IMPLEMENTATION OF ADDER AND SUBTRACTORS USING SLEEP CONVENTION LOGIC

Poloju Samatha[1] , D Santhosh Kumar[2]
[1]P.G Student, Department of Electronics and Communication Engineering
[2]Assistant professor, Avn institute of engineering and technology

Abstract-Testability is a major concern in industry for today's complex system-on-chip design. Design-for-testability (DFT) techniques are essential for any logic style, including asynchronous logic styles in order to reduce the test cost. Sleep convention logic (SCL) is a new promising asynchronous logic style that is based on the more well-known asynchronous logic style NULL convention logic (NCL). In contrast to the NCL, there are currently no designs for testability methodologies existing for the SCL. The aim of this paper is to analyze the various faults within SCL pipelines and propose a scan-based DFT methodology to make the SCL testable. The proposed DFT methodology is then validated through a number of experiments, showing that the methodology provides a high test coverage (>99%). The complete DFT methodologies as well as the scan chain and scan cell design are presented.

Keywords*-Multi threshold NULL convention logic (MTNCL);NULL convention logic (NCL); sleep convention logic (SCL).*

I. INTRODUCTION

In recent decades, power consumption has become a major consideration in integrated circuit design. In high speed systems, clock switching could use a large portion of power. Additionally, leakage power has come to dominate power consumption as process sizes shrink. Adaptive beam forming circuits have many applications where lower power is highly desirable without sacrificing performance. These systems often require GHz range of throughput to accommodate the fast input data stream, while having long idle periods between sets of activities. In order to reduce power, asynchronous design methods have become increasingly attractive over the past two decades. Quasi-delay-insensitive (QDI) asynchronous circuits, such as NULL Convention Logic (NCL) do not use clock; instead, they incorporate handshaking protocols to control the circuit's behavior [1]. By removing the need for clock, switching power can be reduced and power consumption will be more evenly distributed across the chip. The Multi-Threshold NCL (MTNCL) design paradigm incorporates the Multi- threshold CMOS (MTCMOS) power gating mechanism inside every logic gate in order to reduce power even further [2]. This paper presents a fine- grain time delay (FTD) unit and a coarse-grain time delay (CTD) unit for use in an adaptive beam former designed using the MTNCL paradigm for the DARPA Arrays at Commercial Timescale (ACT) program.

II. RELATED WORK

SCL is a self-timed quasi-delay insensitive (QDI) [10] asynchronous logic style based on the NCL. SCL was originally developed in [11]. SCL combines the idea of the NCL with early completion [12] and fine-grained MTCMOS power-gating [8]. During normal operation, each pipeline stage alternates between set and reset phases. In the set phase, data change from a spacer (called NULL) to a proper codeword (called DATA), and in the reset phase it changes back to NULL. SCL uses delay-insensitive encoded data [13] for data communication. The most popular delay-insensitive encoding is dual rail, but other encodings, such as quad rail or in general any mutually exclusive assertion groups can be used. A dual-rail encoded signal D consists of two wires, D0 and D1. D is logic 1 (DATA1) when D1 = 1 and D0 = 0, is logic 0 (DATA0) when D0 = 1 and D1 = 0, and is NULL when both D0 and D1 are 0. The SCL framework is shown in Fig. 1. Similar to the NCL, each pipeline stage contains a combinational logic function block (Fi), a register block (Ri), and a completion detector block (C Di). SCL requires an extra gate to synchronize between DATA and NULL phases. This extra gate is a simple resettable C-element [14] with inverted output, which will be called the completion C-element (Ci) hereafter. Combinational logic blocks in the SCL are made of threshold gates [15], [16] and implement unite functions where no logic inversions are allowed. An SCL gate is generally denoted as THmnWw1,...,wn where n is the number of inputs, m is the threshold of the gate, and w1, w2,...,wn are the weights of inputs when the weights are > 1. Assuming that the inputs of the SCL gate are x1,..., xn, the output of the SCL gate is asserted when $x_1w_1 + \cdots + x_nw_n \geq m$. For example, a TH23 gate consists of three inputs and its threshold is 2. In terms of Boolean logic, the output of the TH23 gate can be described as Z = AB + AC + BC, where A, B, and C are its inputs. Fig. 2 shows the transistor-level design of the SCL TH23 gate. SCL utilizes fine-grained power-gating by incorporating a sleep signal, S, in every single gate. Similar to the NCL gates [17], [18], each SCL gate is made of a set

