Datapath Cells for Null Convention Logic Asynchronous Circuit Area Reduction

Dallas A. Phillips
Dept. of EECS
University of Cincinnati
Cincinnati, OH, USA
phillda@mail.edu

Pingxiuqi Chen
Dept. of EECS
University of Cincinnati
Cincinnati, OH, USA
chenp5@mail.edu

John M. Emmert
Dept. of EECS
University of Cincinnati
Cincinnati, OH, USA
john.emmert@uc.edu

***Abstract*—By desynchronizing digital data processing over time, asynchronous circuits provide strong protection against malicious side channel attacks. Asynchronous circuits also mitigate digital substrate noise and thereby significantly improve signal-to-noise ratios for sensitive analog and radio frequency components for System-on-a-Chip applications. One main drawback to asynchronous circuits is the significant (often more than double) area required for their implementation. To improve community acceptance of asynchronous design paradigms, we propose a new set of datapath cells to significantly reduce the area of asynchronous digital circuits. Our datapath cells have both asynchronous and standard combinational input and output capability. They serve as the basis for a design methodology that converts standard synchronous sequential netlists into asynchronous netlists, and they allow the user to tune or set the level of asynchronous processing. Our approach uses standard, commercial-off-the-shelf integrated circuit synthesis tools, and thus, it does not require any special asynchronous synthesis tools.**

Keywords—asynchronous, null convention logic, systems-on-a-chip, noise reduction, NCL, obfuscation, security, assurance, trust

I. INTRODUCTION

Most fabricated designs are synchronous in nature and require a global clock signal to simultaneously operate. Synchronized power draws of standard circuits makes them very vulnerable to side channel attacks that can expose sensitive data [1]. This leads to methods such as Simple Power Analysis (SPA) and Differential Power Analysis (DPA) to reverse engineer the functionality of the circuit. This global clock signal can also add unwanted power, and it can lead to large inefficiencies.

Asynchronous circuits, which are circuits that do not require a global clock signal to operate, are naturally more secure and allow designers to implement the same circuit functionality without exposing their design easily. One way of implementing an asynchronous circuit is the use of null convention logic (NCL) gates, which according to [2] is a *symbolically complete logic which expresses process completely in terms of the logic itself and inherently and conveniently expresses asynchronous digital circuits*. Designing with NCL gates gives a design more security and assurance due to the dual rail design. NCL gates have two separate wires, representing logic '*0*' and '*1*' respectively. Fant and Brandt showed that with asynchronous registers and NCL gates, a complete delay-insensitive design can be constructed. [2]. Some technological companies have started more research and designing with asynchronous designing, such as Intel and IBM as shown by Nowick and Singh with the introduction of globally asynchronous locally synchronous (GALS) circuits as well as Networks-on-Chip (NoC) architectures [3]. One major drawback to asynchronous designs is due to the large amount of area required (at least twice as much compared to its synchronous equivalent) as well as few modern designers being trained to design using asynchronous techniques. We propose the possibility of combining the assurance and trust that asynchronous circuits provide with the optimized area of synchronous circuits while also being able to reduce any unwanted power during execution.

We propose a new altered design to these NCL gates, that can take both synchronous and asynchronous inputs and perform Boolean functionality in the same manner that would be done in either fully synchronous or fully asynchronous designs. These designs keep the return to zero (RTZ) characteristic that is seen in asynchronous circuits, which allows these new gates to keep the same security benefits as asynchronous circuits. This RTZ design means that the circuit will not execute until the asynchronous inputs have been set, regardless of whether or not the inputs arrive at the gate synchronously. These so called datapath cells can be inserted into a current design directly replacing the timing critical path or paths to add assurance to the design without compromising significant area.

These gates were designed using techniques from both synchronous and asynchronous methods as well simplifications using simple Boolean Algebra minimization techniques. Their designs add no extra delay to the circuit while adding in the security benefits such that their executions are comparable to an only synchronous equivalent design. Future implementation will take a netlist and add these datapath gates into the critical path or paths of a design autonomously such that designers will be able to implement these gates and the security assurance that comes with them without having to be trained in designing asynchronously or giving up significant area requirements for fully asynchronous designs.

