

coverage, test time, and actual time vary across different seeds; however, an untested seed could potentially produce a better fault coverage.

4.2 First Test Case: Area Impact

Since all the first test case simulations successfully completed quickly, those fault results are not discussed. As the first test case had no maximum pattern count, only LFSR lengths up to the BIST stage input length were generated, which yields the minimum sized BIST hardware required. An increased pattern count could produce a slightly larger design, as the LFSR and equivalence circuits would increase in size. A comparison of circuit area for the original and first test case BIST designs, as measured by Synopsys Design Vision using UNCLE's library of gate areas, is listed in Table 1.

For small designs, a large area impact was observed, greater than 200%. This occurs because the ratio of BIST stage I/O to logic is rather high, such that the added BIST logic is more significant compared to the original area. For medium-sized designs, an area impact of approximately 65–132% was generally observed. The c499 design had a significant area overhead due to a higher ratio of BIST stage I/O to original circuit logic. For large designs, area overheads of 37.4% and 27.1% were realized. Overall, area overhead is dependent on the ratio of BIST stage I/O to original circuit logic. This parallel implementation requires additional area overhead compared to the previous work [7], since it requires BIST logic for each pipeline stage, instead of only at the circuit's primary I/O.

4.3 Second Test Case: Maximum Fault Coverage

The results from the second test case with a target of 100% fault coverage and a maximum pattern count limit of 1E6 patterns are shown in Table 2. None of the designs were able to obtain the target fault coverage of 100% within 1E6 patterns; so the maximum fault coverage obtained for any of the 1E6 patterns is shown for each design. In many cases, the maximum fault coverage was obtained for numerous consecutive iterations as more patterns were presented, so the minimum pattern count producing the maximum fault coverage is shown along with the test time and actual time for that specific run.

Table 1 MTNCL BIST Area Comparison

Design Name	Original Area	Area with BIST	% Overhead
c17	153	543	254.9%
adder8	621	1901	206.1%
c499	2200	6873	212.4%
c432	2403	5584	132.4%
c1908	4471	7985	78.6%
c880	4502	9079	101.7%
c1355	5632	9278	64.7%
mult32×32	21,523	29,575	37.4%
c6288	23,731	30,167	27.1%

Table 2 MTNCL BIST Maximum Fault Coverage (1E6 Patterns, 1 Seed)

Design Name	Fault Coverage (%)	Patterns	Test Time (ns)	Actual Time (s)
c17	76.752	160	4649	8
adder8	79.539	163,840	5,542,128	224
c499	84.664	2560	107,529	30
c432	83.237	2560	128,009	21
c1908	81.612	81,920	4,014,089	1090
c880	82.855	5120	235,529	99
c1355	83.848	5120	245,769	99
mult32×32	95.330	655,360	54,583,533	1,233,222
c6288	86.509	5120	471,049	1544

All designs obtained a fault coverage of at least 76.75%, which was unexpectedly lower than the previous work [7]. This decreased fault coverage occurred due to additional nodes being evaluated, such as the outputs from the added BIST multiplexers. Although this may seem low, only output nodes of each BIST stage are being observed; and internal circuitry, such as completion components, do not directly affect data outputs and are therefore difficult for ATPG tools to assess. For all but one design, the maximum fault coverage took less than 25 min to simulate for that specific number of input patterns, which is listed as Actual Time in Table 2. Note that this is not the summation of all simulations, which would be more. The largest design in terms of instance count (but not circuit area), mult32×32, took approximately 342.6 h to run the digital and fault simulations for 655,360 patterns. The only remaining pattern count for this design, 1E6 patterns, took 564 h to run, while achieving the exact same fault coverage. Thus, large designs could become a burden in terms of both CPU resources and run time for high pattern counts. Test Time, which is the time required to perform a complete BIST of the DUT, as approximated by the digital simulator using UNCLE's MTNCL gate models, may also be significant to the user. The mult32×32 design would require approximately 55 ms to complete a BIST operation. This is reduced from 88 ms in the previous work [7], so the parallel MTNCL BIST architecture presented herein increases testing throughput. Depending upon the cost associated with testing, the tradeoff of a lower fault coverage may be enticing to reduce testing time.

