THx2 Programmable Logic Block Architecture for Clockless Asynchronous FPGAs

John M. Emmert, *Senior Member, IEEE*, Anvesh K. Perumalla, *Student Member, IEEE*, Tristan J. Hudson, *Student Member, IEEE*, and Luis M. Concha

***Abstract*— To address some of the challenges of asynchronous design, we propose a new, decomposable asynchronous logic block architecture based on our TH*x*2 programmable threshold cell, and we use it to implement common threshold functions found in asynchronous, null convention logic circuits. At a minimum, programmable gate arrays require a programmable logic cell that can implement a complete set of logic. It is well known that a NAND function forms a complete set of logic, and in null convention logic, the TH12 and TH22 threshold cells are used to form a basic two-input NAND function. The TH*x*2 threshold cell is capable of performing both TH12 and TH22 operations, so it too forms a complete set of logic. In this paper, we present our eight-transistor mask-programmable gate array logic cell, 16-transistor field-programmable gate array logic cell, and new decomposable field-programmable gate array logic block architecture, all based on the TH*x*2 threshold cell and suitable for implementing null convention logic asynchronous functions. To minimize the TH*x*2 threshold cell area for both TH12 and TH22 modes, we designed a layout with common Euler paths and no diffusion breaks for both modes. The highly compact nature of the TH*x*2 threshold cell–along with the symmetry of the mask- and field-programmable gate array logic cells–made it an ideal candidate for an asynchronous field-programmable logic block structure. This paper is part of an ongoing project, and it only addresses the programmable logic block architecture, not a complete FPGA fabric.**

***Index Terms*— Assurance, asynchronous, field-programmable gate array, mask-programmable gate array, null convention logic, programmable logic block, security, side-channel attacks, split manufacturing, threshold gates, trust.**

I. Introduction

SYNCHRONOUS or clocked integrated circuit (IC) systems are susceptible to side-channel attacks (SCAs), especially if they are fabricated at untrusted foundries where malicious Trojan circuits can be inserted. A malicious agent or entity can use these Trojan circuits to steal sensitive information like secret keys by monitoring power consumption, radiation, or other IC characteristics [1].

Manuscript received October 19, 2021; revised March 20, 2022; accepted April 12, 2022. Date of publication April 28, 2022; date of current version June 29, 2022. This work was supported in part by the National Science Foundation (NSF) Center for Hardware and Embedded Systems Security and Trust under Grant 1916722. This article was recommended by Associate Editor E. Blokhina. *(Corresponding author: John M. Emmert.)*

The authors are with the Department of Electrical Engineering and Computer Science, University of Cincinnati, Cincinnati, OH 45221 USA (e-mail: john.emmert@uc.edu; perumaak@mail.uc.edu; hudsonts@mail.uc.edu; luis.concha@uc.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2022.3168420.

Digital Object Identifier 10.1109/TCSI.2022.3168420

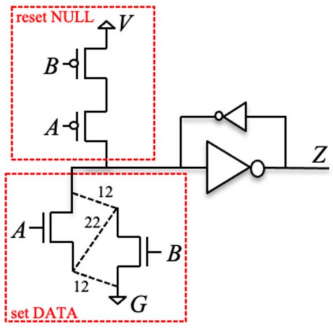


Fig. 1. Connections for TH12 and TH22 modes of the TH*x*2 cell [5]–[7].

One approach that was previously shown to obfuscate synchronized data leakage and mitigate SCAs is to use clockless asynchronous digital design [2]–[4]. Clockless asynchronous logic in the form of programmable gate array (PGA) technology offers a synergistic set of defenses against SCAs that include unsynchronized, time distributed internal switching and very regular physical structures that make it difficult to add undetectable Trojans during IC fabrication. To simplify asynchronous design and to improve capabilities, we developed unique PGA logic structures based on our TH*x*2 threshold cell shown in Fig. 1 [5]–[7]. In Fig. 1, the dotted lines represent programmable connections that enable the TH12 and TH22 modes of the TH*x*2 threshold cell. To enable the TH12 mode of TH*x*2, the dotted lines labeled "12" are connected and the line labeled "22" is disconnected, and the opposite connections are made for the TH22 mode.

In this paper, we extend our previous work and present a new field-programmable logic block (LB) architecture for clockless asynchronous logic design. We leverage our compact multi-mode programmable TH*x*2 cell and describe the development, operation, and analysis of asynchronous functional structures: TH*x*2 mask-programmable gate array (MPGA), field-programmable gate array (FPGA) logic cell, and new, unique decomposable FPGA asynchronous LB (ALB) architecture. Our ongoing work focuses on using our ALB structure to implement a fully asynchronous FPGA. It should be noted that this paper focuses on the ALB architecture, not the complete asynchronous FPGA architecture. The paper is organized as follows: in section II, Background, we briefly cover SCAs, asynchronous TH*mn* threshold gates, null convention logic (NCL), MPGAs, and decomposable FPGA LBs as they apply here. In sections III, IV, V, and VI, we describe the

development, operation, and analysis of the compact THx2 threshold cell, MPGA, FPGA asynchronous logic cell, and new decomposable FPGA ALB, respectively, followed by conclusions in section VII.

II. Background

In this section we provide a brief overview of SCAs, asynchronous TH*mn* threshold logic gates, asynchronous NCL, MPGAs, and decomposable FPGA LBs as they relate to the new ALB structure and its application.

A. Side-Channel Attacks

One way to describe a Trojan circuit is as *an extra circuit added by the manufacturer during the IC fabrication process*. Trojan circuits can be used in a positive way to provide feedback to the manufacturer and designer on the manufacturing process, but they can also be used maliciously to leak private or secret information during normal IC operation. Several types of Trojans have been developed that require minimal area overhead and are very difficult to detect either during regular IC testing or IC reverse engineering [8], [9].

Side-channel attacks indirectly monitor signal values by exploiting existing circuitry or added (Trojan) circuitry to steal secret or private information. With algorithmic or other system implementation details, side-channel attacks can leverage electromagnetic radiation, temperature variations, power usage, or other operational characteristics to allow an untrusted agent or entity to indirectly extract private data during normal IC operation. One common example used to steal secret keys is a power based SCA that uses a malicious off-chip leakage enabled side-channel (MOLES) Trojan circuit [10].

One way that has been shown to mitigate this type of SCA is to leverage asynchronous circuit designs in order to desynchronize or distribute (relative to time) power consumption [2], [3]. This makes it difficult for the attacker to decipher power data. Furthermore, researchers have demonstrated asynchronous logic offers greater tamper-resistance to counteract SCAs [3], [4]. Thus, an asynchronous FPGA/MPGA can be used to mitigate SCAs.

B. Asynchronous THmn Threshold Gates and NCL

There are many types of asynchronous design techniques ranging from locally clocked to completely clockless [11]–[17]. Each has its own advantages and disadvantages. One type of clockless asynchronous logic is NCL [12]. An NCL circuit implementation is self-timed because data flows through NCL networks in waves [18]. A data wave is only processed when all input data is available. Since processing only occurs when data is available, there are no timing assumptions, and thus NCL guarantees data sequencing and correct data arrival at the receiver irrespective of various gate, process, and wire delays [12]. The NCL circuit scheme uses dual-rail, multi-wire encoding. One wire represents a logic '1' and one wire a logic '0.' For example, a dual rail signal A has a logic '1,' A_1, wire and a logic '0,' A_0, wire.