The rest of this paper is set up as follows. Section II describes some relevant previous work in the field that will help to build the reasoning behind the designs of these datapath gates. Section III describes the datapath gate design flow as well as some examples. Section IV explains the simulation

This work was supported by the NSF Center for Hardware and Embedded Systems Security and Trust under Grant 1916722.

results, and Section V features some conclusions as well as future work with datapath gates.

II. Background

The relevant background material for this project comes from efforts in the areas of asynchronous circuits, specifically NCL gates and exploring their security benefits, as well as different methods to convert some portion or the entirety of a synchronous netlist to a functionally equivalent asynchronous circuit.

A. Security Benefits of Asynchronous Circuits and NCL Gates

The security benefits associated with asynchronous circuits have long been known by academia with publications such as [4]. Fournier, Moore, Li, Mullins and Taylor proved with an early asynchronous smart card design that was tested using DPA. In a synchronous circuit, the CMOS transistors leak power which can lead to obfuscation of sensitive data that can be recovered through DPA or Electro-magnetic emittance for example.

B. NCL Gates

NCL gates are specifically good at mitigating these potential data-leaking properties to prevent SCAs [5]. Since synchronous circuits are all clocked at a simultaneous instant, DPA is relatively easier since the power required for the entire system is drawn at that moment. With NCL gates being a dual-rail design, this allows for higher security impacts of the system due to the relatively constant power draw resulting in similar hamming weights as was shown by Wu. As Brandt and Fant proved, a downside to NCL gates, despite all these security benefits, is due to the area consumption compared to a synchronous equivalent circuit. With the security advantages that come with NCL gates, one might wonder why designers do not fully implement asynchronous systems, which would come with a greater assurance and trust that data leakage is less likely. However, asynchronous circuits at minimum double the area which may not be an optimal solution for most designs, as well as the designers not being trained in asynchronous designing techniques.

C. Converting Synchronous Circuits to Asynchronous Circuits

Possible solutions to implement asynchronous circuit from synchronous designs have been explored, such as [6] [7] [8]. Methods like those from Brej as well as Branover, Kol and Ginosar, and Semba and Saito allow current designers to apply asynchronous techniques and advantages, without having to specifically design using asynchronous methods. These applications have been created to convert synchronous netlists to asynchronous circuits for the purpose of adding security to a design. Semba and Saito compared the conversion on both the RT-level as well as Gate-level. The results showed that RTL conversion was better for energy consumption on average about 11.3%. This means that all designs for our project will convert all designs to RTL if possible, based off this finding.

Branover, Kol and Ginosar found that area growth for a synchronous circuit can be up to over 200 percent, with more area growth occurring for smaller synchronous designs. They also use Synopsys to synthesize a synchronous circuit then convert post-synthetization. This is important because it allows for an easier transition for traditional designs and current synthetization tools to remain relevant rather than having to redesign a new software tool specifically for asynchronous designs. Our project will also operate on a synthesized netlist then convert a portion of that design into datapath gates according to the critical path(s) as well as a value determined by the designer, which will be discussed later.

The results from Brej, Branover and Kol and Ginosar, and Semba and Saito all focus on converting the entire synchronous netlist into an asynchronous one, which as Branover, Kol and Ginosar proved can lead to huge area requirements. However, there have been efforts to convert a smaller portion of the entire synchronous netlist (based on a timing critical path) [9]. Here, Park and Kim introduce the idea of converting only critical paths to dual rail gates to match existing standard cell library components instead of implementing the asynchronous portions with dynamic logic gates which is the traditional method of conversion. Our work will not only convert a critical path or paths, which will lower the required power and area needed while adding security, but will also map our new gate designs to a library which can be resynthesized to standard cells. Also, our project will allow designers to control what percentage of the circuit they want to convert, which will give more control to the designer on how much area, speed and power will be consumed with certain implementations of their design.

