2014 17th Euromicro Conference on Digital System Design

Design Techniques for NCL-based Asynchronous Circuits on Commercial FPGA

Matthew M. Kim
Electrical & Computer Engineering
RMIT University,
Melbourne, Australia
e-mail: myungha.kim@rmit.edu.au

Paul Beckett
Electrical & Computer Engineering
RMIT University,
Melbourne, Australia
e-mail: pbeckett@rmit.edu.au

***Abstract*—While asynchronous techniques are of increasing interest in low-power design, designers cannot simply transfer current synchronous techniques to that domain. In particular, commercial FPGA systems and their accompanying EDA tools are not well suited to asynchronous logic design. In this paper we describe and analyze five alternative description methods that allow Null Convention Logic (NCL) based Asynchronous Circuits to be mapped to a commercial FPGA using the Verilog hardware description language and standard FPGA design tools. The techniques enable simple but robust NCL circuits to be developed using conventional methodologies.**

Keywords-asynchronous logic; verilog; Null Convention Logic

I. INTRODUCTION

For many years, synchronous logic design has dominated because of its inherent ease of design and abundant design tools. However, the scaling of semiconductor processes into deep sub-micron and nanoscale dimensions issues such as variability, clock distribution and power have prompted a reevaluation of asynchronous techniques. In particular, the power consumption of high-rate clock trees has become a major concern of IC designers especially considering battery powered mobile and portable systems.

A number of techniques have been proposed to manage the additional design complexity of the asynchronous (i.e., clock-less) approach. In general, the centralized clock is replaced by localized handshaking signals to control data transfer. There are two major types of asynchronous design approaches. They are Bounded-delay model and Delay-insensitive model [1]. Null Convention Logic [2] employs a quasi delay insensitive (QDI) model where the delay constraints are partially relaxed. The fundamental operational unit (gate) in NCL is a complex threshold gate which exhibits internal state holding behavior.

One of the difficulties of all asynchronous logic design approaches is their unsuitability for use in standard FPGA devices using a conventional tool chain. While there have been a number of attempts to implement asynchronous design on commercial FPGA Look-up Tables (LUT) [3-10] these have tended to require either special steps to be inserted into the tool flow, or for the circuit's timing to be (manually) constrained such that it complies with the applicable delay model. For example, a LUT-based method has been described in [3] but their Delay Insensitive approach mandates strict constraints to meet the isochronic fork balance requirement and also the LUT based design is limited to a specific FPGA vendor. In a similar manner, the asynchronous logic implemented in [4] (targeting a Xilinx LUT) and [8] also required complicated area and strict timing constraints to ensure that the synthesized logic met its target delay time. The asynchronous LUT-based technique proposed for sensor network design by Liu [5] also requires very careful timing constraints and delay calculations. Although the XBM controller implemented in [6] did achieve hazard free asynchronous behavior, the approach would be difficult to apply in a general case and is inefficient. Ho [7] also implemented QDI logic but this had to be designed by hand to carefully control the delay time. Brunvand [10] also used an FPGA to implement a library comprising bundled data modules. As these were created using specific Actel macro blocks the approach will not be generally applicable.

In response to these problems, there have been a number of proposals for building specialized LUT structures that directly support NCL gate implementations. The structure suggested in [11] is very similar to the commercial Xilinx Virtex-6 six-input LUT, while the special reconfigurable cell for NCL described [12] is only applicable to their special Atmel FPGA organisation rather than to general commercial FPGA devices.

Even though they are not specifically designed to support asynchronous logic, it would be useful to be able to implement conventional synchronous architectures using commercial FPGA devices without requiring extensive area or timing constraints. Basically, NCL implementation on a synchronous based commercial FPGA is significantly less efficient compared to conventional synchronous design but this approach is still worth pursuing, particularly in the case of prototypes that will eventually target ASIC layouts.

