Area Efficient Asynchronous Circuits for Side Channel Attack Mitigation

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

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

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

2022 IEEE 40th International Conference on Computer Design (ICCD) | 978-1-6654-6186-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCD56317.2022.00089

***Abstract*—Synchronous sequential or clocked digital circuits are susceptible to synchronized side channel attacks (SCAs). By distributing (in time) data processing, one method used to mitigate or defend against SCAs is clockless, asynchronous circuit design. A problem often associated with clockless, asynchronous circuit design methods, like Null Convention Logic (NCL), is the large area for logically equivalent circuits. Typical asynchronous circuits are 2.5 to 3.5x the size of their synchronous sequential counterparts. This work develops a data-path design methodology based on a library of unique hybrid cells (part conventional and part NCL). The new hybrid method has shown a significant reduction in transistor count for asynchronous circuits. It results in logically equivalent asynchronous circuits with only an average transistor count increase of 6% while maintaining distributed (in time) processing advantages. The method, hybrid gate description, and comparison for several benchmark circuits are presented.**

Keywords—asynchronous, data-path, null convention logic, NCL, side channel attacks, SCAs, systems-on-a-chip, SOC

I. Introduction

Side channel attacks (SCAs) are leveraged to indirectly leak private or sensitive information from integrated circuits (ICs). Clocked, synchronous sequential circuits are especially susceptible due to predictable or timed data processing [1]. In addition, susceptibility to SCAs can be enhanced through the addition of *Trojan* circuitry added by untrusted agents or entities during the fabrication process [2][3]. Whether Trojan circuitry is present or not, information leakage is highly dependent on timing and clocked data processing.

Most fabricated designs are synchronous in nature and often require a global clock signal so that data simultaneously arrives at the inputs of combinational processing blocks. Whether a Trojan circuit is present or not, the synchronized power draw of the (often) large combinational processing blocks is the primary contributor to successful SCAs. Asynchronous circuits, which can be completely clockless, provide one approach to mitigating power based SCAs [4]. One way to implement asynchronous circuits is through null convention logic (NCL) gates. Null Convention Logic is a symbolically complete logic which expresses processes completely in terms of the logic itself and inherently and conveniently expresses asynchronous digital circuits [5][6]. Null Convention Logic gates are dual-rail with two separate wires for each signal. One wire represents logic '*0*' and the other logic '*1*.' Fant and Brandt showed that with asynchronous registers and NCL logic gates, a complete delay-insensitive design can be constructed [6]. One big drawback of NCL asynchronous designs is the large area overhead required for logically equivalent asynchronous circuit implementations. Most fully asynchronous equivalent NCL circuits are 2.5 to 3.5x the size of their standard counterparts. In addition, most designers are not trained to design NCL asynchronous circuits. This work presents a hybrid approach to quickly implement asynchronous circuits using standard synchronous sequential designs. The hybrid approach does not require expertise in asynchronous or NCL design, and it has shown significant area reduction for equivalent NCL designs.

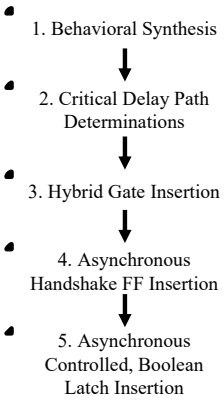

Fig. 1. Flow chart for reduced sized asynchronous NCL circuit generation.

As shown in Fig. 1, the approach to generate asynchronous NCL circuits with low area overhead (on average only 6% transistor count increase) has five basic steps: 1) Generate or synthesize a digital circuit composed of standard cell Boolean gates, 2) determine the critical or longest delay path through each synchronized (clocked), combinational subcircuit block, 3) replace Boolean gates in each of the longest delay paths with their hybrid asynchronous equivalent gates, 4) replace flip-flops that drive the asynchronous NCL inputs of the hybrid gates with their NCL asynchronous equivalents, and 5) replace the remaining flip-flops with low overhead latches controlled

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

2576-6996/22/$31.00 ©2022 IEEE
DOI 10.1109/ICCD56317.2022.00089

by NCL asynchronous handshaking. The key to the overall approach is based on determining the longest delay or critical path in the circuit, and then replacing its (and only its) gates with hybrid "conventional to asynchronous NCL" logic gates. The hybrid conventional/NCL logic gates are designed using techniques from both synchronous and asynchronous circuit design, and an important requirement of the hybrid gates is that no additional delay is added to its Boolean inputs. In previous approaches, signal conditioning added additional processing delays to combinational inputs. The additional input delays led to changes in the longest or critical paths, thereby reducing the reliability of the resulting asynchronous circuit. The rest of this paper is organized as follows. Section II describes relevant background to the reasoning behind the hybrid data-path approach. Section III describes determination of and requirements for data-path insertion, section IV describes the hybrid gate insertion, and section V describes the hybrid conventional/NCL logic gates. Section VI evaluates the approach, and section VII concludes.