Table 3 MTNCL BIST Relaxed Fault Coverage (1E6 Patterns, 1 Seed)

Design Name	Fault Coverage (%)	Patterns	Test Time (ns)	Actual Time (s)
c17	76.519	20	589	9
adder8	78.636	40	1332	8
c499	83.416	640	26,889	15
c432	83.142	320	16,009	13
c1908	80.879	1280	62,729	28
c880	80.584	320	14,729	18
c1355	81.807	640	30,729	22
mult32×32	94.673	320	26,019	1051
c6288	84.123	640	58,889	223

Table 4 LFSR Seed Effect on MTNCL BIST Fault Coverage (1000 Patterns, 1000 Seeds)

Design Name	Fault Coverage (%)	Seed Count	Test Time (ns)	Actual Time (s)
c17	80.023	277	29,009	15
adder8	81.670	339	33,881	16
c499	84.814	49	42,009	24
c432	83.637	768	50,009	20
c1908	82.223	520	49,009	28
c880	84.478	630	46,009	32
c1355	84.812	807	48,009	31
mult32×32	95.380	453	83,285	2065
c6288	88.784	688	92,009	268

Since none of the designs achieved 100% fault coverage, and the maximum fault coverage generally took a moderate actual time to obtain, the lowest fault coverage within 2.5% of the maximum fault coverage was determined, and the circuits were re-analyzed at this fault coverage. These results are shown in Table 3, and demonstrate that although the maximum fault coverage may take a long time to obtain, it may be possible to obtain an acceptable fault coverage within a much shorter timespan by slightly relaxing required fault coverage. In some cases, the fault coverage penalty was much less than 2.5%.

As the fault coverage began to increase for designs with a very large pattern count, substantial savings can be observed for both test time and actual time. For the mult32×32 design, the actual time difference to obtain 94.67% fault coverage instead of the 95.33% maximum fault coverage was approximately 342 h; the slightly lower fault coverage only took 17.5 min at that specific pattern count. Likewise, the BIST test time was also significantly faster, approximately 26 µs vs. 55 ms.

4.4 Third Test Case: LFSR Seed Adjustment

The results from the third test case with a target of 100% fault coverage and a maximum seed count of 1000 seeds are shown in Table 4. For these simulations, all designs used a starting and maximum pattern count of 1000, so exactly 1000 input patterns were presented for all of the various seeds. This may have resulted in slightly larger BIST structures for small-input-count circuits. Like the second test case, none of the designs were able to obtain the target 100% fault coverage. The maximum fault coverage obtained is listed for each design.

For most of the designs, the test and actual times were very similar across all seeds. All the designs were able to achieve a higher fault coverage using different seeds than the initial seed used in the second test case. For all other designs except c17, the 1000 patterns also had a significantly reduced pattern count compared to the maximum fault coverage, and thus smaller test time compared to the second test case, while achieving this higher fault coverage. These effects are design dependent, as is the rate of increasing fault coverage as pattern counts increase across various seeds. The actual times are almost doubled due to additional simulations and file I/O, but test times are significantly reduced compared to the previous work [7]. Note that actual times could be improved via parallel simulation, since all BIST stages function as completely independent circuits.

4.5 MTNCL BIST Automation with Parallelism Performance

To further investigate the performance of the MTNCL BIST automation, all designs were extensively analyzed in terms of their actual simulation time across the second and third test cases. The information for the c1355 design is detailed, as it was the largest of the medium-sized designs.

Fault coverage versus pattern count is shown in Fig. 9 on the primary axis. The implementation achieved an initial fault coverage around 36% using 5 patterns. As pattern count increases, fault coverage also increases, and then flattens out once the maximum is obtained. Actual time vs. pattern count is shown in Fig. 9 on the secondary axis. This time was calculated from the log files by comparing the times between the fault coverage output of each run. For lower pattern counts with this design, the digital simulations took a longer time to complete than the fault simulations. However, the fault simulations began to take more time starting just before 1000 patterns, and increased at a faster rate. The percentages of total run time for the fault simulations, digital simulations, and processing accounted for 91.37%, 8.32%, and 0.31%, respectively.

During analysis of the third test case, histograms showing the distributions are utilized to observe differences using a static 1000 pattern count, as the starting LFSR seed is adjusted for 1000 different seeds. The fault coverage distributions for the c1355 design are shown in Fig. 10, which has a normal distribution, with an average fault coverage of 83.6%. The