III. Datapath Gate Design Flow

A. Datapath Gate Design Methodology

The design method for our data path gates follows a simple five step process as detailed in Table 1, which shows the evolving Boolean equations that represent the pull-down networks (PDNs) and pull-up networks (PUNs) of the CMOS implementations described in two of our designs, for a two-input AND gate and two-input OR gate equivalent (TH22 and TH12, respectively). In these examples, we treat the *A* signal as an asynchronous input, and *B* as a synchronous input. Also, a key design requirement is that the gates should only execute once a value on *A* has been set.

To design a datapath gate, start with the Boolean equation for the function that needs to be converted. Step 1 requires changing any minterm in the Boolean equation with an *A* to *A1*. This will serve as the logic high wire in the datapath gate. Any minterm that does not have an instance of *A* needs to be AND with *A0*. This is a direct result of the requirement of having the gate execute when the asynchronous input has been set which is important considering the arrival timings could vary with synchronous vs asynchronous inputs. Adding the *A0* to any non-*A* term will ensure that the function will execute only when the asynchronous input *A* has been set. Step 1

forms the equation on which the PDN of our logic high implementation for our datapath gate will be based.

Step 2 takes the resulting equation that was formed in Step 1 and changes every AND to OR and vice versa. Note, this is different than inverting the equation because the logic of the signals (inputs) themselves has not changed, only the logical relationship between the signals (inputs). This forms the PUN of our datapath gate for the logic high implementation.

In Steps 3 and 4, we take the equation from Step 1 and fully invert it. After inversion, we change $\overline{A1}$ to $A0$ and $\overline{A0}$ to $A1$. This is functionally the same thing, except in asynchronous designs there exists a RTZ which means that $\overline{A1}$ can happen during both the set or reset phases. To stay consistent with only executing during the set phase of the asynchronous input, these are converted. The resulting equation in Step 4 forms the PDN of the logic low implementation for our datapath gate.

Finally, Step 5 is similar to Step 2, where we change AND to OR and vice versa, except this time we start with the resulting equation from Step 4. This forms the PUN for the logic low implementation. After these simple 5 steps have been completed, a CMOS gate can be designed similar to the ones seen in Figures 1a and 1b.

Designs have been produced for all the NCL gate equivalents according to Figure 1. For gates which have weighted inputs, such as TH24w2, the datapath gate design differs based on which input is treated as asynchronous, so weighted gates will have several implementations to consider for conversion.

TABLE 1. FIVE STEP PROCESS TO CONVERT BOOLEAN EQUATION TO DATAPATH LOGIC.

	2-input AND (TH22)	2-input OR (TH12)
Boolean Eq.	A * B	A + B
Step 1 (F1, PDN)	A1 * B	A1 + A0*B
Step 2 (F1, PUN)	A1 + B	A1 * (A0 +B)
Step 3	$\overline{A1}$ + $\overline{B}$	$\overline{A1}$ *($\overline{A0 + B}$)
Step 4 (F0, PDN)	A0 + $\overline{B}$	A0 * (A1 + $\overline{B}$)
Step 5 (F0, PUN)	A0 * $\overline{B}$	A0 + A1* $\overline{B}$