block and SCL is a self-timed quasi-delay insensitive (QDI) [10] asynchronous logic style based on the NCL. SCL was originally developed in [11]. SCL combines the idea of the NCL with early completion [12] and fine-grained MTCMOS power-gating [8]. During normal operation, each pipeline stage alternates between set and reset phases. In the set phase, data change from a spacer (called NULL) to a proper codeword (called DATA), and in the reset phase it changes back to NULL. SCL uses delay-insensitive encoded data [13] for data communication. The most popular delay-insensitive encoding is dual rail, but other encodings, such as quad rail or in general any mutually exclusive assertion groups can be used. A dual-rail encoded signal D consists of two wires, D0 and D1. D is logic 1 (DATA1) when D1 = 1 and D0 = 0, is logic 0 (DATA0) when D0 = 1 and D1 = 0, and is NULL when both D0 and D1 are 0. The SCL framework is shown in Fig. 1. Similar to the NCL, each pipeline stage contains a combinational logic function block (Fi), a register block (Ri), and a completion detector block (C Di). SCL requires an extra gate to synchronize between DATA and NULL phases. This extra gate is a simple resettable C-element [14] with inverted output, which will be called the completion C-element (Ci) hereafter. Combinational logic blocks in the SCL are made of threshold gates [15], [16] and implement unite functions where no logic inversions are allowed. An SCL gate is generally denoted as THmnWw1,...,wn where n is the number of inputs, m is the threshold of the gate, and w1, w2,...,wn are the weights of inputs when the weights are > 1. Assuming that the inputs of the SCL gate are x1,..., xn, the output of the SCL gate is asserted when x1w1 +⋯+ xnwn ≥ m. For example, a TH23 gate consists of three inputs and its threshold is 2. In terms of Boolean logic, the output of the TH23 gate can be described as Z = AB + AC + BC, where A, B, and C are its inputs.

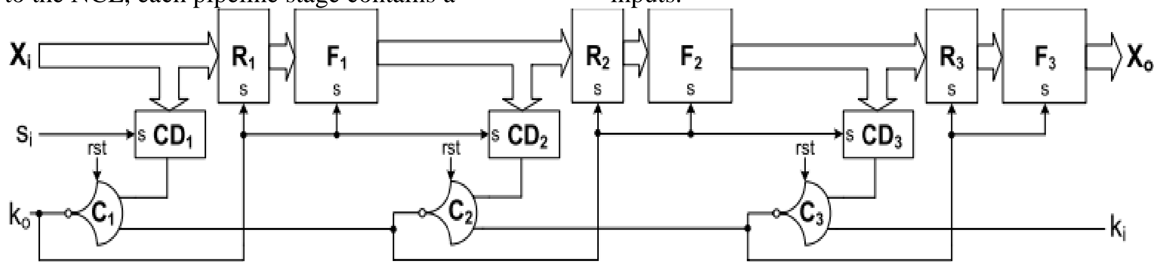

Fig. 1:SCL framework.

III. PROPOSED DESIGN FOR TESTABILITY METHODOLOGY

As discussed before, each stage of the SCL pipeline is made of four separate blocks: combinational logic function (Fi), completion detector (CDi), register (Ri), and completion C-element (Ci). Since the stuck-at faults in each block can impact the SCL pipeline in different ways, each block should be analyzed separately.

A sleep signal that forks to a combinational logic block can be either stuck-at-0 or stuck-at-1. When it is stuck-at-1, the combinational logic will be in sleep mode at all times and no DATA set can then propagate through it, resulting in deadlock. This may not be true if only a few forks of the sleep signal are stuck-at-1, in which case the combinational block may still work correctly, and not cause deadlock, depending on the DATA being processed at the time. Fortunately, through fault collapsing, the stuck-at-1 faults on the sleep signal forks can be detected during stuck-at-0 fault checking on the output of the gates within the combinational block.