To process the dual-rail signals, the backbone of NCL asynchronous circuits is the threshold gate [12], [17]. The threshold gate has the property of hysteresis, and is denoted by TH*mn*, where the output of the gate is asserted (set) if the gate has a valid DATA value on m (threshold) of its n-inputs. In other words when its threshold is met, its output is asserted. The output stays asserted (hysteresis) until all inputs go back to NULL in its reset phase [12]. Fig. 2 shows a generic TH*mn* gate symbol. For practical implementation using only CMOS transistors, a positive supply voltage (*VDD* or '1') represents a DATA value, and a negative supply voltage (*GND*, *VSS*, or '0') represents a NULL value. Our THx2 cell is capable of implementing both TH12 and TH22 gates, thus forming a complete set of logic [5]–[7]. It should be noted that besides being clockless, NCL asynchronous circuits offer advantages to SCA avoidance that include unsynchronized (distributed in time) and low power consumption [3], [4].

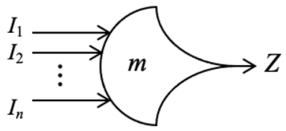


Fig. 2. Example TH*mn* threshold gate symbol with n-inputs and threshold m.

C. Mask-Programmable Gate Arrays

Mask-programmable gate arrays form a special class of application specific integrated circuits (ASICs), and they are manufactured through a two-stage manufacturing process that can take advantage of split manufacturing [19]–[24]. During the first stage, all of the lower layers (wells, diffusion, diffusion vias, poly, etc.) are fabricated. At this stage, a very compact logic *base* cell is repeatedly replicated. A requirement of the base cell is that it forms a complete set of logic, and that way, by carefully wiring a set of base cells together, any digital circuit or system can be formed. For conventional digital logic, the most common base cell is the NAND gate. The NAND gate forms a complete set of logic, so by wiring NAND gates together, any digital circuit can be implemented.

For an MPGA circuit implementation, once stage one is complete, a die or wafer with uncommitted base cells can be shelved for later use. When a digital IC system is needed, the base MPGA is taken off the shelf, and during the second manufacturing stage the rest of the via and metal layers are added to commit or wire together base cells to form the desired digital system. This two-stage process allows stage one to occur at one manufacturing facility, and stage two to occur at a second manufacturing facility, thus providing the extra layer of security available through split manufacturing. We used our THx2 cell as a base cell for an asynchronous MPGA architecture [6].

D. Field-Programmable Gate Array Logic Blocks

Field-programmable gate arrays or FPGAs can be simply defined as reprogrammable ASICs, and their architectures and basic components have been thoroughly studied [25]–[28]. Here, we limit our discussion to LB architectures as they apply to our ALB architecture. Historically LBs implement digital combinational logic functions of their inputs, and they often

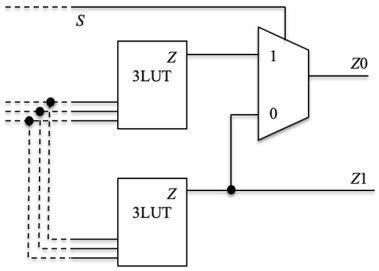


Fig. 3. Example decomposable LB with two 3LUT logic cells.

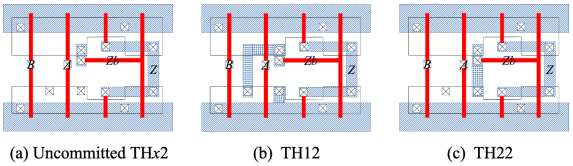
contain a synchronous component in the form of a D flip-flop. The portion of the LB that implements combinational logic functions can generically be described as a functional unit or logic cell. Ideally logic cells should be capable of implementing a complete set of logic, and most conventional, non-asynchronous commercially available FPGAs use one of three basic logic cell types: look-up tables (LUT), multiplexors, or programmable logic arrays. We present an asynchronous logic cell based on our TH$x2$ threshold cell. Our logic cell can implement a complete set of logic, and the field programmable version only requires one memory cell (MC) to program it [5].

Most commercially available LBs include multiple logic cells. This enables them to implement either multiple or larger logic functions. To make them even more utilitarian, most LBs are also decomposable. Decomposability allows multiple logic cells internal to a single LB to be combined to implement larger logic gates with more inputs. Fig. 3 shows an example decomposable LB composed of two, three-input LUT logic cells (3LUTs). With the addition of a two-transistor multiplexor, the two 3LUTs can be combined into a single 4LUT. The select input, S, of the multiplexor forms the most significant address bit of the 4LUT, and as shown by the dotted lines, the three address bits of the 3LUTs can be bitwise connected to form the three least significant bits of the 4LUT address. Our new, unique programmable ALB is based on a decomposable structure that allows us to program not only multiple THmn functions in a single ALB, but also to combine TH$x2$ logic cells to form larger THmn functions with more inputs.

III. Asynchronous TH$x2$ Threshold Cell

For additional background to our new ALB, in this section we briefly describe the design, implementation, and analysis of the TH$x2$ logic cell or threshold gate [5]–[7]. A basic requirement for PGA logic structures is a compact logic cell that forms a complete set of logic. To address this requirement for our ALB, we presented the transistor diagram of our TH$x2$ cell that is capable of operating in either a TH12 or TH22 mode via simple connection changes in Fig. 1 [5]–[7].

The TH$x2$ cell layout area was optimized by minimizing diffusion breaks and determining common node positions for both the TH12 and TH22 modes. To accomplish this, we used Euler graphs and an attribute of the NMOS transistor in the small feedback inverter in Fig. 1 [6], [7]. In the TH12 mode, the NMOS transistor in the small feedback inverter is unnecessary, and by tying both its *source* and *drain* nodes



(a) Uncommitted THx2 (b) TH12 (c) TH22

Fig. 4. Topological layouts of the TH$x2$ threshold cell (a) and modes (b), (c).

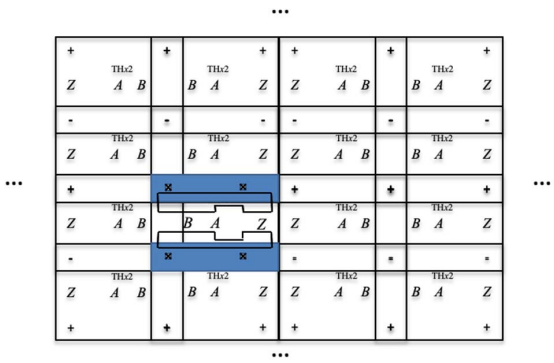


Fig. 5. 2-D array of TH$x2$ cell for MPGA layout.

to G, we determined the following common Euler path for the *gate* nodes of the PMOS and NMOS transistors for both the TH12 and TH22 threshold gates: B, A, Z, Zb. This led to the array-able topological layout of the TH$x2$ cell shown in Fig. 4(a). Fig. 4(b) and Fig. 4 (c) show added connections for the TH12 and TH22 modes of the TH$x2$ cell, respectively. The transistor sizing in Fig. 4 is dependent on the target technology node, and we show sizing for our tests below. The TH$x2$ threshold cell forms the basis for each of our asynchronous PGA structures: MPGA, FPGA logic cell, and decomposable ALB.

IV. Asynchronous TH$x2$ MPGA

The TH$x2$ cell from section III readily migrates to a very regular MPGA structure [6], [7]. The MPGA instantiation of the TH$x2$ cell not only provides a platform to take advantage of asynchronous digital design to protect against SCAs, but implementation as an MPGA provides additional protection as well. Very regular MPGA structures and split manufacturing at multiple fabrication facilities make it more difficult for untrusted agents to insert unnoticed Trojans. We used the highly compact and very array-able TH$x2$ threshold cell shown in Fig 4 as the basis for an MPGA architecture.