As the fundamental NCL gate is more complex than Boolean, it offers more possibilities for circuit optimization [13, 14]. A "natural" design style for NCL is majority threshold logic and various experiments over the years have established that 27 fundamental gates [1] are sufficient to cover all logic functions of between 2 and 5 inputs although, in reality any type of threshold gates can be designed and used. Thus, when we generate an NCL netlist using specialized synthesis tools such as UNCLE [15] or BALSA [16, 17], there is scope to guide the synthesis tools towards optimal area and performance based on a judicious choice of gate complexity.

In this paper, we propose a number of alternative approaches to the design of NCL gates and describe some application examples on commercial FPGA devices. We show five different types of NCL gate descriptions: a Verilog behavioral model, Verilog LUT model, Verilog UDP model, a Schematic design model and a Verilog Boolean model. These implement hysteresis-style NCL gates, as opposed to the alternative *encoded intermediate* form which is much more

978-1-4799-5793-4/14 $31.00 © 2014 IEEE
DOI 10.1109/DSD.2014.85

expensive in terms of gate interconnections. We compare their implementations and resource usage on three commercial FPGA vendors (Xilinx®, Altera® and Actel®). The five styles have been tested on these standard FPGAs and have been shown to offer a useful way to describe NCL, particularly in respect to ASIC prototyping of FPGAs.

The remainder of this paper is organized as follows: Section II addresses the basic principles of asynchronous logic and outlines the NCL design style. Section III describes how the NCL gates are implemented on the commercial FPGA and shows each of the different approaches. Section IV explains the verification methodology for these NCL gates and illustrates their implementation result. Finally, in Section V we conclude the paper and identify future work.

II. Asynchronous Logic Design and NCL

In this section, we briefly describe main differences between conventional synchronous and asynchronous design and introduce Null Convention Logic (NCL) and its gate-level design.

Conventional synchronous design uses a centralized clock signal to control the data flow through combinational logic between edge-triggered register stages. In contrast, asynchronous design employs localized hand-shaking signals (e.g., Request and Acknowledge) to control its data flow. There are two major types of asynchronous models: bounded-delay (BD) and delay-insensitive (DI) [1]. In the BD model, delays in both gates and wires are bounded and delays are evaluated based on worse-case scenarios to avoid hazard conditions [1]. DI models do not need to consider wire and gate delays as the logic works correctly regardless of these delays. DI design requires very little timing analysis compared to the BD model.

Null Convention Logic [2] is an example of a Quasi-delay insensitive (QDI) design style based on threshold logic gates exhibiting internal state holding in each gate. It is a symbolically complete logic system that exhibits delay insensitive behavior within some minor constraints. In the same way as other DI paradigms, NCL assumes that wire forks are *isochronic* [18]. Unlike other DI techniques, NCL gates rely on one simple timing assumption: that the feedback path for its state holding elements must be fast enough compared to the delay through the gate. In the context of FPGA devices, this constraint can be difficult to achieve as the tools typically give only limited and indirect control of individual routing paths.

Input completeness requires that the output signal of the gate may not transition to DATA until all inputs have transitioned to DATA. Similarly, the output will not transition to NULL until all inputs have reached NULL. Input Completeness is achieved by adding the value NULL to the basic logic values (TRUE and FALSE), to represent a status of "no data". Output Data will only be valid when all input signals have transitioned from NULL to DATA. An NCL circuit consists of an interconnection of primitive modules known as M-of-N threshold gates with hysteresis. All functional blocks, including both combinational logic and storage elements, are constructed out of these same primitives. To represent these three values in this work we use a dual rail coding style which expresses the value '00' (NULL), '01' FALSE and '10' TRUE. A NULL wave front separates two DATA wave fronts. Unlike other DI styles, NCL uses only one type of state holding gate and NCL uses this threshold gates as its basic logic elements [11].

