An Equivalence Verification Methodology for Combinational Asynchronous PCHB Circuits

Ashiq A. Sakib, Scott C. Smith and Sudarshan K. Srinivasan
Department of Electrical and Computer Engineering
North Dakota State University
Fargo, ND, USA

Abstract— **Pre-Charge Half Buffer (PCHB) is a Quasi-Delay Insensitive (QDI) asynchronous design paradigm that has found commercial applications in the semiconductor industry. PCHB circuits use dual-rail signals instead of Boolean logic and are unique in that PCHB gates incorporate both registration and a handshaking scheme for synchronization. We have developed a methodology for formal equivalence verification of combinational PCHB circuits against their corresponding Boolean specification circuits. The methodology transforms the PCHB circuit into a Boolean circuit, which can then be checked against a Boolean specification circuit using an existing combinational equivalence checker. The methodology also checks for liveness and handshaking correctness of the original PCHB circuit. The proposed methodology has been demonstrated using several multipliers and ISCAS circuit benchmarks.**

Keywords— **Asynchronous Circuits, Formal Verification, Equivalence Checker, Pre-Charge Half Buffer.**

I. Introduction

The traditional synchronous, clocked approach to digital circuit design is now suffering from many clock related issues, such as timing closure and excessive power dissipation, due to decreasing transistor feature size and increasing operating frequency. On the other hand, asynchronous, clockless Quasi-Delay Insensitive (QDI) circuits eliminate these clock-related issues, and provide a robust, low-power approach to designing digital circuits [1]. Therefore, QDI circuits are becoming more widely utilized in the multi-billion-dollar semiconductor industry, as predicted by the International Technology Roadmap for Semiconductors (ITRS) [2]. Pre-Charge Half Buffer (PCHB) [3] is one such commercially successful QDI paradigm that has been utilized by major semiconductor companies, such as Intel [1].

Nowadays, formal verification is required by industry, especially for safety critical applications, to detect corner case bugs and guarantee circuit correctness. Exhaustive testing can detect shallow bugs, but cannot ensure complete correctness. Although PCHB circuits have found some commercial success, there exist few verification schemes for such circuits; and existing methods have several drawbacks.

This paper proposes a formal verification methodology for combinational PCHB circuits, and compares the results with a previous method [9]. The paper is divided into five main sections. Section 2 contains a brief overview of PCHB circuits and reviews related work on PCHB verification. The proposed PCHB verification methodology is detailed in Section 3; Section 4 discusses the results; and Section 5 provides conclusions and directions for future work.

II. PCHB Background

A. PCHB Functionality

PCHB gates incorporate both registration and handshaking control [3]. In addition to performing specific logic functions they also behave as memory elements. Therefore, an arrangement of multiple gates in a combinational PCHB circuit operates similar to a synchronous pipeline; i.e., PCHB circuits themselves are not combinational; they include internal feedback, as shown in Fig. 1. The control consists of request and acknowledge signals from individual gates, *Rack* and *Lack*, respectively, and a combination of C-elements [4] to establish a well-defined handshaking scheme for synchronization. These unique features add to the complexity of the design, making formal verification of such circuits very challenging.

QDI circuits, such as PCHB, utilize multi-rail logic signals, such as dual rail logic, where a signal consists of two wires, D^0 and D^1. It can represent three states: DATA0 ($D^0=1$, $D^1=0$), DATA1 ($D^0=0$, $D^1=1$) and NULL ($D^0=0$, $D^1=0$). DATA0 and DATA1 are equivalent to Boolean logic '0' and '1', respectively. Unavailability of DATA is represented by NULL. ($D^0=1$, $D^1=1$) is an illegal state, as two rails cannot be asserted simultaneously. A PCHB gate has dual-rail inputs and outputs,

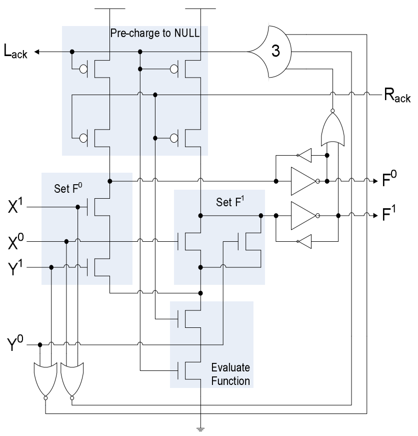

Fig. 1. PCHB NAND2 Circuit [5]