Fig. 5 shows the overlapping layout topology used to form the highly compact asynchronous MPGA layout. It should be noted that the Euler path topology and supply node location allow three of four sides to overlap. To make use of the TH$x2$ logic cell in an MPGA for implementing digital systems, we also need a set of standard digital gates, or a library of digital functions built around the TH$x2$ cell. To accomplish this, we created both a set of standard asynchronous logic gates such as NAND, NOR, AND, OR, etc. and a set of standard threshold gates such as TH12, TH22, TH13, TH23, TH33, etc. As an example, in Fig. 6 (a) we show how the equations for the logic '0' output wire, Z_0, and the logic '1' output wire, Z_1, are determined for a common NCL NAND2 function using a typical k-map structure [12]. The TH12, $A + B$, mode of the

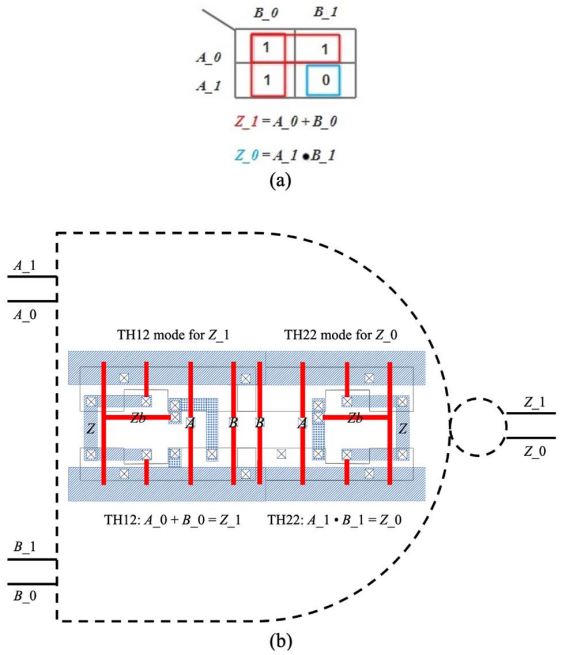


Fig. 6. Example THx2 NAND implementation: (a) Example wire connections for the Z_1 (TH12 mode) and Z_0 (TH22 mode) NAND outputs, and (b) Example k-map showing the dural-rail NCL Z_1 and Z_0 NAND function.

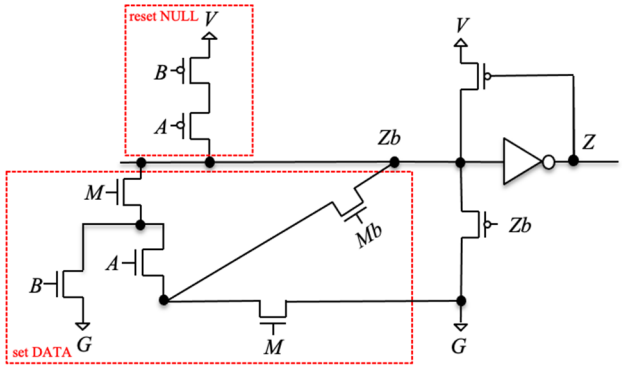


Fig. 7. Transistor diagram of the FPGA THx2 logic cell.

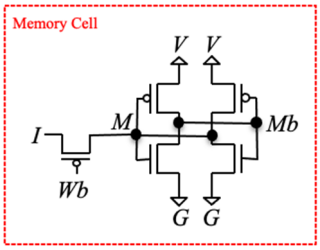


Fig. 8. Five transistor memory cell used for the FPGA THx2 logic cell.

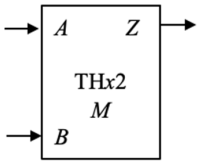


Fig. 9. Symbol for the FPGA THx2 logic cell with programming bit, M.

THx2 cell is used to implement the Z_1 output wire of the NCL NAND2 output, and similarly, the TH22, $A \cdot B$, mode is used to implement the Z_0 output wire. In Fig. 6 (b) we show the commitment of the two THx2 cells that form the NCL NAND2 function. Since the NAND2 is a complete set of logic, this also shows that the THx2 cell forms a complete set of logic. Other common functions are formed in a similar way. It should be noted that the library of functions we developed can be used (if not optimally) with standard computer aided design (CAD) tools to synthesize behavioral digital designs or (more optimally) with specialized CAD tools that target asynchronous designs. The lef, lib, gds2, etc. versions of the cells are compatible with most standard commercial CAD tools to synthesize and layout large digital systems or with specialized CAD tools that specifically target asynchronous designs.

V. Asynchronous THx2 FPGA Logic Cell

Similar to the THx2 MPGA logic cell, the FPGA logic cell is built around the THx2 threshold cell; however, instead of committing to the TH12 or TH22 mode during fabrication, we added eight extra transistors to make its mode programmable [5], [7]. The compact programmable THx2 FPGA logic cell is shown in Fig. 7. For completeness, we briefly describe the operation of the programmable THx2 FPGA logic cell. A detailed development is provided in [5]. In Fig. 7, the mode of THx2 is controlled by setting the value of M (and by default, Mb). A value of M = '1' sets the mode of the THx2 cell to TH12, and a value of M = '0' sets the mode to TH22. The value of M is stored in a standard five-transistor memory cell (Fig. 8). The number and type of each transistor in Fig. 7 and Fig. 8 were carefully selected to provide a highly compact, programmable logic cell suitable for the tight, 2-D arraying required by FPGAs.

The widths of the transistors in Fig. 7 need to be sized, and the sizes are dependent on the target technology node. The width of the NMOS transistor in the output inverter should be minimized, and the PMOS should be large enough to keep the rise time similar to the fall. The widths of the two PMOS HOLD transistors should be set to minimum values since their function is to HOLD the output when it is not transitioning. At a minimum, the widths of the devices in the "set to DATA" and "reset to NULL" sub-circuits in Fig. 7 need to be large enough to overpower the PMOS HOLD transistors. Otherwise, their size can be varied to provide faster switching time (larger width) or vice versa.

VI. THx2 Asynchronous FPGA Logic Block (ALB)

The compact, multi-mode programmable THx2 FPGA logic cell (shown in Fig. 7) forms the basic building block of the new ALB. The symbol for the programmable logic cell with programming bit M is shown in Fig. 9. Most commercially available FPGAs include multiple logic cell functional units in a single LB in order to reduce the number and size of routing resources and overall FPGA IC area [25]–[28]. One approach used in the past to determine the number of logic cell inputs and total number of logic cell units in each LB

was by mapping benchmark (BM) LB circuits to experimental and switch box models and optimizing equations that describe circuit metrics like overall average FPGA IC area [28]. Given the lack of standard asynchronous BM circuits and supporting CAD tools, we focused on implementing as many standard asynchronous TH*mn* threshold functions and conventional logic gate functions as possible in a single ALB.

A. THx2 Decomposable Programmable ALB Development

We considered several topologies with various numbers of TH$x2$ logic cells and input and output combinations. While they are not asynchronous BM circuits, the topology we developed is partially based on analysis of the ISCAS '89 and '99 BM circuits (also known as the ITC '99 BM circuits) [29], [30]. We tabulated the total number of combinational gates with two, three, four, and five-inputs and gate types for each of the BM circuits, and we used the general structure of the NCL implementation of those gates to drive the structure of our ALB. In addition to conventional, standard gates found in the non-asynchronous BM circuits, we also targeted an architecture that supports the standard TH*mn* threshold gates found in most common asynchronous libraries [12]. While supporting the most common TH*mn* gates, an additional goal was to enable the TH*mn* gates that appeared most often in the BM circuits so that it is possible to implement them without having to combine multiple ALBs together.