NCL Gate	Boolean Function	Transistor Count (static)	Transistor Count (semi-static)
TH12	$A + B$	6	6
TH22	AB	12	8
TH13	$A + B + C$	8	8
TH23	$AB + AC + BC$	18	12
TH33	ABC	16	10
TH23w2	$A + BC$	14	10
TH33w2	$AB + AC$	14	10
TH14	$A + B + C + D$	10	10
TH24	$AB + AC + AD + BC + BD + CD$	26	16
TH34	$ABC + ABD + ACD + BCD$	24	16
TH44	$ABCD$	20	12
TH24w2	$A + BC + BD + CD$	20	14
TH34w2	$AB + AC + AD + BCD$	22	15
TH44w2	$ABC + ABD + ACD$	23	15
TH34w3	$A + BCD$	18	12
TH44w3	$AB + AC + AD$	16	12
TH24w22	$A + B + CD$	16	12
TH34w22	$AB + AC + AD + BC + BD$	22	14
TH44w22	$AB + ACD + BCD$	22	14
TH54w22	$ABC + ABD$	18	12
TH34w32	$A + BC + BD$	17	12
TH54w32	$AB + ACD$	20	12
TH44w322	$AB + AC + AD + BC$	20	14
TH54w322	$AB + AC + BCD$	21	14
THxor0	$AB + CD$	20	12
THand0	$AB + BC + AD$	19	13
TH24comp	$AC + BC + AD + BD$	18	12

Figure 1. List of NCL gates, their Boolean functions, and number of transistors required to implement.

B. *Critical Path Selection/Insertion Techniques*

In order to convert a design, a critical path has to be identified such that a smaller portion of the design is asynchronous rather than converting one hundred percent of the circuit which is area inefficient. Park and Kim identify timing critical paths by starting at each primary output of the circuit and then use a backtracking algorithm to ensure that the longest timing critical paths are extracted. They also declare any sub-path in the critical path as also critical.

Pipeline based designs have also been converted based on a critical path [10]. The timing critical path is determined again by the slowest path in a pipeline stage and use synchronizing logic gates instead of conventional logic gates to solve the data-dependency problem of logic gates. This means that the critical paths between two consecutive pipeline stages could differ because of different input patterns or paths being early triggered from outputs of the previous pipeline stage.

Even with the possibility of converting any function from synchronous to asynchronous, paths that pass through both synchronous and asynchronous gates need to be carefully considered to ensure that functionality is not lost on any transitional path between gates. When transitioning from the output of an asynchronous gate to the input of a synchronous gate, the logic '*1*' signal from the output can be directly placed in the input of the synchronous gate since it represents the entire functionality in all cases and does not add any extra parasitics to convert. The transition from the output of a synchronous gate to the input of an asynchronous gate is slightly more complex. According to Park and Kim all sub-paths of critical paths are also considered critical, which means in the backtracking algorithm for a small combinational circuit this could end up converting the entire or at least a significant portion of the design, which is not ideal in terms of area conservation.

Since the NULL wave does not translate well to a synchronous signal, converting from the output of a synchronous gate to an asynchronous input is much more

complex.. In the case of Park and Kim, they converted the memory components to asynchronous using the handshaking method, which means that this special case of conversion was avoided. However, for this project we are only dealing with the combinational circuit conversion. One possible solution resembles the design of a D-type flip-flop where the input signal, *D*, is branched off into two inputs internal to the flip flop. One of those branches is the direct input, and the other is the inverted input before the actual latch or memory implementation starts. Branching off a synchronous input into an inverted and non-inverted branch to represent the logic '*1*' and logic '*0*' wires in an asynchronous circuit would ensure that both wires are not the same value simultaneously, particularly both set to logic HIGH, which is not possible in asynchronous gates. However, adding the inverter to one of the branches will add slight delay to the circuit, which means for a transitioning input value, there exists a very minute period where both wires are the same value. This will have to be taken into consideration to ensure that incorrect output values are not recorded during this transitioning period.

C. Gate Sizing

An important step to designing the datapath gates was determining the appropriate sizes of each transistor. Incorrect sizes could lead to loss of functionality due to a transistor not being able to set the proper output value. Note that only widths of the transistors were altered, all lengths were kept at the same size. According to the example transistor diagrams in Figure 2, the reversely oriented inverters, *G2*, were set at minimal size for all datapath gates. The forward-oriented inverters, *G1*, were doubled relative to a minimally sized inverter. This was also kept constant for every datapath gate design. The PUN and PDN of every design were sized according to the longest existing path in the design either from *VDD* or *GND* to the input of the forward oriented inverter. For example, in Figures 2c and 2d, for the OR gate there exists a path of two transistors (either *A0* and *B* in the PDN of Figure 2d, or *A1* and *A0/B* in the PUN of Figure 2c). In this case, since the longest path is two transistors, the widths of all transistors in both the PDNs and PUNs were doubled relative to the forward oriented inverter (4x the width of a minimally sized transistor).