X and *Y*, and *F*, respectively, for the NAND2 example in Fig. 1. The set functions, F^0 and F^1, are implemented to achieve the particular gate functionality. The 2-input NOR gates connected to both inputs' rails and the output's rails detect when a dual-rail signal is either DATA or NULL; and the C-element connects these completion detection signals to generate the gate's acknowledge signal, *Lack*. The weak inverter arrangement is used to hold the output DATA until pre-charged back to NULL to attain delay-insensitivity. When *Lack* is request-for-data (*rfd*), the inputs will eventually become DATA and when *Lack* is request-for-NULL (*rfn*), the inputs will eventually become NULL. The function evaluates, and the output becomes DATA whenever both *Lack* and *Rack* are *rfd* and the *X* and *Y* inputs are DATA. If *Rack* is *rfd* and *Lack* is *rfn*, or vice versa, the state is held by the weak inverters. When *Lack* and *Rack* are both *rfn*, the output is pre-charged back to NULL. Whenever the inputs and outputs are all DATA, *Lack* changes to *rfn*; and when the inputs and output are all NULL, *Lack* changes to *rfd*.

Handshaking logic between PCHB gates can be implemented using either full-word or bit-wise completion [12], or some combination of the two. Full-word completion requires that the *Lack* signal of each PCHB gate in level_i be conjoined by one or more C-elements to produce a single *Lack* signal, whose output is connected to the *Rack* signal of each PCHB gate in level_{i-1}, where a gate's level is the longest path (in terms of number of PCHB gates) from the circuit's primary inputs to that gate's output. On the other hand, bit-wise completion only sends the completion signal from PCHB gate *b* back to each PCHB gate whose output is an input to gate *b*.

B. *Related Verification Work*

There have been several methodologies implemented to verify different models of asynchronous circuits, with the ones directly applicable to PCHB described below. Chuang et al. developed a deadlock verification scheme for sequential PCHB circuits in [7]; however, the method does not address verification of the combinational units, neither their functionality nor their handshaking connections. They assume that their optimized synthesis method for generating a combinational PCHB circuit from its Boolean specification [6] is correct. For example, inversion in a handshaking signal will cause deadlock and swapped rails of a dual-rail signal will produce incorrect results, but will not deadlock the system, neither of which [7] would detect. In [8], a reverse synthesis-based approach to creating a high-level specification is presented; however, in case of a bug, the methodology does not address the issue of finding the error. Also, [8] is applicable to control circuits, but not datapath circuits. A formal verification methodology for combinational PCHB circuits is presented in [9], based on model checking. The methodology models PCHB gates as transition systems (TS) and verifies a comprehensive set of properties that check for safety and liveness of the circuit. However, scalability is the major drawback of that method, as the resulting circuit TSs suffer from state explosion, which in turn leads to infeasible verification times.

III. Proposed Verification Methodology

This paper proposes a fast and highly scalable formal verification methodology for combinational PCHB circuits. The methodology includes an equivalence verification scheme that verifies the functionality of combinational PCHB circuits against their respective Boolean specifications to check for safety, and a graph-based approach to check for liveness and handshaking correctness, both described below.

A. *Safety Check: An Equivalence Verification Methodology*

The safety check requires two steps. First, a conversion algorithm takes the netlist of a combinational PCHB circuit as input and transforms that into a corresponding Boolean netlist. The generated Boolean circuit is then checked against the Boolean specification using an equivalence checker. To describe the methodology, the 2x2 PCHB multiplier, shown in Fig. 2 is used as an example. Note that although the PCHB multiplier is similar to a Boolean multiplier at the gate level, PCHB gate structures are far more complex. For example, a 2-input Boolean NAND gate only requires 4 transistors; whereas a 2-input PCHB NAND gate requires 46 transistors to account for dual-rail signaling, registration, and handshaking control. In general, PCHB circuits require approximately 8-15 times more transistors than corresponding Boolean circuits due to their complex features, as can be seen in Table 1.

Fig. 3 shows the netlist format of the 2x2 PCHB multiplier. The first two lines correspond to all primary inputs and outputs of the circuit, respectively. A dual-rail signal, *a0* is represented as *"a0_1a0_0"*, where *a0_1* and *a0_0* are $rail^1$ and $rail^0$ of *a0*, respectively. Lines 3 to 10 represent the individual PCHB gates used in the circuit. The first column of each of these lines represent the *type* of the gate, where the ending number represents the number of gate inputs; e.g., *and2* represents a 2-input AND gate. The second column indicates the *level* of the gate, which is the longest path (in terms of number of PCHB gates) from the circuit's primary inputs to that gate's output. The remaining columns list the gate's *input* signals, *Rack* signal, *Lack* signal, and *output* signal, respectively. Type *Cn* in lines 11-13 represent an n-input C-element used to connect the PCHB handshake signals. Following *Cn* are its *n* input signals, and then its output signal.