We analyzed a total of 58 conventional circuits from the ISCAS '89 and '99 BM circuits. On average there were 11,900 two-input, 795 three-input, 349 four-input, and 149 five-input gates in each BM circuit. Statistically, 90.2% were two-input, 6.0% were three-input, 2.6% were four-input, and 1.1% were five-input logic gates. By combining we see 96.2% of the logic gates were two or three-input gates. In addition, the analysis of the '89 and '99 ISCAS BM circuits showed that the majority of the logic gates were simple gates like AND, OR, NAND, and NOR. These basic gates are implementable with NCL TH*mn* threshold gates of the form TH1n and THnn, where n is also the number of logic gate inputs. Our decomposable ALB architecture shown in Fig. 10 is designed to optimize the number of these simple logic gates per ALB. We considered even combinations of TH$x2$ logic cells in each ALB to accommodate the fact that input and output signals for NCL logic gates are multi-rail with both a logic '1' and '0' wire and driving threshold circuit. The ALB in Fig. 10 can implement two simple, two-input NCL logic gates with dual-rail input and output signals or one simple, three-input NCL logic gate. Overall, approximately 96% of the logic gates in the BM circuits can be implemented in a single ALB without requiring external routing.

Besides the ISCAS BM circuits, we also analyzed the standard set of TH*mn* threshold gates found in most asynchronous libraries [31]. We set the number of TH$x2$ logic cells to four in order to accommodate as many of the standard TH*mn* threshold gates as possible in a single ALB using as little external routing as possible. The ALB in Fig. 10 can implement four, two-input TH$m2$ threshold gates and two, three-input TH$m3$ threshold gates. A programming example is shown later in section VI, and Tables I and II provide a general

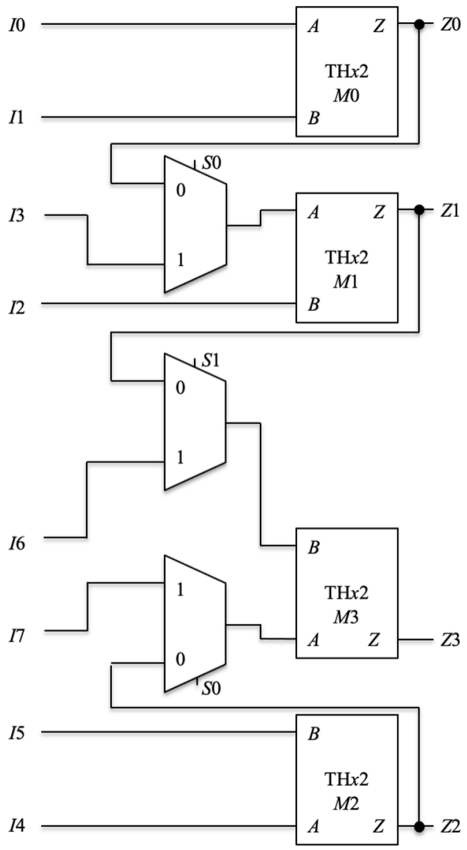


Fig. 10. Topology of the decomposable asynchronous FPGA LB or ALB.

description of how to program the TH$x2$ decomposable ALB to implement several standard logic gates and TH*mn* threshold gates using only a single TH$x2$ ALB. There are a total of 27 fundamental standard NCL gates that are widely used [31]. Each one of these 27 functions can be represented using TH$x2$ gates. As our ALB is made of four TH$x2$ cells, 67% of these standard NCL gates can be represented using one ALB and 100% of them can be implemented using at most two ALBs.

B. Programming the THx2 Decomposable ALB

Fig. 10 shows the basic topology of the FPGA ALB. In Fig. 10, the mode of the ALB is set through the concatenated multiplexor select signals, $S = \{S1S0\}$, and four, TH$x2$ FPGA logic cell programming memory cell (MC) values, $M0$–$M3$.

In Fig. 10, an MC value of '1' enables the TH12 mode of the FPGA logic cell, and a value of '0' enables the TH22 mode. By setting the concatenated select signal, S, to "11," the four FPGA logic cells operate independently as four separate TH$x2$ cells, and by setting the MCs to appropriate values, one ALB can implement two independent, two-input simple NCL logic gates like AND2, OR2, NAND2, and NOR2. It should be noted that the concept of inversion for NCL logic gates is captured in the routing of the circuit. An inverted signal is implemented by crossing the logic '1' and logic '0' wires, so no additional circuitry is required to implement inverted or noninverted signals. For example, the exact same TH12 and TH22 gates can be used to implement an AND2 and a

TABLE I
EXAMPLE MODE SETTINGS FOR COMMON ALB NCL LOGIC GATES

Gates	S	M / Z ($M0/Z0$ $M1/Z1$ $M2/Z2$ $M3/Z3$)	
Input combination		$I0$: A_0 $I1$: B_0 $I2$: A_1 $I3$: B_1	$I4$: C_0 $I5$: D_0 $I6$: C_1 $I7$: D_1
2 x AND2	11	$M0$M1 = "10" ⇒ $Z0$: $F\_0 = A\_0 + B\_0$ $Z1$: $F\_1 = A\_1 \bullet B\_1$	$M2$M3 = "10" ⇒ $Z2$: $G\_0 = C\_0 + D\_0$ $Z3$: $G\_1 = C\_1 \bullet D\_1$
2 x NAND2	11	$M0$M1 = "10" ⇒ $Z0$: $F\_1 = A\_0 + B\_0$ $Z1$: $F\_0 = A\_1 \bullet B\_1$	$M2$M3 = "10" ⇒ $Z2$: $G\_1 = C\_0 + D\_0$ $Z3$: $G\_0 = C\_1 \bullet D\_1$
2 x OR2	11	$M0$M1 = "01" ⇒ $Z0$: $F\_0 = A\_0 \bullet B\_0$ $Z1$: $F\_1 = A\_1 + B\_1$	$M2$M3 = "01" ⇒ $Z2$: $G\_0 = C\_0 \bullet D\_0$ $Z3$: $G\_1 = C\_1 + D\_1$
2 x NOR2	11	$M0$M1 = "01" ⇒ $Z0$: $F\_1 = A\_0 \bullet B\_0$ $Z1$: $F\_0 = A\_1 + B\_1$	$M2$M3 = "01" ⇒ $Z2$: $G\_1 = C\_0 \bullet D\_0$ $Z3$: $G\_0 = C\_1 + D\_1$
Input combination		$I0$: A_0 $I1$: B_0 $I2$: C_0 $I4$: A_1 $I5$: B_1 $I6$: C_1	
1 x AND3	10	$M0$M1M2M3 = "1100" ⇒ $Z1$: $F\_0 = A\_0 + B\_0 + C\_0$ $Z3$: $F\_1 = A\_1 \bullet B\_1 \bullet C\_1$	
1 x NAND3	10	$M0$M1M2M3 = "1100" ⇒ $Z1$: $F\_1 = A\_0 + B\_0 + C\_0$ $Z3$: $F\_0 = A\_1 \bullet B\_1 \bullet C\_1$	
1 x OR3	10	$M0$M1M2M3 = "0011" ⇒ $Z1$: $F\_0 = A\_0 \bullet B\_0 \bullet C\_0$ $Z3$: $F\_1 = A\_1 + B\_1 + C\_1$	
1 x NOR3	10	$M0$M1M2M3 = "0011" ⇒ $Z1$: $F\_1 = A\_0 \bullet B\_0 \bullet C\_0$ $Z3$: $F\_0 = A\_1 + B\_1 + C\_1$	