II. Background

Background in three primary areas is relevant to the work presented: A) SCAs and mitigation through asynchronous circuit design, B) NCL asynchronous circuit design, and C) asynchronous circuit area reduction through data-path cell replacement.

A. Side Channel Attacks

One way to describe a Trojan circuit is *an extra circuit that is added by the manufacturer (unbeknownst to the designer) during the IC fabrication process*. Trojan circuits can be used to provide legitimate feedback to the manufacturer on their manufacturing process; however, they can also be used to indirectly leak secret, sensitive, or private information during regular operation of the IC. Several types of Trojan circuits have been developed that require minimal overhead and are very difficult for designers to detect either during normal IC testing or (low probability) IC reverse engineering [2]. One specific example of a Trojan circuit used for SCAs is a Malicious Off-Chip Leakage Enabled Side-Channel (MOLES) integrated circuit shown in Fig. 2 [3]. The MOLES circuit has very little overhead, which makes it very difficult to detect. The MOLES circuit consists of XOR gates, a pseudorandom number generator (PRNG), and some capacitive loads. The PRNG can either be a single linear shift feedback register or an existing PRNG already located on the IC. The entire MOLES circuit can use just a few hundred out of a multi-million transistor IC, and power consumed by the MOLES circuit shows up as noise during normal IC testing. An untrusted agent can use this extra hardware with a large collection of electromagnetic or transient power readings to decipher secret keys [3]. Power spikes caused by combinational digital switching during every clock cycle, can be collected over long periods of time. Reverse engineering through spread spectrum techniques is used to detect secret keys, $K = k_1k_2k_3...k_n$, stored in IC firmware. Machine learning can accelerate the process, making it so less data is needed to decipher the secret keys.

In general, SCAs make use of indirect measures to exploit either existing or added Trojan circuitry to obtain secret, private, or sensitive information. Trojan circuits are not required, but they can increase the susceptibility of a circuit to SCAs. Side channel attacks can leverage temperature variations, electromagnetic radiation, power usage, or other measurable characteristics during regular IC operation. Asynchronous circuits, that distribute processing (in time), have been shown to reduce the susceptibility of ICs to SCAs [4].

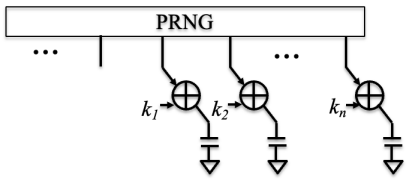

Fig. 2. Trojan MOLES circuit for secret key, $K = k_1k_2k_3…k_n$, detection.

B. Asynchronous Null Convention Logic

There are different types of asynchronous design techniques ranging from locally clocked to clockless, and each type has its own advantages and disadvantages relative to SCAs [5-9]. One type of clockless logic circuit is based on NCL [5][6]. NCL circuits work well for data flow designs because the data flows through the networks in waves. A data wave is only processed when all incoming data is available, making it self-timed. Since data is only processed when available, no timing assumptions are required, and this attribute guarantees data sequencing and correct data arrival at the receiver under varying gate, process, and wire delays. NCL data, logic, and control signals use a multi-rail encoding scheme. One rail represents the logic '1' value and one rail the logic '0' value. For example, a signal *A* has a logic '1' wire, *A*1, and a logic '0' wire, *A*0.

Asynchronous NCL circuits are implemented using threshold gates with hysteresis [6][7]. Threshold gates have two or more inputs and a single output, and most are 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; i.e. when its threshold is met, its output is asserted. The output stays asserted (hysteresis) until all inputs have transitioned back to 'NULL' in the reset phase. Fig. 3 shows a generic TH*mn* gate symbol. It should be noted that besides being **clockless**, NCL asynchronous circuits offer advantages to SCA avoidance that include **distributed** in time (unsynchronized) and **low** power consumption. For implementation purposes, DATA is typically set to Vdd and NULL to Vss.