As already mentioned, NCL threshold gates are state-holding and designed to exhibit hysteresis. Fig. 1(a) presents the basic NCL gate components and it includes Set, Reset, Hold0 and Hold1. The Set and Reset functions have their usual functions while the Hold functions (Hold0 and Hold1) implement hysteresis [19, 20]. Once the set function becomes true the output is asserted. The output then remains asserted through the Hold1 function until all inputs return to NULL [19].

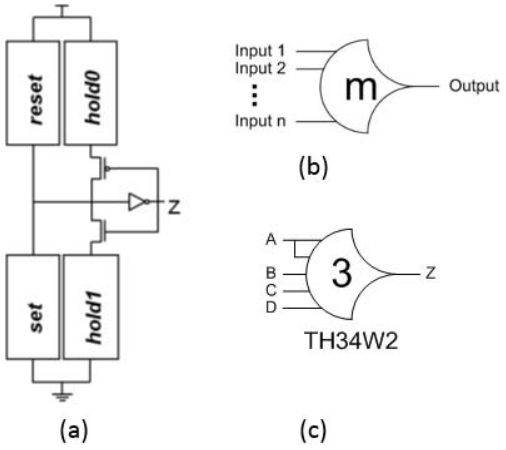

Fig. 1. (a) The structure of NCL gates; (b) General NCL Symbol; (c) Gate Symbol

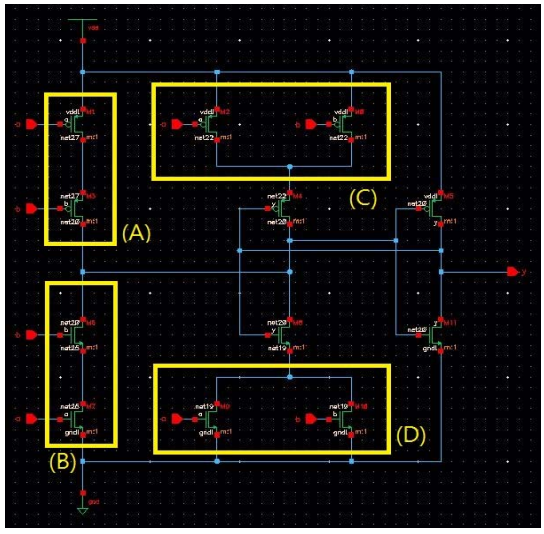

Fig. 2. TH22 Circuit Design

Figure 1 (b) illustrates a basic THmn gate, where $1 \leq m \leq n$. This gate model has *n* input signals and a single output. At least m of n inputs must be asserted before the output is asserted, as indicated by '*m*' on the symbol in Fig. 1b. An example of a second class of (weighted) threshold gate is

shown in Fig. 1(c) [11]. Here, the function is expressed as THmnw1w2..wR, where the weight wR is in the range M ≥ wR ≥ 1.

In the transistor level schematic of a TH22 gate shown in Fig. 2, Block (A) represents the reset part, Block (B) the set part, while Block (C) and (D) are the Hold0 and Hold1 respectively. In this example, the set equation is ($a.b$) and the Hold1 equation is given by ($a+b$). Reset is the complement of Hold1 ($\bar{a}.\bar{b}$) and Hold0 is complement of set ($\bar{a}+\bar{b}$). In general terms, the output Boolean equations can be described as: $Z = set + (Z^{-} \bullet Hold1)$ where Z is the current output and Z^{-} is the previous output of the gate. The complement of Z is Z′ and also can be described as: $Z' = reset + (Z^{-} \bullet Hold0)$ [19].

III. NCL GATES IMPLEMENTATION ON FPGA

While the specific focus of the NCL behavioral models derived in [15] and [21] using Verilog and VHDL has been ASIC design, obviously these models can equally be used for FPGA implementation. Most NCL gate designs for FPGA developed to date have used only a LUT-based definition. In our work we have extended this to encompass five approaches. However, although the definition is general, the target device still greatly influences the outcome. For example, Xilinx Virtex-6 [22] and Altera Stratix-IV [23] FPGA LUT have 6 input structures and so they are wide enough to implement all types of NCL threshold gates including feedback line and reset signal in a single LUT whereas, in smaller devices, an NCL gate would be partitioned across multiple LUTs.