TABLE II
EXAMPLE MODE SETTINGS FOR COMMON ALB TH*mn* THRESHOLD GATES

Gates	S {$S1S0$}	M / Z $M0/Z0$ $M1/Z1$ $M2/Z2$ $M3/Z3$	
4 x TH*m*2	11	$Mj = 1$ ⇒ Zj is TH12 mode $Mj = 0$ ⇒ Zj is TH22 mode	
2 x TH13	10	$I0$=A, $I1$=B, $I2$=C $M0$M1 = "11" ⇒ $Z1 = A + B + C$	$I4$ =A, $I5$=B, $I6$=C $M2$M3 = "11" ⇒ $Z3 = A + B + C$
1 x TH23	00	$I0$ = A, $I1$ = B, $I2$ = C, $I4$ = A, $I5$ = B $M0$M1M2M3 = "1001" ⇒ $Z3 = AB + AC + BC = AB + C\,(A + B)$	
2 x TH33	10	$I0$=A, $I1$=B, $I2$=C $M0$M1 = "00" ⇒ $Z1 = ABC$	$I4$=A, $I5$=B, $I6$=C $M2$M3 = "00" ⇒ $Z3 = ABC$
2 x TH23w2	10	$I0$=A, $I1$=B, $I2$=C $M0$M1 = "01" ⇒ $Z1 = A + BC$	$I4$=A, $I5$=B, $I6$=C $M2$M3 = "01" ⇒ $Z3 = A + BC$
2 x TH33w2	10	$I0$=A, $I1$=B, $I2$=C $M0$M1 = "10" ⇒ $Z1 = A\,(B + C)$	$I4$=A, $I5$=B, $I6$=C $M2$M3 = "10" ⇒ $Z3 = A\,(B + C)$
1 x TH14	00	$I0 = A$, $I1 = B$, $I2$ = C, $I4 = D$, $I5 = D$ $M0$M1M2M3 = "1111" ⇒ $Z3 = A + B + C + D$	
1 x TH44	00	$I0 = A$, $I1 = B$, $I2 = C$, $I4 = D$, $I5 = D$ $M0$M1M2M3= "0000" ⇒ $Z3 = ABCD$	
1 x TH15	00	$I0 = A$, $I1 = B$, $I2 = C$, $I4 = D$, $I5 = E$ $M0$M1M2M3= "1111" ⇒ $Z3 = A + B + C + D + E$	
1 x TH55	00	$I0 = A$, $I1 = B$, $I2 = C$, $I4 = D$, $I5 = E$ $M0$M1M2M3= "0000" ⇒ $Z3 = ABCDE$	
1 x TH34w3	00	$I0 = B$, $I1 = C$, $I2 = D$, $I4 = A$, $I5 = A$ $M0$M1M2M3= "0001" ⇒ $Z3 = A + BCD$	
1 x TH24w22	00	$I0 = C$, $I1 = C$, $I2 = D$, $I4 = A$, $I5 = B$ $M0$M1M2M3= "0011" ⇒ $Z3 = A + B + CD$	
1 x TH54w32	00	$I0 = A$, $I1 = C$, $I2 = D$, $I4 = A$, $I5 = B$ $M0$M1M2M3= "0001" ⇒ $Z3 = AB + ACD$	

NAND2. The only difference between the two instantiations is the logic '1' and '0' output wires are swapped to achieve the inverted functionality.

C. THx2 ALB Circuit Operation

To demonstrate operation of the new THx2 ALB and Table I, we program the MC firmware and concatenated multiplexor switches, S, of a single ALB to implement two simple NCL logic functions: NAND2 and OR2. The ALB in Fig. 10 is broken into two parts. The first part, with MCs $M0$ and $M1$, is programmed to implement a two-input NCL NAND2 gate, and the second part with MCs $M2$ and $M3$, is programmed to implement a two-input NCL OR2 gate. To set the ALB to implement two separate, two-input logic gates, we set S = "11." In this multiplexor select mode, the THx2 Zk outputs are functions of the Ij input signal values and Mk MC values as follows:

0. $Z0$ = THx2($I0$, $I1$, $M0$)
1. $Z1$ = THx2($I2$, $I3$, $M1$)
2. $Z2$ = THx2($I4$, $I5$, $M2$)
3. $Z3$ = THx2($I6$, $I7$, $M3$)

Tables I and II show some common logic functions and TH*mn* threshold gates found in most NCL asynchronous libraries. They describe the number of instantiations for the gates in a single ALB and the S and M settings for the functions. The first column of both Tables lists the number of instances and the logic or threshold gate type. The second column gives the settings for the concatenated multiplexor select signals, S = {$S1S0$}, that set the inputs to each THx2 cell in the ALB. The third column describes the input signal assignments for each THx2 input, Ij; the MC mode settings, Mj; and the THx2 output, Zj, assignments and the corresponding form of the equation assigned to the outputs. It should be noted that it is possible to mix modes by carefully mixing the gate types programmed to each individual THx2

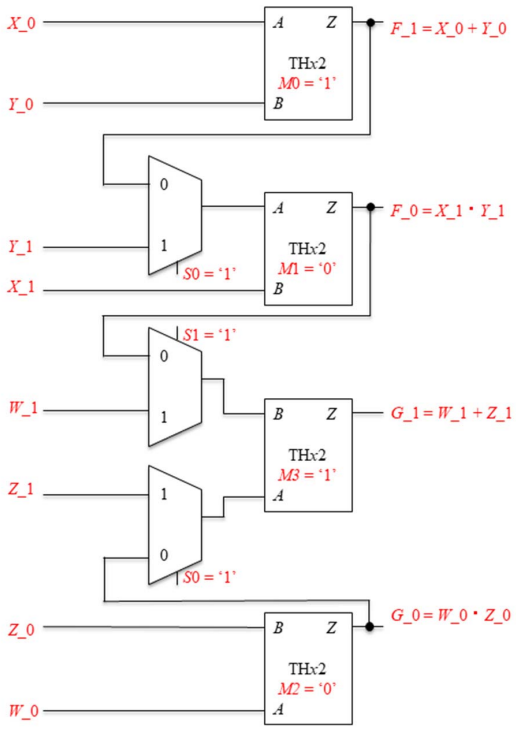


Fig. 11. Example NCL NAND2 and OR2 function using the basic ALB.

cell. The highlighted green section in Table I selects different input signal assignments.

For the TH12 mode, $Z = A + B$, of a THx2 cell, the MC M value is set to '1,' and similarly for the TH22 mode, $Z = A \cdot B$, the MC value M is set to '0.' As shown in Fig. 11, for the NCL NAND2 example, we assign the dual rail X and Y inputs to $I0 - I3$ and the F output to $Z0$ and $Z1$. To program the NCL NAND2 function, $F\_1 = X\_0 + Y\_0$ and $F\_0 = X\_1 \cdot Y\_1$, we set $M0M1$ = "10" implies F: F_1 = TH12(X_0, Y_0, '1') and F_0 = THx2(X_1, Y_1, '0'). Similarly, for the NCL OR2 function, $G\_1 = W\_1 + Z\_1$ and $G\_0 = W\_0 \bullet Z\_0$, we set $M2M3$ = "01" implies G: G_1 = TH12(W_1, Z_1, '1') and G_0 = THx2(W_0, Z_0, '0').

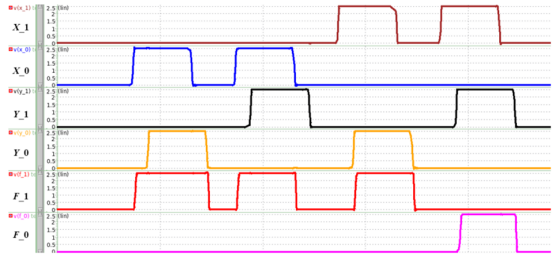
Fig. 11 shows the final IO connections and firmware programming for the NCL NAND2 and OR2 examples. More details on how to use THmn gates to form standard logic gates like AND, OR, NAND, NOR, XOR, etc. are found in [12], [31].