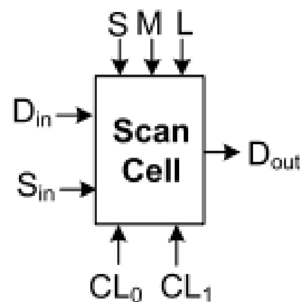

Fig. 2: SCL scan cell.

A. a single {DATA, NULL} pair to propagate through the SCL pipeline, all stuck-at faults on the inputs and output of all completion C-elements can be detected.

B. Based on Theorem 2, by disabling the sleep signal, the SCL combinational logic block becomes a normal Boolean circuit that can then be checked for stuck-at faults using the traditional combinational ATPG tools.

C. The stuck-at faults on the sleep signal forks within a combinational logic block are either untestable

(stuck-at-0 faults) or can be ignored through fault collapsing (stuck-at-1 faults).

D. The stuck-at faults on the sleep signal forks within a completion detector block are either untestable (stuck-at-0 faults) or can be detected during the test of the completion C-elements (stuck-at-1 faults).

E. The stuck-at faults on the sleep signal forks within a register block are best tested through a scan chain design to be discussed. A scan chain is also needed to apply the ATPG generated test patterns to the combinational logic blocks.

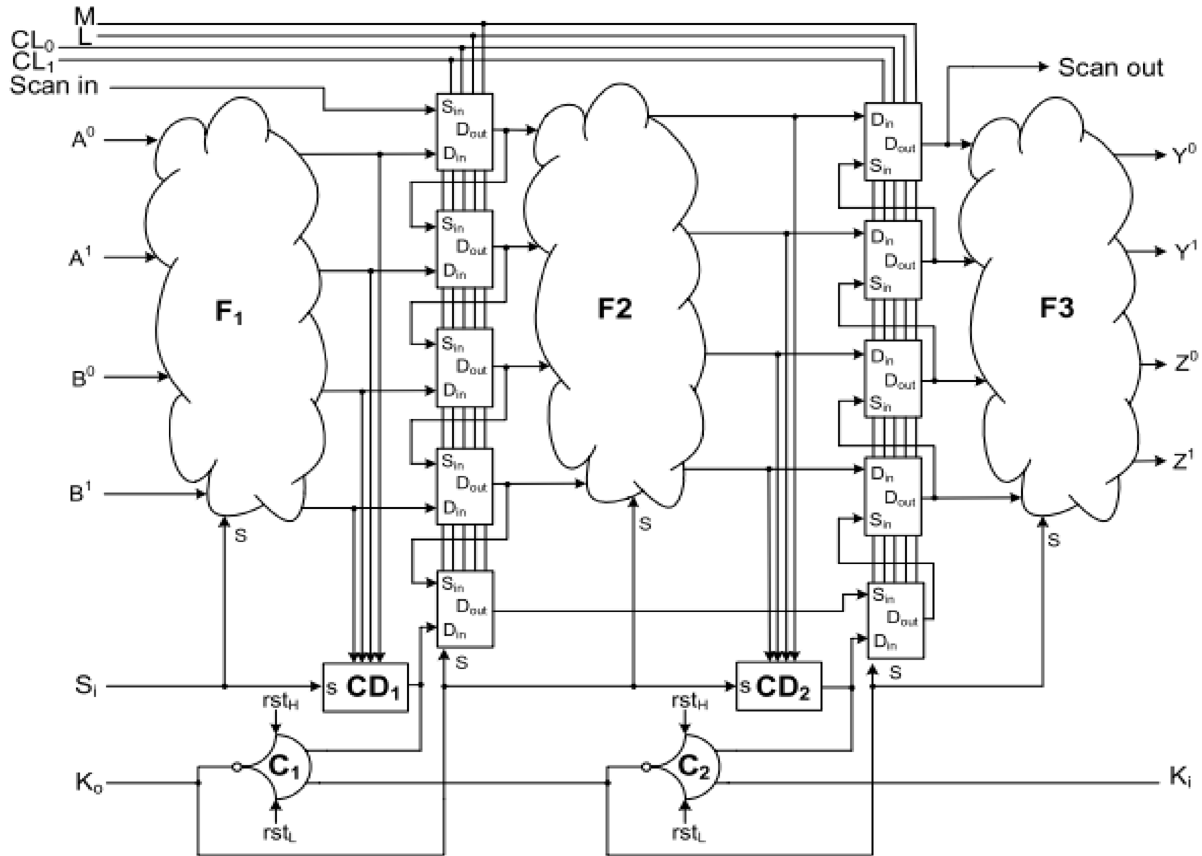

Fig. 3:SCL scan-chain design.

Fig. 3 shows a typical SCL pipeline with two primary inputs (A and B) and two primary outputs (Y and Z), when registers are replaced with scan cells. Similar to a traditional scan chain design, the scan cells form a long shift register in test mode so that the test vectors can be shifted in, and the captured results can be shifted out. There are, however, two major differences compared to the original SCL pipeline in Fig. 1. First, the output of completion detectors is also fed to scan cells. As discussed in Section III-A, stuck-at-0 faults on the output of gates within a completion detector block can be easily detected since they