IV. Simulation Results

Once the datapath gates were designed using our five-step process, they were constructed and simulated using H-Spice, as well as WaveView from Synopsys to visualize the simulation results. Two example simulations are shown in Figure 3. These simulations represent the datapath equivalent of a 2-input AND (Figure 3a) and 2-input OR (Figure 3b) where in both cases signal *A* is treated as the asynchronous input to the gate and *B* is the synchronous signal. The waveforms are set up as follows: the top (green) signal is *A0* input, the second (yellow) is the *A1* input. The third (light blue) is the synchronous *B* signal, and finally the fourth (red) and fifth (purple) signals are the logic high and logic low outputs, respectively. These simulations show full functionality for an exhaustively tested design. The key result is that outputs are never set during a NULL wave (i.e. *A1* and *A0* are both logic low) as well as both outputs, logic high and logic low, are never set high simultaneously. This result is critical in preserving the functionality of the circuit while keeping the RTZ characteristic of asynchronous circuits.

V. Conclusions

This work has shown an alternative method to designing NCL gates which are able to function in hybrid circuits. These datapath gates through simulation results from HSpice proved to be delay-insensitive (DI), meaning the unsynchronized arrivals of inputs will not affect the functionality of the gate itself. We have shown a design methodology for implementing these datapath gates based on combinational Boolean logic equations. These datapath gates can be used to convert a portion of a synchronous netlist, based on a critical path or paths, such that area overhead is kept as minimal as possible, while maintaining Boolean functionality as well as introducing the security advantages that NCL gates provide.

References

[1] P. Kocher, J. Jaffe, and B. Jun, "Introduction to Differential Power Analysis and Related Attacks," 1998.

[2] K. M. Fant and S. A. Brandt, "NULL Convention Logic/sup TM/: a complete and consistent logic for asynchronous digital circuit synthesis," Proceedings of International Conference on Application Specific Systems, Architectures and Processors: ASAP '96, 1996, pp. 261-273.