D. THx2 ALB Test

To demonstrate the spice simulation of the ALB, we programmed the firmware (Fig. 10) to instantiate the NAND2 and OR2 NCL logic functions shown in Fig. 11 by writing M *(M0M1M2M3)* = "1010" to the MCs. The spice simulations in Fig. 12 and Fig. 13 show the output waveforms for the NAND2 and OR2, respectively. We used the freely available TSMC 250 nm SCMOS technology [32] and 45nm FreePDK kit [33] to perform our analysis. For 250nm technology, we used minimum length, $L = 2\lambda$, and width, $W = 3\lambda$, for all of our transistors except for two pull down NMOS transistors, A and M, in THx2 logic cell, where we set $W = 4\lambda$,

X_1	X_0	Y_1	Y_0	F_1	F_0
NULL	NULL	NULL	NULL	NULL	NULL
NULL	DATA	NULL	DATA	DATA	NULL
NULL	NULL	NULL	NULL	NULL	NULL
NULL	DATA	DATA	NULL	DATA	NULL
NULL	NULL	NULL	NULL	NULL	NULL
DATA	NULL	NULL	DATA	DATA	NULL
NULL	NULL	NULL	NULL	NULL	NULL
DATA	NULL	DATA	NULL	NULL	DATA
NULL	NULL	NULL	NULL	NULL	NULL

(a)

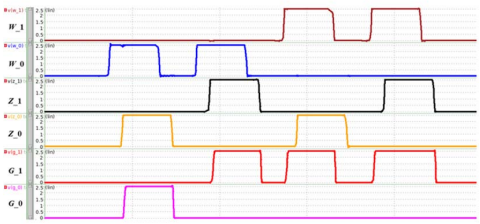


(b)

Fig. 12. ALB simulation for NCL NAND2 function.

W_1	W_0	Z_1	Z_0	G_1	G_0
NULL	NULL	NULL	NULL	NULL	NULL
NULL	DATA	NULL	DATA	NULL	DATA
NULL	NULL	NULL	NULL	NULL	NULL
NULL	DATA	DATA	NULL	DATA	NULL
NULL	NULL	NULL	NULL	NULL	NULL
DATA	NULL	NULL	DATA	DATA	NULL
NULL	NULL	NULL	NULL	NULL	NULL
DATA	NULL	DATA	NULL	DATA	NULL
NULL	NULL	NULL	NULL	NULL	NULL

(a)



(b)

Fig. 13. ALB simulation for NCL OR2 function.

in order to make the effective resistance of the path similar to that of B. It should be noted that for any target technology node, transistor sizing can be further optimized to improve timing; however, our focus was only on demonstrating ALB functionality. The worst-case delay path in Fig. 10 ($I0 \rightarrow Z0 \rightarrow Z1 \rightarrow Z3$) of our ALB using 250nm technology with $VDD = 2.5$V is 9.1671e-10 sec (1.09 GHz).

Similarly, for 45nm technology, we used minimum length, $L = 2\lambda$, and width, $W = 3\lambda$, for all of our transistors except five NMOS transistors in the pulldown network in THx2 logic cell, where we set $W = 4\lambda$, to have enough signal strength to reach the proper output. Also, for this 45nm process node, multiplexor 2×1 is designed as transmission gate model instead of pass transistors, by adding 2 more additional PMOS transistors to be able to have enough threshold to reach to the next stage with proper signal conditioning. We performed various corner case analysis using this process node.

The worst-case delay of our ALB with $VDD = 1.0$V is in between 1.86e-10 to 2.97e-10 sec that is 5.37GHz to 3.37GHz.

For the NAND2 mode (Fig. 12), we set the inputs, $I0 = X\_0$, $I1 = Y\_0$, $I2 = X\_1$, and $I3 = Y\_1$ to NULL ('0') to reset the outputs, $Z\_0 = F\_1$ and $Z\_1 = F\_0$, to NULL ('0'). Then we cycled through all sequence combinations of the inputs, X and Y, set to DATA and NULL.

Fig. 12 (a) shows the input sequence and expected output sequence in tabular form, and Fig. 12 (b) shows the actual input and output waveforms of the simulation. The output verified that anytime all input values were set to NULL, the output was also NULL. The first test case for the NAND2 was $XY =$ "00," where we expected a NAND2 output of $F =$ '1.' For the dual-rail NCL NAND2, we applied DATA values to the logic '0' inputs, X_0 and Y_0, of the NAND2 function, and we saw the expected DATA value on the F_1 NAND2 output and NULL value on the F_0 NAND2 output. We then cycled the inputs back to all NULL values to reset the outputs back to NULL values. The next test case we applied was $XY =$ "01," where we expected a NAND2 output of $F =$ '1.' For the dual-rail NCL NAND2, we applied a DATA value to the logic '0' input, X_0, and a DATA value to the logic '1' input, Y_1, of the NAND2 function, and we saw the expected DATA value on the F_1 NAND2 output and NULL value on the F_0 NAND2 output. Again, we cycled the inputs back to all NULL values to reset the outputs back to NULL values. The third test case we applied was $XY =$ "10," where we again expected a NAND2 output of $F =$ '1.' For the dual-rail NCL NAND2, we applied a DATA value to the logic '1' input, X_1, and a DATA value to the logic '0' input, Y_0, of the NAND2 function, and we saw the expected DATA value on the F_1 NAND2 output and NULL value on the F_0 NAND2 output. Again, we cycled the inputs back to all NULL values to reset the outputs back to NULL values. Finally, we applied the $XY =$ "11" test case where we expected a NAND2 output of $F =$ '0.' For the dual-rail NCL NAND2, we applied DATA values to the logic '1' inputs, X_1 and Y_1, of the NAND2 function, and we saw as soon as both logic '1' inputs have DATA on them, the F_0 output switched to the expected DATA value and the F_1 output remained NULL. Again, we cycled the inputs back to all NULL values to reset the outputs back to NULL values.

Similarly, for the OR2 mode (Fig. 13), we set the inputs, $I4 = W\_0$, $I5 = Z\_0$, $I6 = W\_1$, and $I7 = Z\_1$ to NULL ('0') to reset the outputs, $Z\_3 = G\_1$ and $Z\_2 = G\_0$ to NULL ('0'). Then we cycled through all combinations of the inputs, W and Z, set to DATA and NULL. Fig. 13(a) shows the input sequence and expected output sequence in tabular form, and Fig. 13 (b) shows the actual input and output waveforms of the simulation. The output verified that anytime all input values were set to NULL, the output was also NULL. For the first test case of the OR2, $WZ =$ "00," we expected an OR2 output of $G =$ '0.' For the dual-rail NCL OR2, we applied DATA values to the logic '0' inputs, W_0 and Z_0, and we saw the expected DATA value on G_0 and NULL value on G_1. We then reset all inputs back to NULL to reset the outputs. The next test case we applied was $WZ =$ "01," where we expected $G =$ '1.' For the dual-rail NCL OR2, we applied DATA values to inputs W_0 and Z_1, and we saw the expected DATA value on the G_1 and NULL value on the G_0. Again, we reset all inputs back to NULL to reset the outputs. The next test case we applied was $WZ =$ "10," where we expected $G =$ '1.' For the dual-rail NCL OR2, we applied DATA values to inputs W_1 and Z_0, and we saw the expected DATA value on the G_1 and NULL value on the G_0. Finally, we applied $WZ =$ "11," where we expected $G =$ '1.' For the dual-rail NCL OR2, we applied DATA values to inputs W_1 and Z_1, and we saw the expected DATA value on the G_1 and NULL value on the G_0. Last, we cycled the inputs back to all NULLs to reset the outputs back to NULL values.

The simulation outputs all corresponded to the expected values for the NCL NAND2 and OR2 logic functions, and thus validated the THx2 ALB for these two modes. Other similar simulations validate the other modes shown in Tables I and II.