The PCHB netlist is converted into its corresponding Boolean netlist, shown in Fig. 4a, by replacing each PCHB gate with its corresponding Boolean gate, and each dual-rail signal with a corresponding Boolean signal. Swapped rails of a dual-rail signal result in the introduction of an inverter. For example, if line 3 of Fig. 3 was instead "and2 1 a0_0a0_1 …", this would result in the following additional line in Fig. 4a: "inv 1 a0, a0_bar", and line 3 of Fig. 4a to be changed to "and2 2 a0_bar, b0 p0". If a PCHB gate input's $rail^1$ and $rail^0$ are not part of the same dual-rail signal, an error message is generated noting the misconnection between rails and where this occurs. The converted Boolean netlist is then encoded in the Satisfiability Modulo Theory Library (SMT_LIB) language,

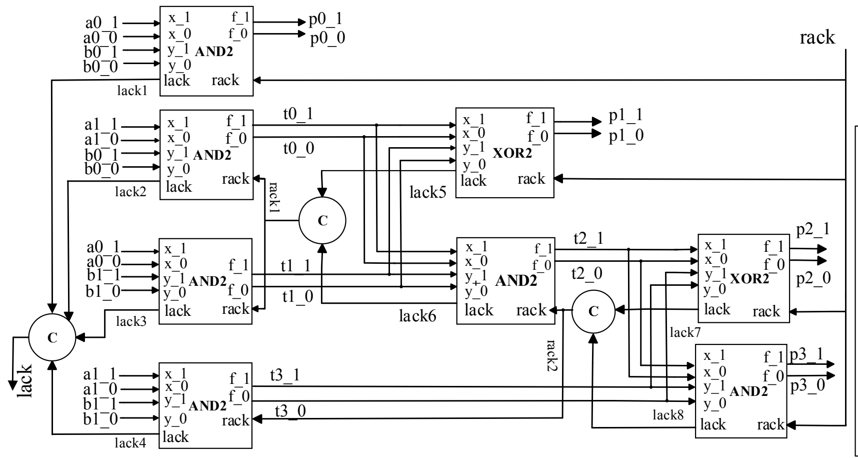

Fig. 2. PCHB 2x2 multiplier

```
1.  a0_1a0_0, a1_1a1_0, b0_1b0_0, b1_1b1_0
2.  p0_1p0_0, p1_1p1_0, p2_1p2_0, p3_1p3_0
3.  and2 1  a0_1a0_0, b0_1b0_0  rack  lack1  p0_1p0_0
4.  and2 1  a1_1a1_0, b0_1b0_0  rack1  lack2 t0_1t0_0
5.  and2 1  a0_1a0_0, b1_1b1_0  rack1  lack3 t1_1t1_0
6.  and2 1  a1_1a1_0, b1_1b1_0  rack2  lack4 t3_1t3_0
7.  xor2 2  t0_1t0_0, t1_1t1_0  rack  lack5  p1_1p1_0
8.  and2 2  t0_1t0_0, t1_1t1_0  rack2  lack6  t2_1t2_0
9.  xor2 3  t2_1t2_0, t3_1t3_0  rack  lack7  p2_1p2_0
10. and2 3  t2_1t2_0, t3_1t3_0  rack  lack8  p3_1p3_0
11. C2    lack5, lack6   rack1
12. C2    lack7, lack8   rack2
13. C4    lack1 ,lack2, lack3, lack4  lack
```

Fig. 3. PCHB netlist of a 2x2 Multiplier

(a)	(b)		
1. a0, a1, b0, b1	a0:	fanout: [1 3]	comp_fanin: [1 2 3 4]
2. p0, p1, p2, p3	a1:	fanout: [2 4]	comp_fanin: [1 2 3 4]
3. and2 1 a0,b0 p0	b0:	fanout: [1 2]	comp_fanin: [1 2 3 4]
4. and2 1 a1,b0 t0	b1:	fanout: [3 4]	comp_fanin: [1 2 3 4]
5. and2 1 a0,b1 t1	1:	fanout: 0	comp_fanin: 0
6. and2 1 a1,b1 t3	2:	fanout: [5 6]	comp_fanin: [5 6]
7. xor2 2 t0, t1 p1	3:	fanout: [5 6]	comp_fanin: [5 6]
8. and2 2 t0, t1 t2	4:	fanout: [7 8]	comp_fanin: [7 8]
9. xor2 3 t2, t3 p2	5:	fanout: 0	comp_fanin: 0
10. and2 3 t2, t3 p3	6:	fanout: [7 8]	comp_fanin: [7 8]
	7:	fanout: 0	comp_fanin: 0
	8:	fanout: 0	comp_fanin: 0