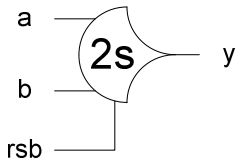

Fig. 3. TH22s Threshold Gate Symbol

TABLE I. BASIC TRUTH TABLE OF TH22S NCL GATE

rsb	a	b	y
0	X	X	1
1	0	0	0
1	0	1	Hold
1	1	0	Hold
1	1	1	1

In this section, we illustrate the five different approaches to NCL description using an example of a TH22s gate implementation on a Xilinx Virtex-6 FPGA. Fig. 3 is the NCL threshold gate symbol of TH22s and Table I shows the truth table of the gate. It performs the same function as a TH22 with an added *rsb* signal for initialization.

1) Verilog Behavioral Model

This technique uses the behavioral Verilog design of the UNCLE simulation model [15]. Similar behavioral models using VHDL have also been derived in [21]. It can be seen in Fig. 4 that the design model for the TH22s gate comprises four basic parts: reset, transition to DATA, transition to NULL and data hold. The reset part is included for initialization in the case where the gates are used to create an asynchronous latch. The final part handles the State Hold that maintains the current state when the input is in its hysteresis condition. Being a behavioral description, the actual logic resulting from this description type depends entirely on the synthesis tools. This will be our reference model for the tests in Section IV.

```
// th22s - Verilog behavioral
module th22s(y,a,b,rsb);
    output y;
    input a;
    input b;
    input rsb;

    reg yi;
    always @(a or b or rsb) begin
        if (rsb == 0) begin // reset
            yi <= 1;
        end
        else if (((a) & (b))) // Data
        begin
            yi <= 1;
        end
        else if (((a==0) & (b==0))) // Null
        begin
            yi <= 0;
        end // else State Hold
    end
    assign #1 y = yi;
endmodule
```

Fig. 4. Behavioral Model of TH22s NCL Gate

```
// th22s - LUT
module th22s(y,a,b,rsb);
    output y;
    input a;
    input b;
    input rsb;

    wire yi;
    assign #1 y = yi;

    // LUT4: 4-input Look-Up Table with general output
    //       Virtex-6
    // Xilinx HDL Language Template, version 13.2
    LUT4 #(
        .INIT(16'hE8FF)  // Specify LUT Contents
    ) LUT4_inst (
        .O(yi),   // LUT general output
        .I0(a), // LUT input
        .I1(b), // LUT input
        .I2(yi), // LUT input - Feedback of TH gates
        .I3(rsb)  // LUT input - reset
    );
    // End of LUT4_inst instantiation
endmodule
```

Fig. 5. LUT-based Model of TH22s NCL Gate

2) Verilog LUT model

Commercial FPGA tools generally support the creation of a dedicated LUT function using a common hardware description language. The example of Fig. 5 uses a 4-input LUT to implement the TH22s gate. The LUT contents (0xE8FF in this case) arise from a direct evaluation of the truth table (Table II) for this gate. The TH22s gate is mapped to the LUT4 component with an external feedback connection from the LUT output to input *i2*. In case of the Xilinx Virtex-6, the feed-

back path (yi) has to use an external switch-box because the slices have no internal feedback paths. It is the timing of this path that can cause problems for QDI techniques such as NCL, particularly when the feedback routing is unconstrained. Even when routing though a local switchbox, it is sometimes difficult to maintain the necessary timing behavior. Otherwise, the maximum of six inputs available on the Xilinx Virtex-6 LUT makes it a good solution for NCL gates of up to four data inputs, reset and the feedback path.