Asynchronous circuits implemented with NCL TH*mn* gates are particularly good at mitigating potential data leak from ICs and help prevent SCAs [8][9]. Since synchronous circuits are all clocked simultaneously, it is simpler for an untrusted agent to reverse engineer information from indirect measurements taken from the IC. However, Brandt and Fant showed the downside to NCL gates, despite all the security benefits, is the area required to implement an equivalent asynchronous version of a standard circuit [6]. Fig. 4 shows an example 2-input NAND gate and its equivalent NCL implementation. The standard 2-input NAND gate requires four metal oxide semi-conductor (MOS) field effect transistors (FETs) while the logically equivalent asynchronous NCL version requires 14 FETs. In general, equivalent NCL combinational circuits are between 2.5 and 3.5x

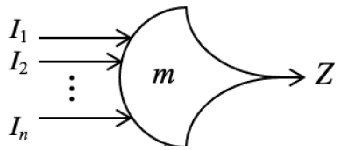

Fig. 3. Example TH*mn* threshold gate symbol with *n* inputs and threshold *m*.

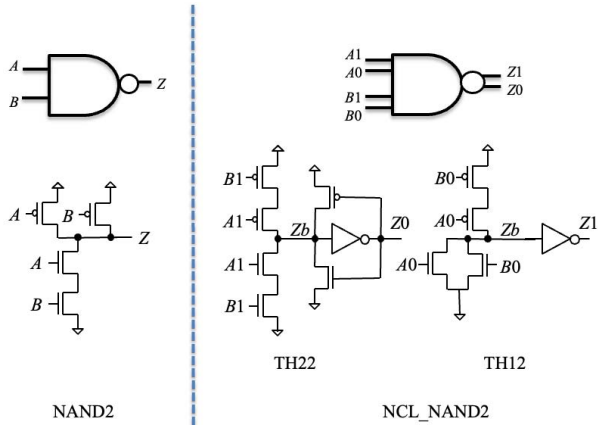

Fig. 4. Example standard and NCL equivalent combinational NAND gate.

the size of their standard logic equivalent circuits. The area increase can be prohibitive due to costs.

C. Conversion: Synchronous to Asynchronous Circuits

Due to area overhead, the cost of asynchronous versions of standard circuits can be prohibitively expensive. Several efforts have sought to reduce the area overhead of asynchronous circuits [10-14]. One approach to create asynchronous circuits is to replace all or part of the gates in the standard circuit with their asynchronous equivalent gates. Methods like those from Brej, Branover, and Semba allow designers to apply asynchronous techniques and advantages without having to specifically design asynchronous circuits [10-12]. These approaches were created to convert synchronous netlists to asynchronous circuits for the purpose of adding security to designs. Semba and Saito compared the conversion on both the register transfer level (RTL) as well as gate-level. The results showed that RTL conversion was on average 11.3% better for energy consumption [11]. However, Branover, Kol, and Ginosar found that area growth for an asynchronous circuit can be over 200% (4x), with more area growth occurring for smaller synchronous designs [12]. They used Synopsys to synthesize a synchronous version. Post-synthesis, they converted it to its asynchronous equivalent. It was significant because it enabled a simple transition from traditional designs and standard synthesis tools rather than having to redesign a new software tool specifically for asynchronous designs.

A popular approach to reduce area was to replace one or more of the standard Boolean circuit's critical or longest paths with their equivalent asynchronous paths. While Brej, Branover, and Semba all converted the entire synchronous netlist into an asynchronous one [10-12], both Park and Xia use the idea of converting only longest timing paths to dual-rail gates by matching existing standard library components to dual-rail equivalent gates [13][14]. Fig. 5 shows an example longest path.

A problem with these techniques results in two possible outcomes, both of which are suboptimal. To interface standard Boolean logic gates with their asynchronous NCL counterparts requires conversion between standard single rail Boolean signals and dual-rail NCL signals. Converting NCL gate output signals to standard Boolean input signals is trivial. Fig. 6 shows an example where the logic '1' NCL output wire, *A*1, is used to connect to a standard Boolean signal, *A*, that can be used as a standard gate input.

On the other hand, the conversion of a Boolean signal to its dual-rail equivalent requires signal conditioning circuitry. In Fig. 6, the B2NCL block asserts a '1' or Vdd value on the *A*1 or *A*0 NCL input wire depending on the logic value of Boolean input signal *A*. While the B2NCL circuitry is not complicated, it does add delay to the arrival time of the input signal driving the standard Boolean logic gate. As shown in Fig. 7, the additional input delay caused by signal conditioning circuitry when NCL gates are used to replace standard Boolean gates in a critical path can result in a modified or different longest critical path.