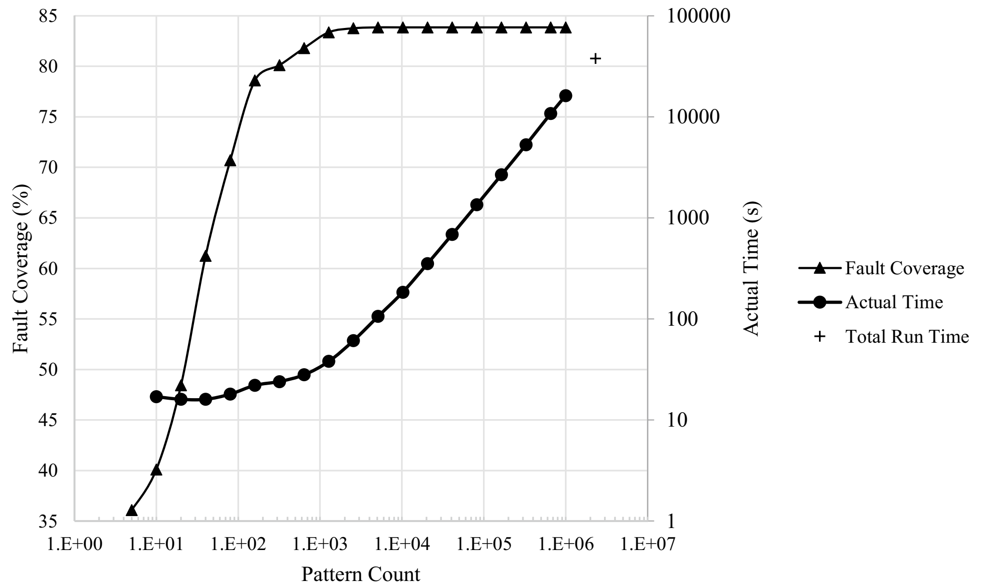

Fig. 9 Fault Coverage and Actual Time vs. Pattern Count for c1355 Design

maximum fault coverage obtained was 84.8%, 1% higher than in the second test case, and requiring only 20% of the pattern count. The average actual time was 9.85 s; and the percentages of total run time for the fault simulation, digital simulation, and processing accounted for 49.18%, 30.64%, and 20.18%, respectively.

5 Additional Considerations

5.1 Feedback Compatibility

Although UNCLE is incapable of synthesizing designs that include feedback, the pipeline stage parallelism method presented herein can be utilized for designs with data feedback, such as Finite State Machines (FSMs) and datapath feedback; however, some manual adjustments are required. One approach is to avoid breaking feedback loops by assigning an entire feedback circuit to a single pipeline stage, while other pipeline stages can still be parallelized. However, as discussed in [7], an MTNCL BIST circuit with data feedback is operational, but due to TetraMAX limitations, fault coverage cannot be calculated. Hence, a better approach is to partition each feedback loop into multiple BIST stages, such that no BIST stage contains data feedback. This enables TetraMAX to calculate fault coverage for each BIST stage, which are then combined to produce the DUT overall fault coverage for the DUT, as explained in Sect. 3.2.