TABLE II. BASIC TRUTH TABLE OF TH22S NCL GATE

Virtex-6 LUT	I3	I2	I1	I0	O
TH22S	rsb	yi	b	a	yi
	0	0	0	0	1
	0	0	0	1	1
	0	0	1	0	1
	0	0	1	1	1
	0	1	0	0	1
	0	1	0	1	1
	0	1	1	0	1
	0	1	1	1	1
	1	0	0	0	0
	1	0	0	1	0
	1	0	1	0	0
	1	0	1	1	1
	1	1	0	0	0
	1	1	0	1	1
	1	1	1	0	1
	1	1	1	1	1

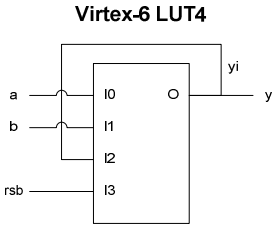

Fig. 6. LUT Mapping of TH22s NCL Gate

3) Verilog UDP model

The Verilog User Defined Primitive (UDP) function can be used to generate any type of user defined primitives, including basic Boolean gate functions. Traditionally, UDP has been used for verification purposes via test bench code but commercial FPGA vendors have now begun to support UDP as a synthesizable function. Verilog code for the UDP primitive describing the TH22s is shown in Fig. 7. This is very useful in the case of NCL gate generation because the NCL gates can be set up as user defined primitives. Unlike the LUT case, UDP is part of the standard Verilog language therefore the design can target any of the FPGA vendors as long as their specific synthesis tool supports these UDP descriptions.

```
// th22s - Verilog UDP
module th22s(y,a,b,rsb);
    output y;
    input a;
    input b;
    input rsb;

    wire yi;
    assign #1 y = yi;

    //instantiate the udp
    TH22S_UDP INST_TH22S_UDP(yi, a, b, rsb);
endmodule

primitive TH22S_UDP (Z, A, B, I);
   output Z;
   input  A, B, I; // I is "Active Low"
   reg    Z;
   initial Z = 0;
   table
      //A B I : current_state : next_state;
      ? ? 0 : ? : 1;
      0 0 1 : ? : 0;
      0 1 1 : ? : -;
      1 0 1 : ? : -;
      1 1 1 : ? : 1;
   endtable
endprimitive // TH22S_UDP
```

Fig. 7. Verilog UDP design model of TH22s gate

```
// th22s - boolean gates
module th22s(y,a,b,rsb);
    output y;
    input a;
    input b;
    input rsb;

    wire g1_out, g2_out, g3_out, g4_out
        , g5_out, g6_out, g7_out;

    // Gate Instantiation
    and G1 (g1_out, a, b); // Data
    nor G2 (g2_out, a, b); // Null

    not G3 (g3_out, rsb); // Active Low Reset
    or G4 (g4_out, g3_out, g1_out);
    and G5  (g5_out, rsb, g2_out);

    // SR Flip-Flop
    nor G6 (g6_out, g4_out, g7_out);
    nor G7 (g7_out, g5_out, g6_out);

    // Output Assignment
    assign #1 y = g7_out;
endmodule
```

Fig. 8. Boolean based TH22s gate

4) Verilog Boolean gates model

The general Boolean gate description (Fig. 8) is most useful in a structural design style where the gate instantiation (Fig. 9) can point to any FPGA vendor and/or device family without limitation, and, equally, to an ASIC standard cell component.

5) Schematic design model

The basic schematic design style for NCL was first proposed in [2], which describes a complete NCL gates design methodology using general Boolean gates including initialization. The schematic diagram of the TH22s (Fig. 10) illustrates the concept. The module comprises three basic parts: the RS-Latch at its output to implement the hysteresis function; the initialization stage (Reset in this case) plus the input logic stage that defines the logic of the gate.

```
// th22s - schematic
module th22s(y,a,b,rsb);
    output y;
    input a;
    input b;
    input rsb;

    wire yi;
    assign #1 y = yi;

    TH22S_SCH INST_TH22S
    (
    .A          (a),
    .B          (b),
    .RSB        (rsb),
    .Z          (yi)
    );
endmodule
```