Two approaches, both suboptimal, were used to address the longest path issue described above. Instead of only replacing the Boolean gates in the longest path with their NCL equivalent gates, all the gates driving any Boolean gate in the longest path were replaced. As shown in Fig. 8, even for a lone longest path, significant numbers of Boolean gates are replaced. Conversion still results in a significant area increase.

The other approach was to modify the new longest path. While this results in lower area overhead, it still requires additional analysis, and due to its iterative nature, is difficult to implement. The hybrid gates, presented below, include all required signal conditioning, and since they don't increase combinational input delay, the only requirement is to replace Boolean gates in the single longest combinational path.

III. Critical or Longest Data-path

To reliably implement the approach for asynchronous circuit generation, it is important to determine the critical or longest delay paths through all combinational subcircuit blocks. This can be accomplished using standard commercial-off-the-shelf synthesis tools that perform static timing analysis, or it can be accomplished using an O(N•log(N)) breadth first search. To make it compatible with most commercially available tools, a tool that 1) parses a structural Verilog netlist, 2) writes it to a hyper-graph, 3) determines the critical delay combinational paths, 4) replaces the standard Boolean gates in the critical path with their hybrid gate equivalents, and 5) writes back out a modified, structural Verilog netlist, is optimal. For the data presented here, commercial tools were used to determine the critical Boolean paths (similar to Fig. 9), and to identify the Boolean cells for replacement by their equivalent hybrid gates.

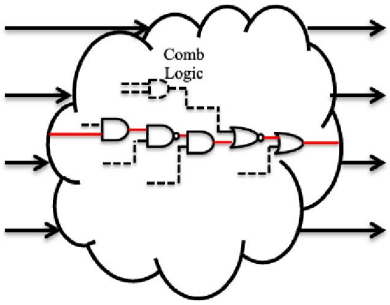

Fig. 5. Example of longest delay combinational data-path.

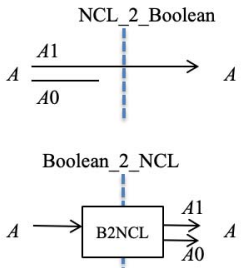

Fig. 6. Dual-rail NCL and Boolean signal conversion.

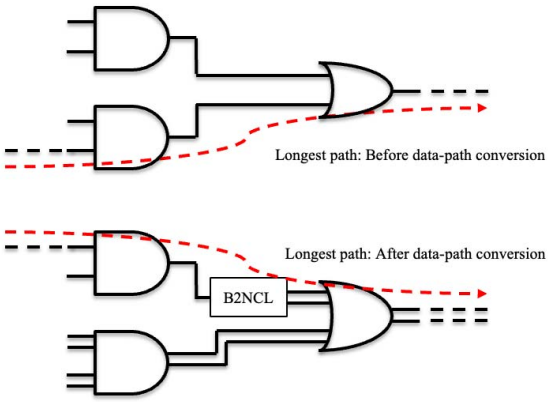

Fig. 7. Example data-path conversion resulting in a *new* longest path.

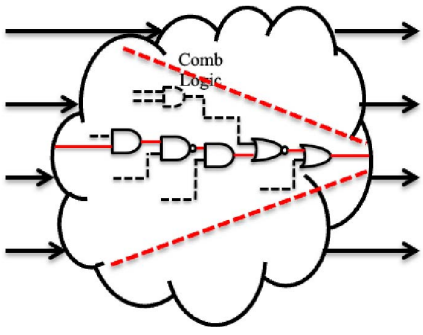

Fig. 8. Data-path replacement of all cells driving any critical path gates.

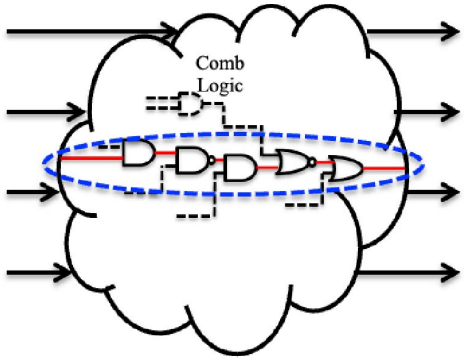

Fig. 9. Hybrid gate replacement of only Boolean critical path cells.