Fig. 4. a) Converted Boolean netlist b) fan_out and comp_fanin structure

using a developed automated conversion tool, which is then input to an SMT solver to check for functional equivalence between the transformed Boolean version of the original PCHB circuit and its corresponding Boolean specification. For the 2x2 multiplier example, the SMT solver checks for the following safety property: $F_{PCHB\_Bool\_Equivalent}$ (*a0, a1, b0, b1*) = MUL *(a, b),* where (*a1, a0*) and (*b1, b0*) are the (Most Significant Bit, Least Significant Bit) of *a* and *b*, respectively. We use the z3 SMT solver [11] to check for equivalence verification, but any combinational equivalence checker could be used.

B. Liveness and Handshaking Correction Check

Liveness means absence of deadlock in a circuit. For combinational PCHB circuits, proper connections between handshaking signals ensures liveness and proper synchronization. The same PCHB netlist shown in Fig. 3 and used as input for the safety check method, is utilized as input for the liveness check to trace back the handshaking paths and C-element connections to verify proper handshaking, ensuring that every output generated by a particular input, acknowledges that input. For each PCHB gate, *i*, its output is compared with every other PCHB gate *j*'s inputs, *i≠j*, to generate a fanout list, *fanout(i),* for PCHB gate *i*. For example, referring to Fig. 3, fanout for the *and2* gate on line 4 would contain the *xor2* gate on line 7 and the *and2* gate on line 8. For each PCHB gate, *i*, its *Rack* input is compared with every other PCHB gate *j*'s *Lack* output, *i≠j*, and every C-element's output, to generate a completion fanin list, *comp_fanin(i),* for PCHB gate *i*. For example, referring to Fig. 3, *comp_fanin* for the *and2* gate on line 4 would contain the *xor2* gate on line 7 and the *and2* gate on line 8, since they are both inputs to the C-element on line 11, whose output is the *Rack* input of the *and2* gate on line 4. Similarly, a *fanout* and *comp_fanin* list is generated for each external input.

After *fanout* and *comp_fanin* for each PCHB gate and external input is calculated, as shown in Fig. 4b for the 2x2 multiplier example, *fanout(k)* is checked to ensure that it is a subset of *comp_fanin(k),* for all PCHB gates and external inputs. Bit-wise completion results in *fanout(k)* being equal to *comp_fanin(k)*, while full-word completion results in *fanout(k)* being a proper subset of *comp_fanin(k)*, with the restriction that each gate that is in *comp_fanin(k)* and not in *fanout(k)* must be from the immediate subsequent level of gate/input *k*. *fanout(k)* not being a subset of *comp_fanin(k)* could result in deadlock, while *fanout(k)* being a proper subset of *comp_fanin(k)* but violating the level restriction described above, could decrease circuit performance, but will not result in deadlock. Hence, if *fanout(k)* is a proper subset of *comp_fanin(k)*, then each gate that is in *comp_fanin(k)* and not in *fanout(k)* must be inspected to ensure that it meets this level restriction. If not, it is flagged as an incorrect connection. Note that *fanout* 0 indicates an external output, while *comp_fanin* 0 denotes an external *Rack* input. The running time for this liveness check algorithm is *O(I+P)*(P+C),* where *I*, *P and C* are the number of external inputs, PCHB gates, and C-elements in the circuit, respectively.

IV. Results

The proposed methodology has been demonstrated on several multipliers and ISCAS-85[10] combinational circuit benchmarks; and the verification times are compared with those from an existing PCHB formal verification methodology based on model checking [9]. As shown in Table 1, the proposed methodology is significantly faster than the model checking based approach for every circuit. Furthermore, the proposed methodology was able to verify complex circuits with hundreds of gates, such as a 12x12 multiplier; whereas, the model checking approach Timed Out (TO) for much smaller circuits, demonstrating the scalability of the approach

TABLE I. VERIFICATION RESULTS FOR VARIOUS COMBINATIONAL PCHB CIRCUITS.