Fig. 9. Xilinx schematic module instantiation of TH22s gate

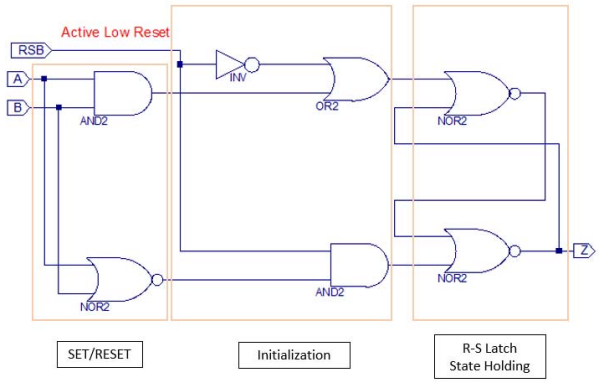

Fig. 10. Xilinx schematic design of TH22s gate

The five description methods shown above can be divided into two classes: vendor agnostic and vendor specific. It can be seen that the Behavioral, UDP and Boolean descriptions do not rely on a specific FPGA vendor or synthesis design tool because they are using standard Verilog library. In contrast, the LUT and Schematic methods are specific to a particular chip type. The Behavioral Verilog and UDP models used separate Latch components internal to the CLB to implement the RS-Latch of each threshold gate but others use only LUTs.

IV. NCL GATES VERIFICATION AND RESULTS

Asynchronous digital logic, especially NCL gates are vulnerable to glitches at their inputs that can cause hazards within the internal hysteresis latch function. In [2], Fant shows that, as long as the input transitions are monotonic, there can be no races or hazards and thus no spurious result symbols during the propagation of the monotonic wave front of correct results through the combinational expression. Maintaining monotonic behavior and avoiding glitches mandates careful testing of the NCL circuits.

Table IIIa shows the input stimulus of used in testing the example TH22s gates. If rsb is '0' the y out is always '1' and when rsb is '1' the output follows the truth table of the TH22s gate. When the input condition is *state hold*, the output maintains its previous y value. The functional simulation waveform of Fig. 11 and Table IIIb compares the result of behavioral model (Y_REF) and the other generated models (Y). The timing simulation was also set up to use the same stimulus vectors.

TABLE III. STIMULUS INPUTS AND GATE OUTPUT COMPARISON

(a)

rsb	a	b	y
0	0	0	1
0	0	1	1
0	1	1	1
0	0	0	1
0	1	0	1
0	1	1	1
0	0	0	1
0	1	1	1
0	1	1	1
0	1	0	1
0	0	0	1
0	1	1	1
0	0	1	1
0	0	0	1
0	1	1	1
0	0	0	1
1	0	0	0
1	0	1	0
1	1	1	1
1	0	0	0
1	1	0	0
1	1	1	1
1	0	0	0
1	1	1	1
1	1	1	1
1	1	0	1
1	0	0	0
1	1	1	1
1	0	1	1
1	0	0	0
1	1	1	1
1	0	0	0

(b)

```
RSB = 0, STM_VALUE = 0b00, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b01, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b00, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b10, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b00, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b10, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b00, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b01, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b00, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 0, STM_VALUE = 0b00, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b00, Y_REF = 0, Y = 0
RSB = 1, STM_VALUE = 0b01, Y_REF = 0, Y = 0
RSB = 1, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b00, Y_REF = 0, Y = 0
RSB = 1, STM_VALUE = 0b10, Y_REF = 0, Y = 0
RSB = 1, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b00, Y_REF = 0, Y = 0
RSB = 1, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b10, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b00, Y_REF = 0, Y = 0
RSB = 1, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b01, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b00, Y_REF = 0, Y = 0
RSB = 1, STM_VALUE = 0b11, Y_REF = 1, Y = 1
RSB = 1, STM_VALUE = 0b00, Y_REF = 0, Y = 0
```