IV. Hybrid Gate Insertion

To attain a significant area reduction for an equivalent asynchronous NCL combinational circuit (on the order of only a 5-10% area increase instead of 2.5 to 3.5x area increase), a minimum number of Boolean combinational gates should be replaced with their, hybrid counterparts. To achieve this, only Boolean gates in the longest or critical delay path of a synchronized combinational block of logic should be replaced with their hybrid counterparts. Given that a static timing analysis tool has determined the list of Boolean gates in the critical or longest delay path, for a reliable asynchronous circuit the following assertion must be made:

- Replacing Boolean logic gates in the critical path with hybrid equivalent gates must not change the critical path.

A result of this assertion is:

- All non-critical path (standard Boolean) signal values arrive at critical path hybrid gate inputs before the critical path asynchronous NCL signals.

The hybrid gates have two types of inputs: Standard Boolean and a dual-rail asynchronous NCL inputs. Since the hybrid gate is creating an asynchronous path through the combinational block, it only has a dual-rail asynchronous output. Except for rare cases, like certain combinational block inputs, the hybrid gate is only required to have a single dual-rail NCL input, and the rest of the inputs will be standard Boolean. To make sure the assertion is adhered to, the hybrid gates must require no external signal conditioning on its inputs and hybrid gate design must guarantee signal propagation delay through the hybrid gate is ≥ the delay through the Boolean gate it is replacing. This is usually not an issue since the hybrid gates have at least two levels of transistor delay and additional delay can be added by carefully sizing the transistors, but it should be verified before fabrication or implementation.

V. Hybrid Gates

Data-path solutions to asynchronous area reduction have been investigated in the past [10-12]. The drawback has been the added input delays associated with signal conditioning. To address this delay, signal conditioning has been included in the

new hybrid cells themselves. A hybrid gate can be directly inserted into a critical path. Its single, dual-rail NCL input directly interfaces to the NCL output of the previous hybrid gate in the critical delay path (if it is the first gate in the path, its NCL input is fed by an NCL register cell). Its other (non-NCL) Boolean inputs are driven by the outputs of non-critical path Boolean gates. The single NCL output of the hybrid gate drives the NCL input of the next hybrid gate in the critical delay path.

To describe the hybrid gate design process, first compare a standard Boolean gate implementation (Fig. 10) to a fully NCL gate implementation (Fig. 11). The Boolean OR2 gate in Fig. 10 has two Boolean input signals, *A* and *B*, and a single Boolean output signal, *Z*. Each signal in Fig. 10 can have a Boolean value of logic '1' or '0.' The NCL OR2 gate in Fig. 11 has two dual-rail input signals, *A* (*A*1 and *A*0 wires) and *B* (*B*1 and *B*0 wires), and one dual-rail output signal, *Z* (*Z*1 and *Z*0 wires). Each dual-rail signal has two wires, a logic '1' and logic '0,' which must be driven by two sub-circuits. The biggest difference for the NCL OR2 in Fig. 11 is that a DATA (Vdd) value applied to either wire (logic '1' wire or '0' wire) asserts its logic value. In other words, a Vdd applied to the logic '1' wire implies the NCL signal has a logic '1' on it, and a Vdd applied to the logic '0' wire implies a logic '0' value on the NCL signal. A hybrid gate that must support both single rail Boolean inputs and asynchronous dual-rail NCL input and output. The general design flow for hybrid gate Boolean inputs is shown in Fig 12.

The difficult part of the design of a hybrid gate is handling a Boolean logic '0' input. Since the hybrid gate output is a dual-rail NCL output signal, a logic '0' value is represented by a Vdd voltage level on the logic '0' wire. So, hybrid gates need to process Boolean logic '0' inputs and generate Vdd voltage levels on NCL logic '0' output wires. Traditional CMOS gate design does not work because it requires inverting logic '0' Boolean signal values, and the extra inversion can change the circuit critical path. To handle this, unconventional, weak transistor design is leveraged. Fig. 13 shows an example OR2 hybrid gate designed using the flowchart in Fig. 12. Other hybrid gates can be designed using this generic approach.

For the OR2 hybrid gate in Fig. 13, there are three critical cases to analyze: A = '1' $\Rightarrow$ Z = '1,' B = '1' $\Rightarrow$ Z = '1,' and A = B = '0' $\Rightarrow$ Z = '0.' It is assumed that the A input is the dual-rail, asynchronous NCL input, B is a Boolean combinational input, and Z is the dual-rail, asynchronous NCL output.