cause the pipeline to stall. However, detecting stuck-at-1 faults is not as easy, since the sleep signal hides those faults. Therefore, in order to detect stuck-at-1 faults, a traditional ATPG method must be used similar to the case of combinational logic faults. Since the output of a completion detector is not readily available for observation, adding an extra scan cell solves the problem. The overhead associated with this extra scan cell is negligible considering that only a single additional scan cell is needed per pipeline stage. The second difference of the new SCL pipeline is adding an additional input signal, rstL , to the completion C-element gates. This additional signal disables the sleep signals in test mode by forcing them to low. Signal rstH , however, does the same thing that rst does in the original pipeline, i.e., initializing the circuit to an all-NULL state by putting all the blocks into sleep mode.

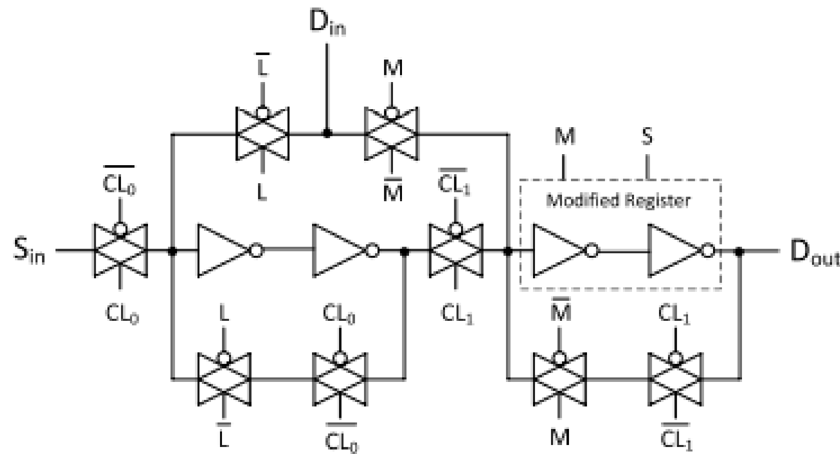

Fig. 4.:SCL scan cell design.

Fig. 4 shows our proposed implementation of the SCL scan cell. The design is made of two D-latches, one of them being the original SCL register, as shown in Fig. 3 that is reconfigured by signals M and S to become a D-latch. The modified version of the original SCL register for a single rail is shown in Fig. 10. This modified version makes use of three additional transistors to cut the feedback path and make the first half of the register look like an inverter when M = 1. For the second half of the register to look like an inverter, it is enough to just disable the sleep signal, i.e., S = 0. The entire implementation requires 32 transistors; however, in a dual-rail implementation each rail requires its own scan cell, so a dual-rail signal requires twice the number of transistors.