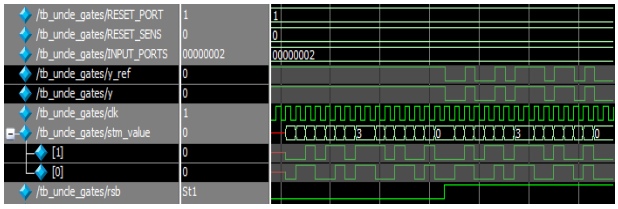

Fig. 11. Simulation Waveform of TH22s gate

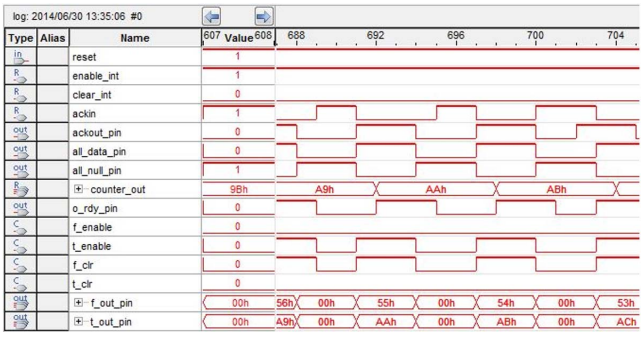

Fig. 12. SignalTap® waveforms for NCL up-counter

Fig. 12 shows a representative data capture using the Altera SignalTap® II Embedded Logic Analyzer (ELA) tool on a Cyclone-4 device. This example is based on a simple 8 bit up-

counter that was created using the Boolean gate model outlined above (see Fig. 8). In this test, while the NCL counter logic operated asynchronously (i.e., without a synchronized clock signal), we used a 200MHz sampling clock and a buffer depth of 4096 for the capture. Note that the numbers at the top of the waveform are sample numbers (not time, directly) and the combinational delays of the NCL logic have been "sampled" by the SignalTap process such that they appear on 5nS (i.e., single sample) boundaries.

In Fig. 12, the output signals `f_out_pin` and `t_out_pin` (both 8 bit vectors) represent the dual-rail output of the NCL circuit (i.e., the false and true bit lines respectively). The overall acknowledge input and output signals are `ackin` and `ackout_pin`. It can be seen that the dual-rail outputs of the counter exhibit the expected sequence of DATA and NULL values and are counting correctly. These dual-rail signals have been recombined to a single binary output (`counter_out`) that exhibits an approximate 30nS delay time, implying that the 8-bit counter is operating at about 33MHz.

Using devices that contain a 6 input LUT, each NCL gate can be mapped to a single LUT block. However, the quality of the final result depends entirely on the synthesis tools. We have tested a number of application on three different FPGA tools and device environments from Xilinx® (aimed at a Virtex-6 device), Altera® Stratix-IV and Actel® ProASIC3E devices. Table IV shows the logic resource usage of the examples used in this work. The first five items in the table show the results for a single TH22s gate, while the second set are for a Dual-Rail Latch with NULL Reset logic, which is typically used as a basic latch element for NCL. The third block of results is for a one-bit NCL Full-Adder and the final set shows results for an 8x8 NCL Multiplier with input and output Dual-rail Latch.

It can be seen that the functional representation always results in the most LUTs (e.g., three each for the TH22s). The Xilinx results show fairly similar resource usage for all types of models but in case of Altera, the Boolean, LUT and Schematic models are more efficient than the others. Actel synthesis tools do not support LUT and UDP models so we have tested only Behavioral, Boolean and Schematic models in that case.