For the A = '1' case, output Z should become a logic '1' regardless of the value on the Boolean input, B. In asynchronous NCL, this corresponds to a DATA (Vdd) on the logic '1' wire, $Z1$, and a NULL (Vss) on the logic '0' wire, $Z0$. Since the NCL input signal A is in the critical path, the dual-rail NCL wire $A1$ will become Vdd after the arrival of the value on Boolean input B (which is a don't care for this case). Since $A1$ becomes Vdd, according to NCL convention, $A0$ will remain Vss. In Fig. 13 with $A1$ = Vdd, $Z1$ will be Set to Vdd, and with $A0$ = Vss, $Z0$ will stay at Vss. Further, $Z1$ will remain Vdd (hysteresis) until $A1$ is Reset to Vss.

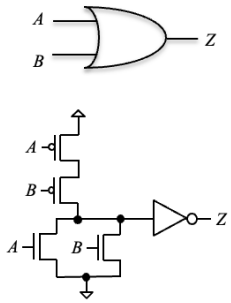

Fig. 10. Standard combinational OR2 gate design.

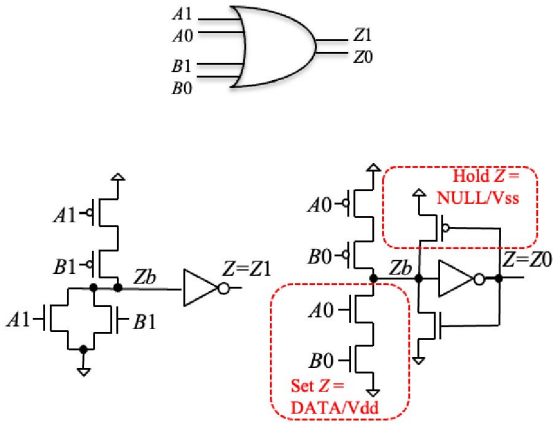

Fig. 11. Fully asynchronous NCL_OR2 combinational gate design.

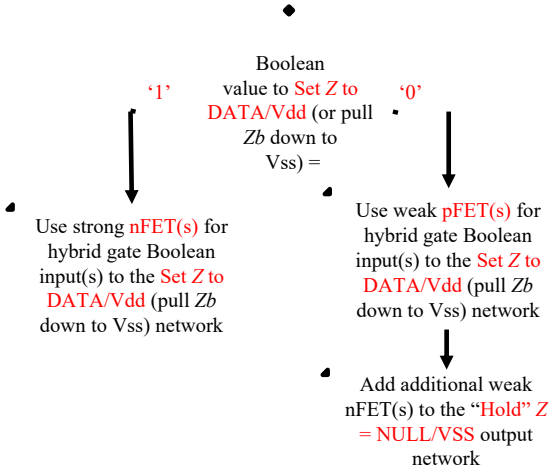

Fig. 12. Hybrid gate design flow.

For the B = '1' case, output Z should go to a logic '1' regardless of the value that the NCL input signal A eventually becomes. In the new data-path approach, B becomes '1,' and then either $A1$ or $A0$ will become DATA (Vdd), while the other remains NULL (Vss). In the circuit shown in Fig. 13, if $A1$ = Vdd ($A0$ = Vss), the logic '1' output wire, $Z1$, will become Vdd, and the logic '0' output wire, $Z0$, will be Vss. Else if $A0$ = Vdd

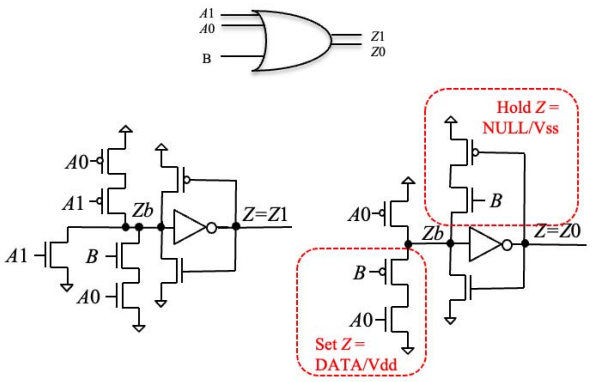

Fig. 13. Hybrid data-path replacement OR2 with built-in signal conditioning.

(*A*1 = Vss), the logic '1' output wire, *Z*1, will become Vdd, and the logic '0' output wire, *Z*0, will remain Vss. So, the logic '1' output wire, *Z*1, is Set regardless of the value signal *A* becomes. Further, *Z*1 will remain Vdd until *A* is Reset (*A*1 = *A*0 = Vss).

It should be noted that for both the previous cases, the Boolean input is either a logic '1' or a don't care. For the final case, where the Boolean signal has a controlling value of '0,' the pFET design flow in Fig. 12 comes into play.