E. THx2 Decomposable ALB Analysis

The ALB topology shown in Fig. 17 can be implemented using MCs to store the S_j values, or the S_j values could be routed and set at runtime for a dynamically reconfigurable ALB option. Assuming MCs are included for programming $S0$ and $S1$, the four TH$x2$ FPGA logic cell version of the ALB requires 80 transistors, $4 \times 16 = 64$ for the four TH$x2$ FPGA logic cells, 6 transistors for the three multiplexors, and 10 transistors for the two MCs to program $S0$ and $S1$.

F. Comparison of the ALB

Similar research has been performed to develop programmable asynchronous FPGAs and asynchronous logic blocks [34]. To perform a fully comparative asynchronous FPGA analysis would require a comprehensive set of asynchronous BM circuits and CAD tools shown to optimally map the BM circuits to the asynchronous FPGA architectures. While future work includes development of a full FPGA based on our TH$x2$ ALB architecture, in this work we are only presenting the ALB. We were able to qualitatively and to a limited extent quantitatively compare our ALB architecture to that of a previously presented asynchronous LB [34].

The functional block in [34] implements any function of four variables and supports carry generation. Its functional unit consists of address decoder—converts four inputs into 1by16 encoded addresses to access LUT, asynchronous LUT (ALUT), and XOR output stage. Its ALUT consists of 16 LUT elements, virtual ground inverter, two PMOS pre-charge transistors, and two output inverters.

The number of transistors required to implement the ALUT in [34] is 184, which is significantly higher than our ALB, which consists of 80 transistors (86 for lower technology nodes such as 45nm due to transmission gate logic applied to mux-2 $\times$ 1). On a foundational level, their four input LUT structure is similar to traditional Xilinx LUT except the output is divided into dual rails, thus making it a pseudo-asynchronous FPGA. In our case, the ALB structure is used to implement any of the traditional clockless NCL TH*mn*

gates, making it fully asynchronous and ideal for clockless, handshake based asynchronous circuits. The earlier work done in [34] is pioneering relative to asynchronous FPGAs, and their research laid the groundwork for our simpler, scalable, decomposable ALB. Again, in this paper we only present the asynchronous ALB and not a full asynchronous FPGA architecture. Our future work includes addition of interconnect logic, BM circuits, and supporting CAD tools to complete a fully Asynchronous FPGA.

There have also been proposals to use existing synchronous FPGAs to implement asynchronous designs [35], [36]. Their implementations are very resource heavy. For example, a conventional FPGA requires a minimum of 152 transistors for each 4-input LUT (where the SRAM memory cells are based on the 5-transistor model). For a single TH12 threshold gate, one 4-input LUT or 152 transistors is required, and that can be compared to 16 transistors for our programmable TH$x2$ cell. Again, our work focuses on efficient cells for NCL asynchronous FPGAs, and conventional FPGAs were not designed for asynchronous circuit implementations.

It should also be noted that multiple copies of our basic ALB can be included in a single LB to increase the number of possible TH*mn* threshold logic gates and complete NCL logic gates that can be implemented in a single LB. For example, while we can currently implement a four-input NCL logic gate using our basic ALBs shown in Fig. 11, two of these Fig. 10 units in a single LB would enable a complete four-input NCL AND, OR, NAND, or NOR gate in a single LB.

Researchers have used quantitative benchmark circuits available at OpenCores to analyze their FPGA prototypes [34], [37], [38]. Our future work includes implementing those BMs on a fully asynchronous FPGA for comparison. We will also explore the existing CAD tools such as VPR (an open-source place and route tool for FPGA research) and ACT (an asynchronous circuit toolkit) to further develop our asynchronous MPGA/FPGA structures [39]–[41].

VII. Summary and Conclusion

There are several advantages to clockless asynchronous digital design [12]. Some examples include: 1) the asynchronous nature of clockless logic reduces opportunities for power, electromagnetic radiation, temperature, and other SCAs, and digital noise reduction for sensitive, mixed-signal ICs; 2) data is processed at average speed versus the worst-case clock speed for synchronous sequential circuits; and 3) the difficult clock-routing step is eliminated from the IC design flow. Some common drawbacks include increased logic area, dual rail wires for all signal nets, and lack of specialized CAD tools specifically for mapping and optimizing asynchronous circuits. Given the high number of routing layers in state-of-the-art IC fabrication nodes, dual rail wires required by asynchronous techniques like NCL logic is less of a problem; However, to make asynchronous design more acceptable and commonplace, drawbacks like increased logic area and the need for specialized CAD tools should be addressed. Overall, it needs to be easier to implement asynchronous logic technologies like NCL.

To better enable simple implementation of asynchronous NCL circuits we have presented PGA structures based on our eight transistor TH$x2$ threshold logic cell. The main advantage of the TH$x2$ threshold cell is it forms a complete set of logic, making it capable of implementing complete digital systems. Since the eight transistor TH$x2$ cell is so compact and easy to array, it is very amenable to MPGA structures. Also, since the majority of most combinational digital logic circuits are based on two-input gates (as demonstrated by the ISCAS '89 and '99 BM circuits), the asynchronous MPGA based on the TH$x2$ cell offers an ideal area efficient, SCA mitigating platform for circuit implementation.

While the time required to fabricate an MPGA is significantly shorter than a standard ASIC, it is still sometimes a limiting factor. Our 16 transistor FPGA logic cell and new 80 transistor ALB offer a faster alternative to MPGA NCL asynchronous circuit implementation. The TH$x2$ ALB is decomposable and very area efficient. It can instantiate the most commonly used logic gates and TH*mn* threshold gates in a single ALB, and decomposition enables multiple gates per ALB. This results in near 100% utilization for most ISCAS BM circuits. While the MPGA architecture is fairly complete, we are currently working on a programmable interconnect structure, switch box, and connector box topology to complete the FPGA fabric. Once finished, the complete digital asynchronous FPGA implementation technology will provide the basis for quick testing and prototyping of NCL asynchronous circuits. When compared to other digital circuit technologies, FPGAs can quickly support and satisfy most prototyping and even implementation platform needs. When you combine FPGA technology with the protection provided by asynchronous circuits against malicious attacks and low power advantages, the asynchronous FPGA becomes a good choice for many applications. Our future work also includes a set of optimization and mapping tools to allow behavioral type designs to be directly mapped to our asynchronous PGA structures and then evaluating side-channel resistance of our design.

References

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annu. Int. Cryptol. Conf.* Springer, 1999, pp. 388–397.

[2] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, "Improving smart card security using self-timed circuits," in *Proc. 8th Int. Symp. Asynchronous Circuits Syst.*, Silver Spring, MD, USA, Apr. 2002, pp. 211–218.

[3] J. J. Fournier, S. Moore, H. Li, R. Mullins, and G. Taylor, "Security evaluation of asynchronous circuits," in *Cryptographic Hardware and Embedded Systems–(CHES)*, vol. 2779. Berlin, Germany: Springer, 2003, pp. 137–151, doi: 10.1007/978-3-540-45238-6_12.

[4] K.-S. Chong *et al.*, "Side-channel-attack resistant dual-rail asynchronous-logic AES accelerator based on standard library cells," in *Proc. Asian Hardw. Oriented Secur. Trust Symp. (AsianHOST)*, Dec. 2019, pp. 1–7.

[5] J. M. Emmert, A. Perumalla, and L. Concha, "An asynchronous FPGA THx2 programmable cell for mitigating side-channel attacks," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 840–843.

[6] J. M. Emmert and A. Perumalla, "An asynchronous MPGA THx2 cell and architecture for mitigating side-channel attacks," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON)*, Jul. 2019, pp. 232–235.

[7] J. M. Emmert, "Systems and methods for asynchronous programmable gate array devices," U.S. Patent Appl. 2020033681 A1, Dec. 2020.

[8] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan./Feb. 2010.