IV. EXPERIMENTAL RESULTS

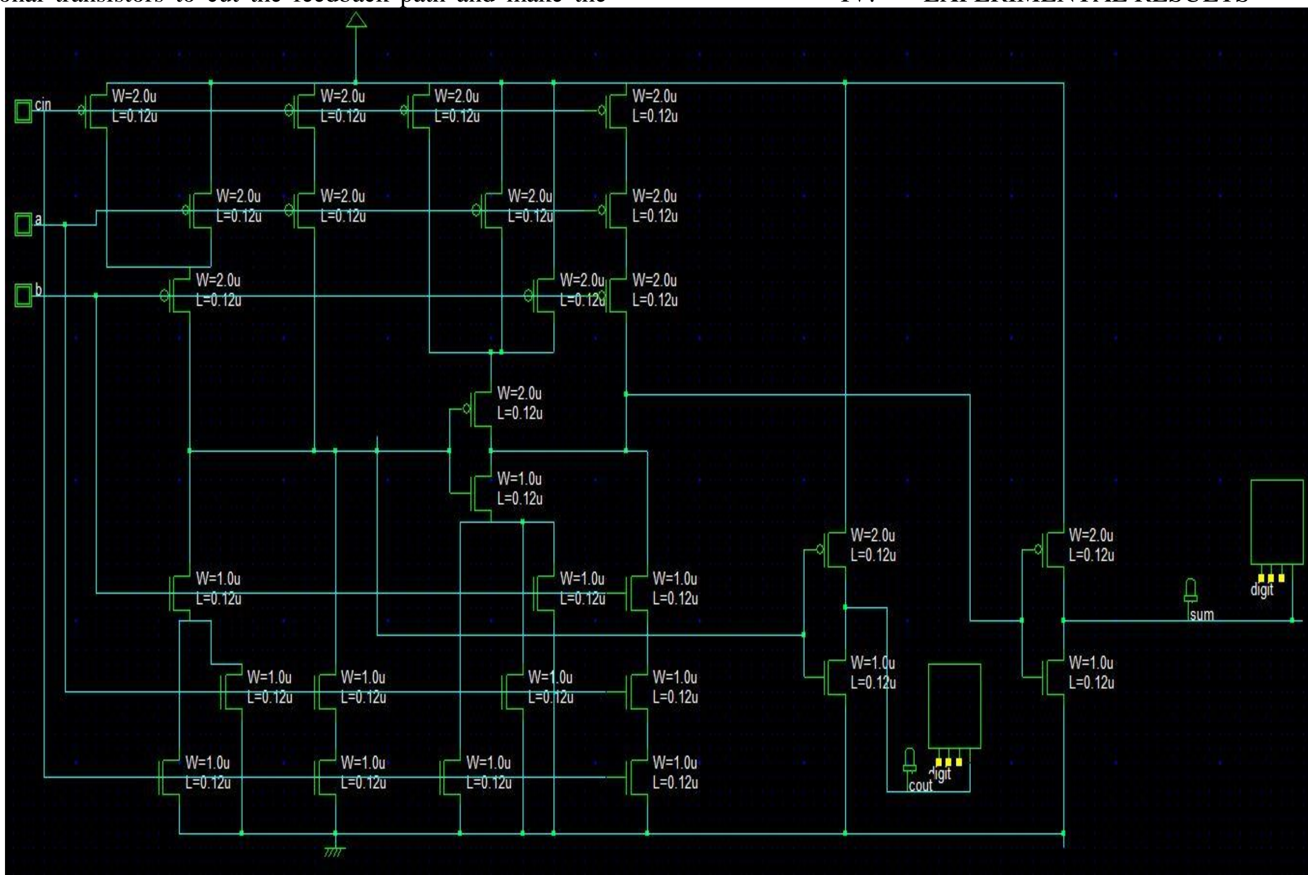

Fig. 5:SCL scan cell design.