PCHB Circuits	# Transistors in Boolean Circuit	# Transistors in PCHB Implementation	Proposed Method Time (sec)	Model Checking [9] Time (sec)
Full Adder	36	266	<10 ms	21.54
ISCAS C-17	24	320	0.01	138.62
4-to-16 decoder	136	1,284	0.01	222.6
32- bit MUX	386	4,836	0.23	413.20
4x4 Multiplier	504	3,324	0.06	**TO**
6x6 Multiplier	1,332	8,720	0.33	**DNS**
8x8 Multiplier	2,544	16,620	12.18	**DNS**
9x9 Multiplier	3,294	21,470	96.08	**DNS**
10x10 Multiplier	4,140	26,960	741.84	**DNS**
11x11 Multiplier	5,082	33,070	7323.07	**DNS**
12x12 Multiplier	6,120	39,812	62,823.67	**DNS**
10x10Mul-B1	4,140	26,960	1.01	**DNS**
10x10Mul-B2	4,140	26,960	0.06	**DNS**
10x10Mul-B3	4,140	26,960	0.84	**DNS**
10x10Mul-B4	4,140	26,960	0.79	**DNS**
ISCAS c432	590	7,362	1.31	**DNS**

Note that DNS in Table 1 stands for "Did Not Simulate"; since the model checking approach Timed Out for a 4x4 multiplier, we can safely assume that it would time out for more complex circuits, such as ISCAS c432 and other higher order multipliers. 10x10Mul-B1, 10x10Mul-B2 and 10x10Mul-B3 are buggy circuits where bugs were introduced in data signals, logic elements and handshaking units, respectively. 10x10Mul-B4 is a buggy circuit with swapped rail connections. In every case, the proposed methodology detected and identified each bug very fast. Verification was performed using the z3 SMT solver [11] on an Intel® Core™ i7-4790 CPU with 32GB of RAM, running at 3.60 GHz. The verification times in Table 1 only include the z3 runtime, as the netlist conversion times and time required to verify the handshaking signals were negligible in comparison.

V. CONCLUSIONS AND FUTURE WORK

Combinational PCHB circuits operate similar to a synchronous pipeline, which makes verification much more difficult. The proposed equivalence verification methodology for combinational PCHB circuits presented herein is significantly faster and much more scalable compared to the previous model checking approach; and it detected all inserted test bugs. Future work consists of extending the proposed methodology to sequential PCHB circuits, including those with data feedback.

ACKNOWLEDGEMENT

This paper is based upon work supported by the National Science Foundation under Grant No. CCF-1717420.

REFERENCES

1. S. C. Smith and J. Di, "The future of asynchronous logic", *in VLSI: Circuits for Emerging Applications*, CRC Press, 2014.
2. International Technology Roadmap for Semiconductors, 2013 (most recent) Edition, http://www.itrs2.net/2013-itrs.html. [Accessed: 2/28/18].
3. A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on chip design", *in Proceedings of the IEEE,* pp. 1089-1120, Vol. 94/6, 2006.
4. D. E. Muller, "Asynchronous logics and application to information processing", *in Switching Theory in Space Technology*, Stanford University Press, pp. 289- 297, 1963.
5. L. Zhou and S. C. Smith, "Static implementation of Quasi-Delay-Insensitive Pre-Charge Half-Buffers", *in IEEE Midwest Symposium on Circuits and Systems,* pp. 636-639, August 2010.
6. C. Chuang, Y. Lai, J. R. Jiang, "Synthesis of PCHB-WCHB hybrid quasi-delay insensitive circuits", *in Design Automation Conference (DAC)*, pp. 1-6, 2014.
7. C. Shih, Y. Lai and J. R. Jiang, "SPOCK: Static performance analysis and deadlock verification for efficient asynchronous circuit synthesis", *in IEEE/ACM International Conference on Computer-Aided Design (ICCAD),* pp. 442-449, 2015.
8. S.J. Longfield, R. Manohar, "Inverting Martin Synthesis for verification", *in International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2013.
9. A. A. Sakib, S. C. Smith, and S. K. Srinivasan, "Formal modeling and verification for pre-charge half buffer gates and circuits", *in IEEE International Midwest Symposium on Circuits and Systems*, pp. 519-522, August 2017.
10. D. Bryan, 'The ISCAS '85 benchmark circuits and netlist format', https://ddd.fit.cvut.cz/prj/Benchmarks/iscas85.pdf. [Accessed: 1/21/18].
11. L. M. de Moura and N. Bjørner, "Z3: An efficient smt solver*", in TACAS, ser. Lecture Notes in Computer Science*, C. R. Ramakrishnan and J. Rehof, Eds., vol. 4963. Springer, pp. 337–340, 2008.
12. S. C. Smith, "Completion-Completeness for NULL Convention Digital Circuits Utilizing the Bit-wise Completion Strategy," *in International Conference on VLSI*, pp. 143-149, June 2003.