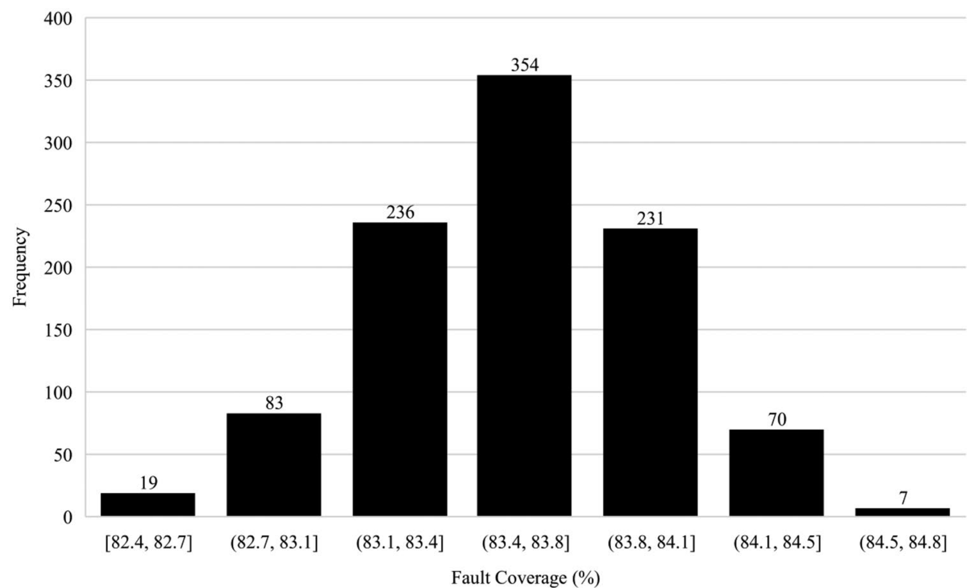

Fig. 10 Fault Coverage Distribution for c1355 Design with 1000 Seeds

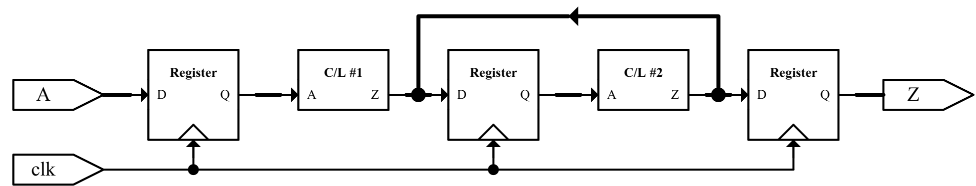

Fig. 11 Synchronous Pipeline with Datapath Feedback

Take for example the synchronous circuit shown in Fig. 11, which includes internal datapath feedback, and its MTNCL implementation, shown in Fig. 12, which includes the minimum required 3 registers in the feedback path [8]. Note that C/L #2 in Fig. 11 is partitioned into C/L #2A and #2B in Fig. 12 to increase throughput. Also note that any registers that provide feedback must be implemented as Reset to DATA (RTD) NCL registers to provide a valid DATA wavefront for the feedback path upon initialization; all non-feedback registers remain as MTNCL registers. The dual-rail feedback path from the feedback loop's output (e.g., primary output, *Z*, in Fig. 12) into the feedback loop's MTNCL register feedback input (e.g., input *D* of top most MTNCL register in Fig. 12) must also be incorporated into the previous stage's completion component to detect when a valid DATA wavefront has arrived at the input to the entire feedback loop stage. Similarly, as shown in Fig. 12, a TH22 gate must be added to merge the *ko* generated from the stage immediately prior to the feedback loop with the *ko* generated from the next stage after the feedback loop, to ensure that the feedback loop's NCL register holds a valid DATA wavefront until the DATA wavefront has been processed by both the feedback loop and the subsequent stage.

As shown in Fig. 12, this design could be partitioned into 2 BIST stages, with the entire feedback loop implemented as a single BIST stage. It is important to note that additional multiplexers and BIST input generation logic are needed to fully decouple the feedback path when in test mode. One dual-rail bus multiplexer must be inserted at the feedback datapath completion input port, labelled as *A* in Fig. 12, along with additional BIST input logic, in order to provide complete DATA/NULL wavefronts to the completion logic when the stages are separated in test mode. A single multiplexer is required at the merging TH22 gate's output, labelled as *B* in Fig. 12, to break the feedback handshaking during test mode so the BIST stages operate independently. However, as mentioned previously, this would be a functional system, but would not allow for simulation or calculation of fault coverage due to TetraMAX limitations with feedback loops.

As shown in Fig. 13, the feedback path could instead be broken and partitioned into multiple BIST stages, thereby allowing BIST circuitry to provide inputs to all BIST stages during each individual stage's TetraMAX fault simulation, such that fault coverage for the DUT can be calculated. Same as Fig. 12, the addition of multiplexers at *A* and *B*, and additional BIST input logic at *A*, is necessary to enable both testing capability and normal design functionality.

It is worth noting that TetraMAX simulation of settable NCL gates does not exhibit proper hysteresis behavior, as the output of a RTD gate with one data input asserted, is de-asserted immediately after its *reset* input is de-asserted, whereas the gate should remain asserted due to hysteresis. To remedy this, the external RTD NCL registers in feedback paths must be extracted up one level of hierarchy from the BIST stage to the BIST block level, shown in Fig. 7. Correct hysteresis functionality will then be exhibited via the Value