[9] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proc. IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug. 2014, doi: 10.1109/JPROC.2014.2334493.

[10] L. Lin, W. Burleson, and C. Parr, "MOLES: Malicious off-chip leakage enabled by side-channels," in *IEEE/ACM Int. Conf. Comput.-Aided Design-Dig. Tech. Papers*, Nov. 2009, pp. 117–122.

[11] R. Sridhar, "Asynchronous design techniques," in *Proc. 5th Annu. IEEE Int. ASIC Conf. Exhibit*, Sep. 1992, pp. 296–300.

[12] K. Fant and S. Brandt, "NULL Convention Logic: A complete and consistent logic for asynchronous digital circuit synthesis," in *Proc. Int. Conf. Appl. Specific Syst., Archit. Processors*, Aug. 1996, pp. 261–273.

[13] S. Hauck, "Asynchronous design methodologies: An overview," *Proc. IEEE*, vol. 83, no. 1, pp. 69–93, Jan. 1995, doi: 10.1109/5.362752.

[14] S. M. Nowick and M. Singh, "Asynchronous design—Part 1: Overview and recent advances," *IEEE Des. Test*, vol. 32, no. 3, pp. 5–18, Jun. 2015, doi: 10.1109/MDAT.2015.2413759.

[15] S. M. Nowick and M. Singh, "Asynchronous design—Part 2: Systems and methodologies," *IEEE Des. Test*, vol. 32, no. 3, pp. 19–28, Jun. 2015, doi: 10.1109/MDAT.2015.2413757.

[16] M. Alain, "The design of an asynchronous microprocessor," Tech. Rep., 1989.

[17] M. Ligthart, K. Fant, R. Smith, A. Taubin, and A. Kondratyev, "Asynchronous design using commercial HDL synthesis tools," in *Proc. 6th Int. Symp. Adv. Res. Asynchronous Circuits Syst. (ASYNC)*, Apr. 2000, pp. 114–125, doi: 10.1109/ASYNC.2000.836983.

[18] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL convention self-timed circuits," *Integration*, vol. 37, no. 3, pp. 135–165, 2004, doi: 10.1016/j.vlsi.2003.12.004.

[19] C. Abraham and S. Chiao, "The FPGA to MPGA ASIC conversion process," in *Proc. 40th Midwest Symp. Circuits Syst. Dedicated Memory Professor Mac Van Valkenburg*, vol. 2, Aug. 1997, pp. 1426–1429, doi: 10.1109/MWSCAS.1997.662351.

[20] D. Marple and L. Cooke, "An MPGA compatible FPGA architecture," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1992, p. 4, doi: 10.1109/CICC.1992.591107.

[21] L. Pileggi *et al.*, "Exploring regular fabrics to optimize the performance-cost trade-off," in *Proc. 40th Conf. Design Autom. (DAC)*, 2003, pp. 782–787.

[22] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi, "Efficient and secure intellectual property (IP) design with split fabrication," in *Proc. IEEE Int. Symp. Hardware-Oriented Secur. Trust (HOST)*, May 2014, pp. 13–18, doi: 10.1109/HST.2014.6855561.

[23] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2013, pp. 1259–1264, doi: 10.7873/DATE.2013.261.

[24] K. Xiao, D. Forte, and M. M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2015, pp. 14–19, doi: 10.1109/HST.2015.7140229.

[25] S. Trimberger, *Field Programmable Gate Array Technology*. Springer, 1994.

[26] (2021). [Online]. Available: https://www.xilinx.com

[27] (2021). [Online]. Available: https://www.altera.com

[28] J. Rose *et al.*, "Architecture of field-programmable gate arrays," *Proc. IEEE*, vol. 81, no. 7, pp. 1013–1029, Jul. 1993.

[29] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Des. Test Comput.*, vol. 17, no. 3, pp. 44–53, Jul./Sep. 2000.

[30] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 1989, pp. 1929–1934.

[31] S. Smith and J. Di, *Designing Asynchronous Circuits Using NULL Convention Logic (NCL)*. Morgan & Claypool, 2009.

[32] *NCSU Cadence Design Kit*. [Online]. Available: https://www.eda.ncsu.edu/wiki/NCSU_CDK

[33] *NCSU FreePDK45*. [Online]. Available: https://eda.ncsu.edu/freepdk/freepdk45/

[34] J. Teifel and R. Manohar, "Highly pipelined asynchronous FPGAs," in *Proc. ACM/SIGDA 12th Int. Symp. Field Program. Gate Arrays (FPGA)*, New York, NY, USA, 2004, pp. 133–142, doi: 10.1145/968280.968300.

[35] C. Pham-Quoc and A.-V. Dinh-Duc, "New approaches to design asynchronous circuits on FPGAs," in *Proc. Int. Conf. Adv. Technol. Commun.*, Oct. 2009, pp. 63–67, doi: 10.1109/ATC.2009.5349341.

[36] Q. T. Ho, J. Rigaud, L. Fesquet, M. Renaudin, and R. Rolland, "Implementing asynchronous circuits on LUT based FPGAs," in *Proc. 12th Int. Conf. Field Program. Logic Appl.* Berlin, Germany: Springer-Verlag, 2002, pp. 36–46.

[37] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007, doi: 10.1109/TCAD.2006.884574.

[38] *OpenCores*. [Online]. Available: https://opencores.org/

[39] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Proc. 7th Int. Workshop Field Program. Logic Appl. (FPL)*, Berlin, Germany: Springer-Verlag, 1997, pp. 213–222.

[40] S. Ataei *et al.*, "An open-source EDA flow for asynchronous logic," *IEEE Des. Test*, vol. 38, no. 2, pp. 27–37, Apr. 2021, doi: 10.1109/MDAT.2021.3051334.

[41] R. Manohar, "An open source design flow for asynchronous circuits," in *Proc. Government Microcircuit Appl. Crit. Technol. Conf. (GOMACTech)*, Mar. 2019.

John M. (Marty) Emmert (Senior Member, IEEE) was born in Lexington, KY, USA. He received the B.Sc. degree in electrical engineering from the University of Kentucky, the M.Sc. degree in electrical engineering from the Air Force Institute of Technology, and the Ph.D. degree in computer science and engineering from the University of Cincinnati. He is currently a Professor with the Department of Electrical Engineering and Computer Science, University of Cincinnati, and the Director of the NSF Center for Hardware and Embedded Systems Security and Trust (CHEST) I/UCRC. He is also a Graduate of the Air War College and a retired Colonel from the U.S. Air Force Reserves.

Anvesh K. Perumalla (Student Member, IEEE) received the M.Sc. degree in electrical engineering from Wright State University and the Ph.D. degree in computer engineering from the University of Cincinnati. He is currently a Post-Doctoral Researcher with the Department of Electrical Engineering and Computer Science, University of Cincinnati. His current research focus is on hardware security topics, such as physically unclonable functions, counterfeit IC detection, FPGA reverse engineering, and asynchronous circuit design methodologies.

Tristan J. Hudson (Student Member, IEEE) received the M.Sc. degree in computer engineering from the University of Cincinnati, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science. His research focus on FPGAs, hardware security and trust, and asynchronous digital design.

Luis M. Concha was born in Dayton, OH, USA. He received the B.Sc. and M.Sc. degrees in electrical engineering from Wright State University. He is currently the Managing Director of the NSF Center for Hardware and Embedded Systems Security and Trust (CHEST) I/UCRC. Before that, he served for 32 years as an Air Force Civil Servant in a wide range of engineering and leadership roles at the Air Force Research Laboratory and the Air Force Life Cycle Management Center. He is a Graduate of the Air War College and an Excellence in Government Senior Fellow. He has been awarded the Air Force Outstanding Civilian Career Medal.