For the *A* = *B* = '0' case of the OR2 example in Fig. 13, the output should eventually become a logic '0.' For NCL, both *A*1 and *A*0 are initialized to NULL (Vss). In the example OR2 circuit (Fig. 13), these NULL values force *Z*1 = *Z*0 = Vss regardless of the value on Boolean input *B*. Based on the data-path assertions, *B* becomes '0' before *A* is asserted. With *B* = '0,' when *A*0 is asserted = Vdd (*A*1 still = Vss), *Z*1 will remain Vss, however, *Zb* in the *Z*0 subcircuit will be pulled down through the weak *B* pFET, and *Z*0 will become Vdd. So, the logic '0' output wire, *Z*0, is Set. Further, *Z*0 will remain Set to Vdd until *A* is Reset (*A*1 = *A*0 = Vss).

To satisfy the assertion in section IV, the propagational delay of a replacement hybrid gate must be ≥ the delay of the replaced Boolean gate. Given the assertion is true, it can be safely assumed that all Boolean combinational signal values arrive before the NCL dual-rail signal values. To guarantee the assertion, designers can carefully control hybrid gate transistor sizing. One simple approach is to start with standard proportional transistor widths for the nMOS and pMOS transistors in the hybrid gates and performing spice simulations to compare propagation delays between the Boolean logic gates and their hybrid replacements. The widths of the hybrid gate transistors can be adjusted to increase the hybrid propagation delay (delay between NCL dual-rail input change and corresponding dual-rail output change) to a percent difference that guarantees reliable operation for the envelop of a particular target technology fabrication node.

For comparison purposes, Table I below shows the relative transistor count for a set of standard Boolean logic gates, fully NCL asynchronous equivalent logic gates, and the critical path hybrid replacement gates. It should be noted these are the gates

TABLE I. BOOLEAN, NCL, AND HYBRID GATE COMPARISON

	Transistor Count		
	Std Boolean	***Semi-static NCL***	***Hybrid***
nand2	4	14	17
nand3	6	18	20
nand4	8	22	23
nand5	14	36	40
nor2	4	14	17
and2	6	14	17
and3	8	18	20
and4	10	22	23
or2	6	14	17
or3	8	18	20
or4	10	22	23
xor2	12	24	22

required for comparison of the benchmark circuits analyzed below in Table II. More details are found in [15][16].

VI. EVALUATION AND ANALYSIS

A set of benchmark circuits (including circuits from the literature and circuits from common signal processing and cyber physical system (CPS) applications) was used to compare transistor counts of several standard Boolean, fully asynchronous, and hybrid circuits [17]. Since the method is applied to the combinational blocks of logic in synchronous sequential systems, only combinational circuits are evaluated. First, a completely asynchronous version of each benchmark circuit was generated by replacing the standard Boolean gates and signals with their dual-rail, semi-static NCL equivalents. To implement the hybrid version of each benchmark circuit, the Cadence *Genus* static timing analysis tool was used to determine the critical path through each of the combinational Boolean circuits. Then the transistor level library of hybrid, Boolean to NCL interface gates (shown in Table I above) was used to replace the standard Boolean gates in the critical paths.

The average number of MOS FETs per standard Boolean benchmark circuit was 5,438, and the *number of transistors* (*# FETs*) and *percent increase* (*Tran Inc*) for each version of the individual benchmark circuits is shown below in Table II. For the fully asynchronous NCL semi-static version of the benchmark circuits, the average number of transistors was 13,433 or 2.47x the size of the Boolean circuit. In contrast, for the asynchronous hybrid version, the average number of transistors was only 5,787 or 1.06x the number in the Boolean circuit for an average increase of only 6%. Based on the data from Table II, larger circuits seem to benefit more than smaller ones; however, more data is needed to verify this trend. Future work and analysis includes applying the technique to complete synchronous sequential systems, and then, quantifying any improvement in digital noise floor for SOC applications and SCA protection for security applications.