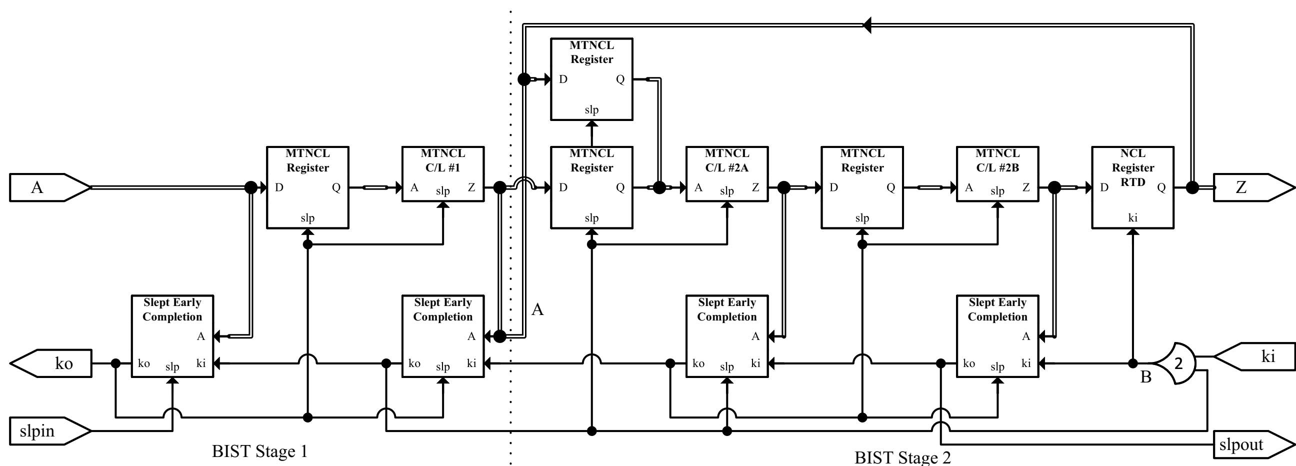

Fig. 12 MTNCL Pipeline with Datapath Feedback as a Single BIST Stage

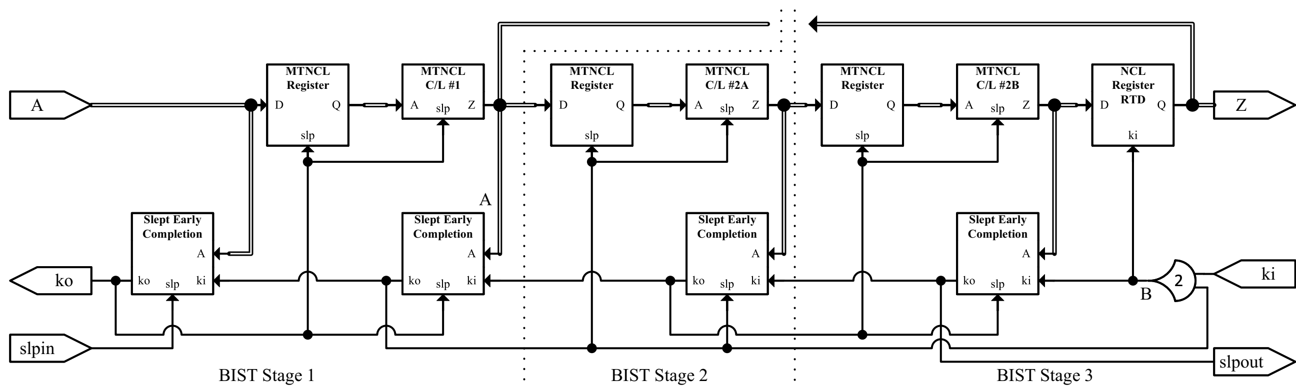

Fig. 13 MTNCL Pipeline with BIST Datapath Feedback Partitioning

Change Dump (VCD) files through digital simulation, so TetraMAX simulation will be possible. However, faults on this now external RTD NCL register are no longer included.

5.2 Improving Fault Coverage via Fault Exclusion

As MTNCL circuits operate in an asynchronous fashion with local handshaking, specific combinations of stuck-at fault type and asynchronous gate function can be applied to the TetraMAX verbose fault list to better depict actual fault coverage. For example, if one of the BIST stage sleep nets is stuck-at-1, then that entire stage will always be slept, such that it will never transition to DATA. Likewise, if one of the BIST stage sleep nets is stuck-at-0, that would cause the previous stage's sleep net to be stuck-at-1 (i.e., the previous stage's slept early completion component final TH22 NCL gate, shown in Fig. 2, would be stuck-at-0 due to its *ki* input, which is the stuck-at-0 sleep net; and this TH22 NCL gate output is inverted to generate the previous stage sleep net, which would therefore be stuck-at-1). Both of these examples would cause the circuit to deadlock. Furthermore, any slept early completion component gate output (except for the final inverter, which is already considered in the previous case) that is stuck-at-0 will cause the sleep net generated by that component to be stuck-at-1, which would cause the circuit to deadlock, as mentioned above. Since these scenarios cause the circuit to immediately deadlock, any undetected faults on any of these nets flagged by TetraMAX can be ignored, since they would be immediately detected, in either test mode or normal operation, due to circuit deadlock. A summary of exclusion rules is provided below; and applying these exclusion rules to the single BIST stage c17 circuit [7] increases fault coverage from 86.93% to 90.74%.

1. Stuck-at faults on sleep nets can be excluded.
2. Stuck-at-0 faults on slept early completion component gate outputs can be excluded.