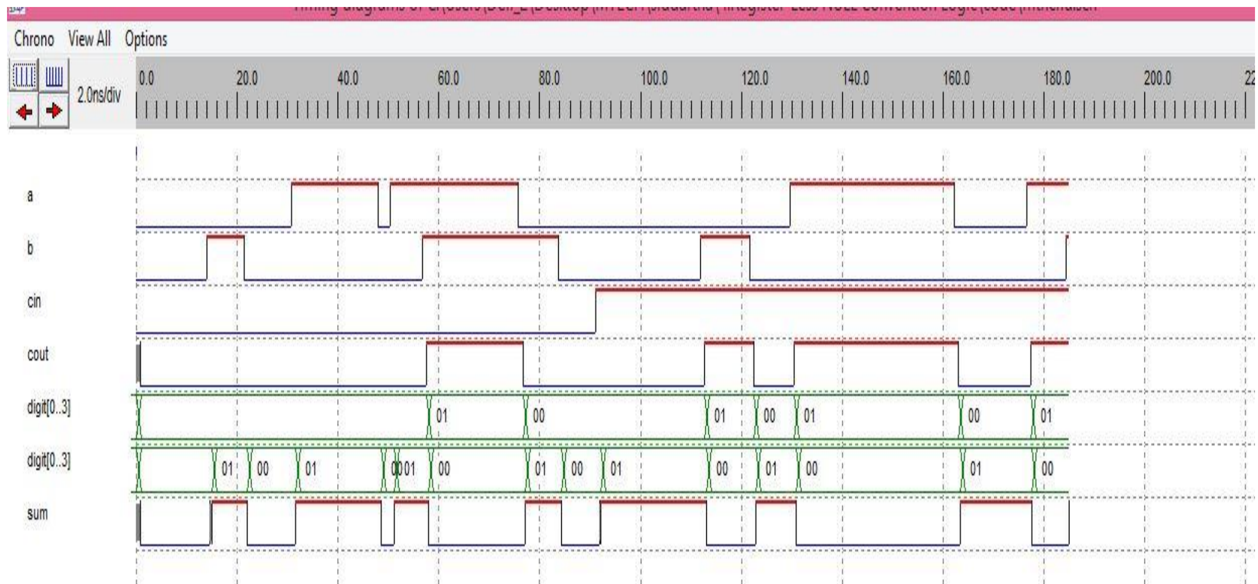

Fig. 6: Adder and subtractor simulate using SCL .