TABLE II. BENCHMARK CIRCUIT COMPARISON

	Transistor Count Comparison				
	Std Boolean	*Semi-static NCL*		*Hybrid Circuit*	
	# FETs	*# FETs*	*Tran Inc*	*# FETs*	*Tran Inc*
c432	826	1894	229 %	1008	22 %
c499	1796	3564	198 %	1904	6 %
c880	1802	4324	240 %	2037	13 %
c1355	2276	6892	303 %	2556	12 %
c1908	3330	7402	222 %	3628	9 %
c2670	5008	10382	207 %	5169	3 %
c3540	7194	14980	208 %	7518	5 %
c5315	11332	22676	200 %	11700	3 %
c6288	10112	33376	330 %	11031	9 %
c7552	15624	32108	206 %	15939	2 %
WTM4	468	1120	239 %	609	30 %
WTM12	876	2040	233 %	1309	49 %
RISC V	10048	33868	337 %	10826	8 %

VII. CONCLUSIONS

Area overhead has been a major hurdle to the practicality of asynchronous circuits for both Systems-on-a-Chip (SOC) and security and trust applications. The improved data-path method presented here is based on replacing standard Boolean logic gates in combinational logic critical delay paths with special hybrid logic gates. The hybrid logic gates have a dual-rail, asynchronous NCL input; one or more standard logic Boolean inputs; and a dual-rail, asynchronous NCL output. A key advantage of the hybrid gates is that all signal conditioning required to convert standard Boolean inputs to asynchronous inputs is included within the hybrid gate, so no additional signal conditioning is required. This trait, along with controlled delay, guarantees that the critical path will remain intact when the hybrid gates are inserted. The method can be applied by nonexperts in asynchronous circuit design using standard state-of-the-art industry synthesis tools. Most important, the resulting hybrid circuits retain the advantages of asynchronous circuit processing (distributed in time processing) with an average transistor increase of only 6% for the circuits tested here. Future work includes fabricating and testing a library of hybrid cells, taping out example circuits, and implementation of both SOC and CPS applications to quantify improvement to both noise mitigation as well as SCA protection.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO 1999, LNCS*, M. Wiener Ed., vol. 1666, Springer, Heidelberg, 1999, pp. 388-397.

[2] M. Tehranipoor and F. Koushanfar, "A survey of HW Trojan taxonomy and detection." in *IEEE Des. Test Comput.*, vol. 27, 2010, pp. 10–25.

[3] L. Lin, W. Burleson, and C. Parr, "MOLES: malicious off-chip leakage enabled by side-channels," in *Proc. IEEE/ACM International Conference on CAD (ICCAD)*, Nov. 2009, pp. 117–122.

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

[5] R. Sridhar, "Asynchronous design techniques," in *Proc. 5th Annual IEEE International ASIC Conference*, Sep. 1992, pp. 296-300.

[6] K. M. Fant and S. A. Brandt, "Null convention logic: a complete and consistent logic for asynchronous digital circuit synthesis," *Proc. IEEE Int. Conf. on Application Specific Systems, Architectures and Processors*, Aug. 1996, pp. 261-273.

[7] S. M. Nowick and M. Singh, "Asynchronous design—part 1: overview and recent advances," in *IEEE Des. Test*, vol. 32, no. 3, Jun. 2015, pp. 5-18.

[8] J. J. A. Fournier, S. Moore, H. Li, and R. Mullins, and G. Taylor, "Security evaluation of asynchronous circuits," in *CHES 2003, LNCS*, vol. 2779, Springer, Berlin, Heidelberg, 2003, pp. 137-151.

[9] J. Wu, "Null convention logic applications of asynchronous design in nanotechnology and cryptographic security," Ph.D. dissertation, Missouri S&T, 2012.

[10] C. F. Brej, "An automatic synchronous to asynchronous circuit convertor," in *Proc. 11th UK Asynchronous Forum*, Dec. 2001.

[11] S. Semba and H. Saito, "Comparison of RTL conversion and GL conversion from synchronous circuits to asynchronous circuits," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1-4.

[12] A. Branover, R. Kol, and R. Ginosar, "Asynchronous design by conversion: converting synchronous circuits into asynchronous ones," in *Proc. Des., Automation and Test in Europe Conference and Exhibition*, vol. 2, Feb. 2004, pp. 870-875.

[13] H. Park and T. Kim, "Synthesizing asynchronous circuits toward practical use," in *Proc. IEEE Comp. Soc. Annual Symp. on VLSI (ISVLSI)*, Jul. 2016, pp. 47-52.

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

[15] D. Phillips, P. Chen, and J. M. Emmert, "Data-path cells for NCL asynchronous circuit area reduction," *Proc. IEEE Nationtal Aerospace & Electronics Conference (NAECON)*, Aug. 2021.

[16] J. M. Emmert, "Area efficient asynchronous circuit generator," U.S. Patent Appl. 2022/63346711, May 2022.

[17] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits," *Proc. IEEE Int. Symp. Circuits and Systems*, May 1985, pp. 695-698.