Through review of verbose fault lists produced by TetraMAX for several designs, it was determined that many undetectable faults were located in the slept early completion logic. When slept early completion components are designed using MTNCL threshold gates (with sleep input), stuck-at-1 faults can be masked by the sleep mechanism, and therefore cannot be excluded. However, if NCL gates (with hysteresis) are used to implement the early completion logic instead of MTNCL gates (i.e., replace the MTNCL gates in Fig. 2 with NCL gates), then any stuck-at-1 fault in this logic will result in a stuck-at-0 fault on its corresponding subsequent sleep net, and can therefore be excluded. A summary of these rules is provided below; and applying these additional exclusion rules to the single BIST stage c17 circuit [7] further increases fault coverage from 90.74% to 98.10%. The tradeoff for using this method to increase fault coverage is a slight decrease in performance and slight increase in area, energy/operation, and leakage power, as NCL gates are larger, with increased leakage power and energy per transition, compared to their MTNCL equivalent, and this requires a NULL input to flow through the non-slept early completion logic instead of all gates being simultaneously slept to 0 [2].

3. Stuck-at-1 faults on early completion component gate outputs can be excluded, when designed using NCL gates.
4. Stuck-at-1 faults on any input to an early completion component can be excluded, when designed using NCL gates.

5.3 Improving Fault Coverage via Increased Controllability

Since reported fault coverages are slightly lower than current industry-standard requirements, the ability to add controllability and observability points in asynchronous dual-rail logic was investigated. Controllability points can be added to dual-rail nets to inject a DATA value in order to improve fault coverage, using the hardware shown in Fig. 14. When

Ctrl Sel is asserted, the *Ctrl* D^0 and *Ctrl* D^1 inputs replace D^0 and D^1 generated by the preceding C/L, respectively, allowing for injection of a DATA0 (D^0 asserted, D^1 de-asserted), a DATA1 (D^0 de-asserted, D^1 asserted), or even an INVALID (both D^0 and D^1 asserted) value, as desired. Note that an INVALID value should only be injected if not part of a single stage BIST feedback loop; otherwise, this could result in perpetual INVALID values in the feedback loop until the circuit is reset. Also, note that a NULL value should not be injected, as this may cause the pipeline to deadlock. Furthermore, each dual-rail signal transitions to NULL after every DATA value, so there would be no need to inject a NULL value. Note that C/L could also be utilized instead of two multiplexers; however, multiplexers offer a higher level of control, with increased area as the tradeoff.

5.4 Improving Fault Coverage via Increased Observability

It may also be desirable to increase observability separately, or in addition to increased controllability. To view any signal, the pipeline may be stalled so that a DATA wavefront exists on all dual-rail nets. This is the same methodology used to enable the asynchronous fault simulation; the additional dashed TH22 gate in Fig. 7 that ties the BIST stage *ko*, and *slpout* or added completion tree component, together forces pipeline stalls. Any net, such as a single rail of a dual-rail signal or an acknowledge net, as shown in Fig. 15, can be probed for improved observability when the pipeline is stalled, which requires one additional MISR bit for each net probed. Note that inserted observability points bypass the final BIST stage's added completion tree component, shown in a dashed box in Fig. 7. Adding controllability hardware and observability points into the single BIST stage c17 circuit [7] increases fault coverage from 86.93% to 91.17%, and requires 6 additional multiplexers and 5 additional MISR bits. Applying this after applying the previous fault exclusions further increased fault coverage from 98.10% to 98.54%.