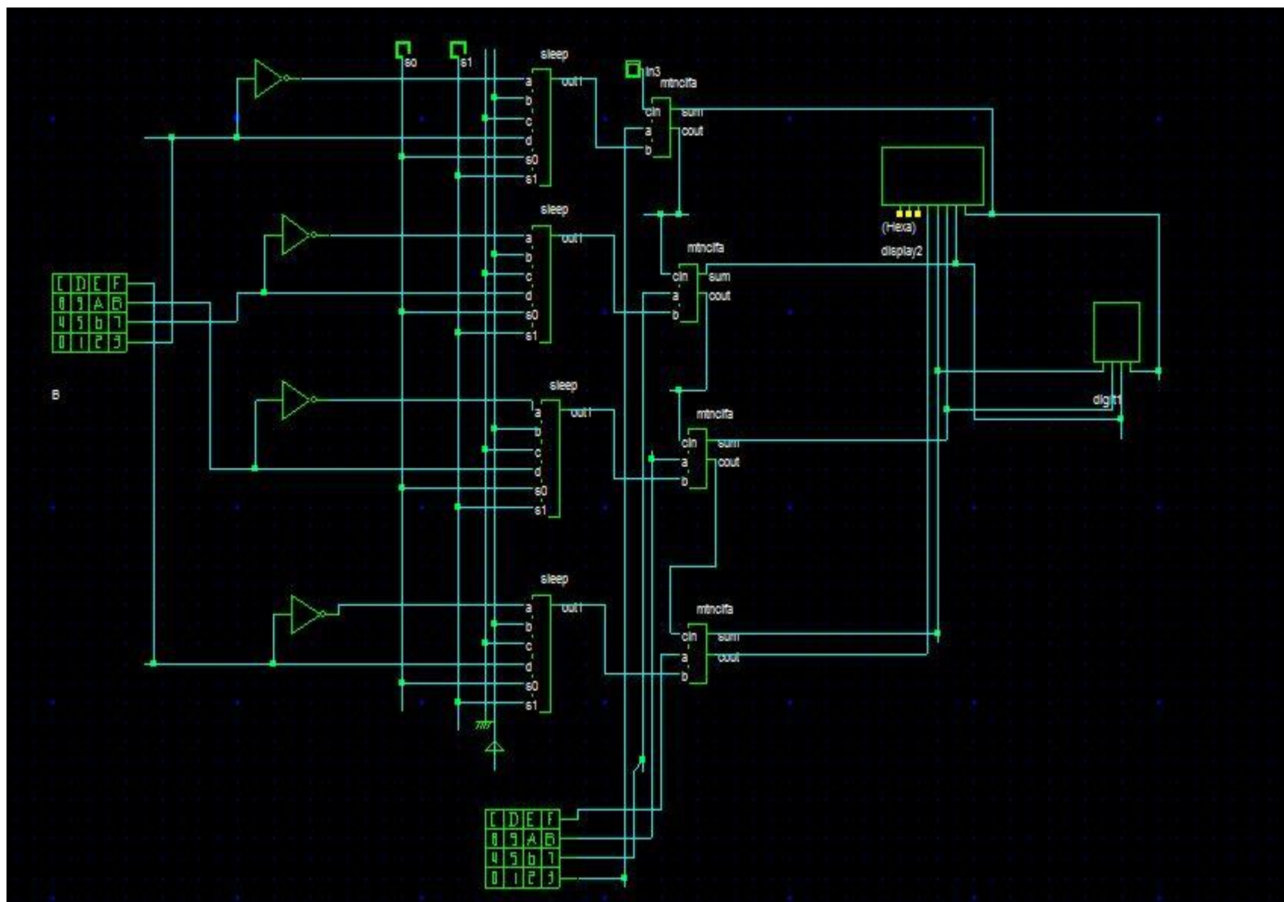

Fig. 7:Adder and subtractor design using SCL scan-chain.

V. CONCLUSION

The problem of testing SCL circuits for stuck-at faults was investigated. The faults were initially divided into two separate categories: 1) faults on logic gates and 2) faults on sleep signal forks. The faults within each category were then analyzed separately, and the impact of the faults inside each SCL component in the SCL pipeline was discussed. A comprehensive scan-based DFT methodology was then proposed based on the fault analysis and the architecture of the scan chain; and the implementation of the scan cells was elaborated. Finally, the proposed DFT methodology was validated through experimental results, showing that the methodology provides a high test coverage (more than 99%) at the cost of the usual area overhead associated with scan chain insertion.

REFERENCES

[1]. J. Di and S. C. Smith, "Ultra-low power multi-threshold asynchronous circuit design," U.S. Patent 7 977 972, Jul. 12, 2011.

[2]. J. Di and S. C. Smith, "Ultra-low power multi-threshold asynchronous circuit design," U.S. Patent 8 207 758, Jun. 26, 2012.

[3]. J. Di and S. C. Smith, "Ultra-low power multi-threshold asynchronous circuit design," U.S. Patent 8 664 977, Mar. 4, 2014.

[4]. S. C. Smith and J. Di, "Designing asynchronous circuits using NULL convention logic (NCL)," Synth. Lect. Digit. Circuits Syst., vol. 4, no. 1, pp. 1–96, 2009.

[5]. K. M. Fant and S. A. Brandt, "NULL convention logic: A complete and consistent logic for asynchronous digital circuit synthesis," in Proc. Int. Conf. Appl. Specific Syst., Archit., Processors (ASAP), Aug. 1996, pp. 261–273.

[6]. K. M. Fant, Logically Determined Design: Clockless System Design With NULL Convention Logic. New York, NY, USA: Wiley, 2005.

[7]. S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V power supply high-speed digital circuit technology with multi threshold-voltage CMOS," IEEE J. Solid-State Circuits, vol. 30, no. 8, pp. 847–854, Aug. 1995.