[3] S. M. Nowick and M. Singh, "Asynchronous Design—Part 1: Overview

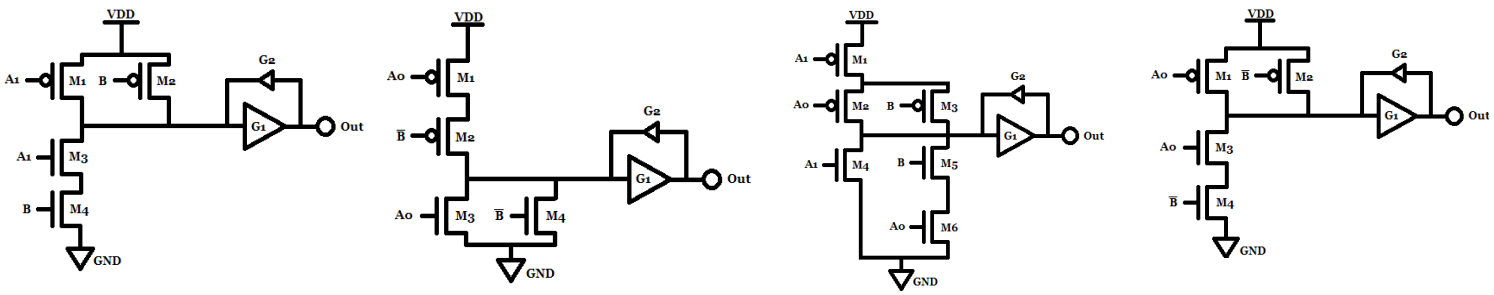
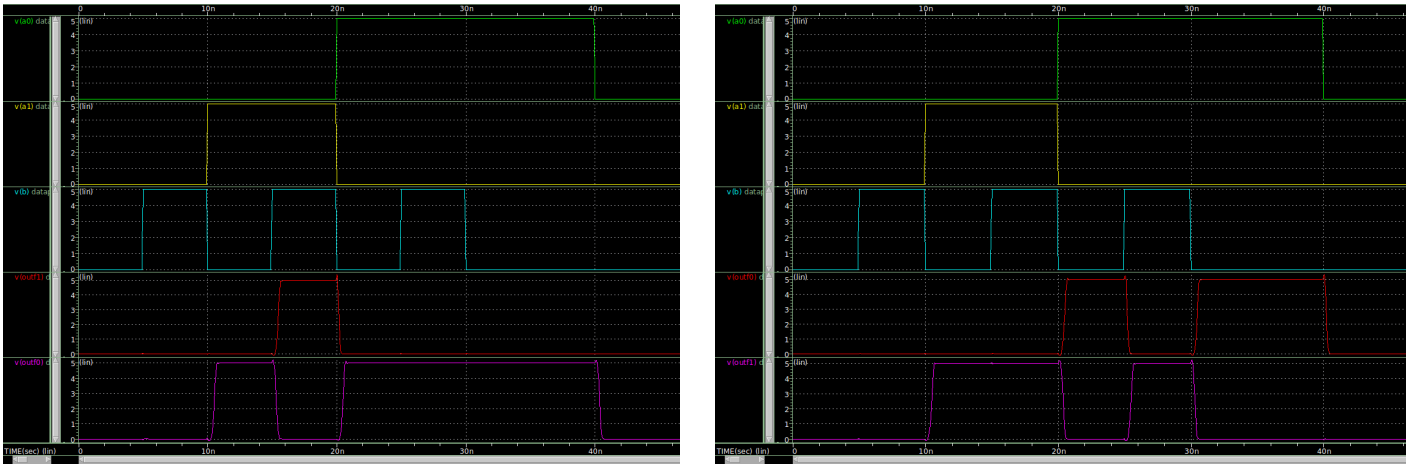


Figure 2: CMOS Designs for Datapath Gates (TH22 and TH12)



a) b)

Figure 3: HSpice waveform simulation results via WaveView for a TH22 (a) and TH12 (b) equivalent datapath gate.

and Recent Advances," in IEEE Design & Test, vol. 32, no. 3, pp. 5-18, June 2015

[4] Fournier J.J.A., Moore S., Li H., Mullins R., Taylor G. (2003) Security Evaluation of Asynchronous Circuits. In: Walter C.D., Koç Ç.K., Paar C. (eds) Cryptographic Hardware and Embedded Systems - CHES 2003. CHES 2003. Lecture Notes in Computer Science, vol 2779. Springer, Berlin, Heidelberg.

[5] Wu, Jun, "Null Convention Logic applications of asynchronous design in nanotechnology and cryptographic security" (2012). Doctoral Dissertations. 1971.

[6] C.F. Brej, "An automatic synchronous to asynchronous circuit convertor", 11th UK Asynchronous Forum, 2001.

[7] A. Branover, R. Kol and R. Ginosar, "Asynchronous design by conversion: converting synchronous circuits into asynchronous ones," Proceedings Design, Automation and Test in Europe Conference and Exhibition, 2004, pp. 870-875 Vol.2.

[8] S. Semba and H. Saito, "Comparison of RTL Conversion and GL Conversion from Synchronous Circuits to Asynchronous Circuits," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, pp. 1-4

[9] H. Park and T. Kim, "Synthesizing Asynchronous Circuits toward Practical Use," 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2016, pp. 47-52

[10] Z. Xia, S. Ishihara, M. Hariyama and M. Kameyama, "Dual-rail/single-rail hybrid logic design for high-performance asynchronous circuit," 2012 IEEE International Symposium on Circuits and Systems (ISCAS), 2012, pp. 3017-3020