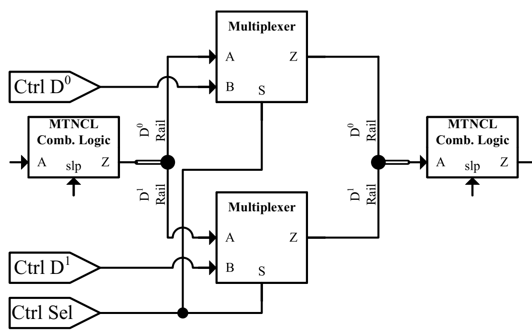

Fig. 14 MTNCL Controllability Hardware

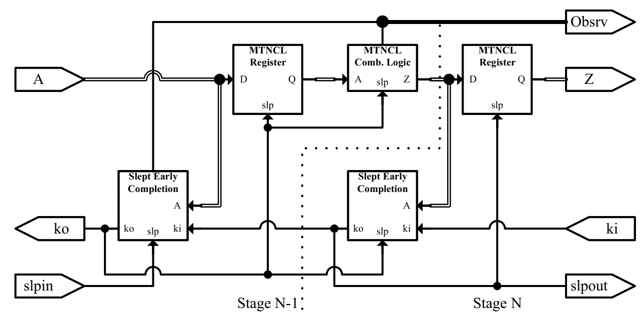

Fig. 15 Added Observability Points for Stage N-1

6 Conclusion and Future Work

This paper presents an automated method for MTNCL BIST insertion to achieve a desired fault coverage, which utilizes industry-standard tools within the design flow for digital and fault simulations. The proposed method separates the MTNCL DUT into multiple parallel BIST circuits, which yields a reduction in test time at the expense of increased area overhead, compared to the previous work in [7] that implemented the entire DUT as a single BIST stage. The method presented herein also allows for fault calculation of circuits that include data feedback, whereas the previous work in [7] produced functional BIST circuits for designs with data feedback, but could not calculate their fault coverage. Furthermore, fault exclusion rules were developed, based upon the operating principles of MTNCL circuits, to better depict actual fault coverage. And, additional controllability and observability hardware was proposed to further improve fault coverage for MTNCL circuits.

Compared to the single BIST circuit implementation in [7], a substantial area increase (average of 44% for the 9 test circuits) is required for the pipelined BIST due to additional LFSR and MISR circuitry for each pipeline stage, as shown in Table 5. The pipelined BIST implementation reduced test time for maximum fault coverage by an average of 51% for 6 of the 9 test circuits, compared to the single BIST implementation, while increasing test time for the other 3 test circuits, as shown in Table 6. However, fault coverage was slightly lower compared to the single BIST circuit implementation due to additional inserted nodes being evaluated for fault coverage, such as the outputs from added BIST multiplexers. Nevertheless, the parallel BIST implementation will significantly benefit from the addition of controllability and observability points, as each pipeline stage has access to all internal nodes during normal BIST operation, which can significantly increase overall fault coverage to above 98.5%, as detailed in Sect. 5.4. The single BIST implementation may also benefit from these, but the pipeline must be stalled to ensure a DATA wavefront is present to the entire pipeline, which would significantly increase test time.

Table 5 MTNCL BIST Area Comparison between Single and Pipelined BIST

Design Name	Original Area	Area with Single BIST	Area with Pipelined BIST
c17	153	355	543
adder8	621	1245	1901
c499	2200	3891	6873
c432	2403	3317	5584
c1908	4471	5822	7985
c880	4502	6370	9079
c1355	5632	7323	9278
mult32×32	21,523	24,513	29,575
c6288	23,731	25,260	30,167

The current automated flow increases pattern count in increasingly larger steps (i.e., doubled each iteration) to achieve the desired or highest attainable fault coverage, which could result in substantially more patterns than the minimum required. Hence, once the desired or maximum fault coverage is obtained, the associated pattern count, *N*, could potentially be further iterated upon to lower the minimum required pattern count to somewhere between *N/2* and *N*, while maintaining target/maximum attainable fault coverage. This would increase actual time, but potentially decrease test time and reduce area overhead. Additionally, other methods of pattern generation, instead of the LFSRs used in this work, could also be considered, such as BILBO registers [11]. Furthermore, the processes of fault exclusion and addition of controllability and observability points presented in Sects. 5.2 – 5.4, respectively, could be automated to make it feasible to apply these techniques to large circuits to significantly increase fault coverage, as shown for the small example c17 circuit where these techniques were manually applied to increase fault coverage to more than 98.5%, as described in Sect. 5.4.

Table 6 MTNCL BIST Maximum Fault Coverage Comparison between Single and Pipelined BIST (1E6 Patterns, 1 Seed)

Design	Single BIST		Pipelined BIST	
Name	Fault Coverage (%)	Test Time (ns)	Fault Coverage (%)	Test Time (ns)
c17	86.929	1.6E3	76.752	4.6E3
adder8	87.956	8.0E6	79.539	5.5E6
c499	86.783	327.7E3	84.664	107.5E3
c432	87.492	209.9E3	83.237	128.0E3
c1908	86.552	1.6E6	81.612	4.0E6
c880	86.097	3.1E6	82.855	235.5E3
c1355	86.920	384.0E3	83.848	245.8E3
mult32×32	93.396	87.7E6	95.330	54.6E6
c6288	90.424	26.9E3	86.509	471.0E3

Data Availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Competing Interests The authors have no relevant financial or non-financial interests to disclose.

List of Differences The work presented in this paper is a continuation of our previous work, titled "Built-In Self-Test for Multi-Threshold NULL Convention Logic Asynchronous Circuits," which was published in the *2020 IEEE VLSI Test Symposium* [7]. Our previous work implemented the entire MTNCL DUT as a single BIST stage, while this paper partitions the DUT, such that each MTNCL pipeline stage is its own BIST stage, in order to parallelize BIST operation. The method proposed herein not only decreases test time, but also allows for fault calculation of MTNCL circuits with data feedback, which was not possible in our previous work. Additional new contributions include development of fault exclusion rules, based upon MTNCL circuit operating principles, to better depict actual fault coverage, and hardware for increasing controllability and observability, in order to increase fault coverage. We estimate that this paper consists of approximately 50% new material compared to our VTS paper, which is well above the minimum requirement of 30% new material.

References

1. Fant KM, Brandt SA (1996) NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis. Proc Int Conf Appl Specific Syst Architect Process 261–273
2. Zhou L, Parameswaran R, Parsan F, Smith SC, Di J (2015) Multi-Threshold NULL Convention Logic (MTNCL): An Ultra-Low Power Asynchronous Circuit Design Methodology. J Low Power Electron Appl 5(2):81–100
3. Reese RB, Smith SC, Thornton MA (2012) Uncle - An RTL Approach to Asynchronous Design. Proc IEEE Int Symp Asynchronous Circuits Syst 65–72
4. Parsan F, Smith SC, Al-Assadi WK (2016) Design for Testability of Sleep Convention Logic. IEEE Transactions on VLSI Systems 24(2):743–753
5. Nemati N, Beckett P, Reed MC, Fant K (2018) Clock-less DFT-less Test Strategy for Null Convention Logic. IEEE Trans Emerg Top Comput 6(4):460–473
6. Nemati N, Reed MC, Fant K, Beckett P (2016) Asynchronous Interleaved Scan Architecture for On-line Built-in Self-test of Null Convention Logic. Proc IEEE Int Symp Circuits Syst 746–749
7. Sparkman B, Smith SC, Di J (2020) Built-In Self-Test for Multi-Threshold NULL Convention Logic Asynchronous Circuits. Proc IEEE VLSI Test Symp 1–6
8. Smith SC, Di J (2009) Designing Asynchronous Circuits using NULL Convention Logic (NCL). Synthesis Lectures on Digital Circuits and Systems, Morgan & Claypool Publishers, Vol. 4/1
9. Nagle HT, Roy SC, Hawkins CF, McNamer MG, Fritzemeier RR (1989) Design for Testability and Built-In Self Test: A Review. IEEE Trans Industr Electron 36(2):129–140

10. Fišer P. Collection of Digital Design Benchmarks. Czech Technical University in Prague, [Online]. Available: https://ddd.fit.cvut.cz/www/prj/Benchmarks/. Accessed June 2022
11. Wang LT, McCluskey EJ (1987) Built-in self-test for sequential machines. Proc Int Test Conf 334–341

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Brett E. Sparkman received B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Arkansas, in 2011, 2017, and 2020, respectively. From 2009 to 2015, he transitioned from an Engineering Intern to a Senior Engineer at Arkansas Power Electronics International, working on extreme-environment IC design and system implementations. Upon APEI's acquisition by Cree/Wolfspeed in 2015, he continued work as a System Design Engineer at Wolfspeed in Fayetteville, AR, USA. His current research interests include asynchronous circuits, embedded systems, wireless sensor networks, wide-bandgap power electronics, and automation.

Scott C. Smith received B.S. degrees in electrical engineering and computer engineering and the M.S. degree in electrical engineering from the University of Missouri, Columbia, in 1996 and 1998, respectively, and the Ph.D. degree in computer engineering from the University of Central Florida, Orlando, in 2001. He is a Professor of Electrical Engineering and Computer Science at Texas A&M University-Kingsville. He has published over 100 refereed journal/conference papers, 8 U.S. patents, 2 books, and 4 additional book chapters. His research interests include Asynchronous Logic, NULL Convention Logic, Computer Architecture, Embedded Systems, Digital Logic, FPGAs, CAD Tools for Digital Design, Computer Arithmetic, VHDL, VLSI, Secure/Trustable Hardware, Wireless Sensor Networks, Robotics, and Cyber Physical Systems. Dr. Smith is a senior member of IEEE, and a member of the National Academy of Inventors, Sigma Xi, IEEE-HKN, and Tau Beta Pi.

Jia Di received the B.S. and M.S. degrees from Tsinghua University, China, and the Ph.D. degree in electrical engineering from the University of Central Florida, in 1997, 2000, and 2004, respectively. He joined the Computer Science and Computer Engineering Department of the University of Arkansas as an Assistant Professor in Fall 2004, where he is now a Professor, Rodger S. Kline Chair, and the Department Head. His research area is asynchronous integrated circuit design and hardware security. He has published two books and more than 100 papers in technical journals and conferences. He also has 5 U.S. patents. He is a senior member of the IEEE and an elected member of the National Academy of Inventors.