TABLE IV. IMPLEMENTATION ON XILINX®, ALTERA® AND ACTEL® FPGA DEVICES

	Xilinx Virtex-6 FPGA							Altera Stratix-4 FPGA				Actel ProASIC3E FPGA				
th22s	LUT	O6 output only	O5 output only	O6 and O5	unused Flip Flop	Fully used (LUT+F/F)	occupied Slices	ALUTs Used	Combinational ALUTs	Total ALMs	Total LABs	Used Core	COMB Instances	COMB Tiles	SEQ Instances	SEQ tiles
Behavioral	3	2	0	1	3	0	1	3	3	2	1	4	2	2	1	2
Boolean	1	0	0	1	1	0	1	1	1	1	1	3	3	3	0	0
LUT	1	1	0	0	1	0	1	1	1	1	1	NA				
SCH	1	1	0	0	1	0	1	1	1	1	1	3	3	3	0	0
UDP	1	0	0	1	1	0	1	3	3	3	1	NA				

	Xilinx Virtex-6 FPGA							Altera Stratix-4 FPGA				Actel ProASIC3E FPGA				
drlatn	LUT	O6 output only	O5 output only	O6 and O5	unused Flip Flop	Fully used (LUT+F/F)	occupied Slices	ALUTs Used	Combinational ALUTs	Total ALMs	Total LABs	Used Core	COMB Instances	COMB Tiles	SEQ Instances	SEQ tiles
Behavioral	3	1	0	2	1	2	2	9	9	5	1	14	10	10	2	4
Boolean	3	3	0	0	3	0	2	3	3	2	1	9	9	9	0	0
LUT	3	3	0	0	3	0	2	3	3	2	1	NA				
SCH	3	3	0	0	3	0	2	3	3	2	1	9	9	9	0	0
UDP	3	1	0	2	1	2	3	7	7	6	1	NA				

	Xilinx Virtex-6 FPGA							Altera Stratix-4 FPGA				Actel ProASIC3E FPGA				
fa1	LUT	O6 output only	O5 output only	O6 and O5	unused Flip Flop	Fully used (LUT+F/F)	occupied Slices	ALUTs Used	Combinational ALUTs	Total ALMs	Total LABs	Used Core	COMB Instances	COMB Tiles	SEQ Instances	SEQ tiles
Behavioral	44	6	0	38	3	41	39	148	148	76	9	242	160	160	41	82
Boolean	51	40	0	11	51	0	29	38	38	20	2	131	131	131	0	0
LUT	47	47	0	0	47	0	18	37	37	19	2	NA				
SCH	47	47	0	0	47	0	19	38	38	20	2	152	152	152	0	0
UDP	51	15	0	36	15	36	39	108	108	60	6	NA				

	Xilinx Virtex-6 FPGA							Altera Stratix-4 FPGA				Actel ProASIC3E FPGA				
mul8x8	LUT	O6 output only	O5 output only	O6 and O5	unused Flip Flop	Fully used (LUT+F/F)	occupied Slices	ALUTs Used	Combinational ALUTs	Total ALMs	Total LABs	Used Core	COMB Instances	COMB Tiles	SEQ Instances	SEQ tiles
Behavioral	442	0	0	442	16	426	430	1339	1339	673	70	2388	1486	1486	426	852
Boolean	458	97	0	361	458	0	227	442	442	233	26	2163	2163	2163	0	0
LUT	450	450	0	0	450	0	271	442	442	231	24	NA				
SCH	450	450	0	0	450	0	259	442	442	231	25	2459	2459	2459	0	0
UDP	458	32	0	426	32	426	431	1294	1294	704	77	NA				

The ProASIC3E has a 3- input LUT therefore requires at least three logic cores to make a single TH22s gate. The advantage of this architecture is that it contains no internal flip-flops, and the tight local feedback on each LUT allows the creation of state elements as required during synthesis. Thus there are fewer wasted resources in this case: LUTs can be fully assigned to create the various components of the NCL gate (as in Fig. 10). The Actel ProASIC3E results show that the Behavioral model is less efficient for small designs but the results of each approach will be similar for larger designs.

V. CONCLUSION

In this paper we have described the implementation of asynchronous logic on commercial FPGA, especially for NCL design. Asynchronous digital logic design is an area which needs more research and, in particular, more specific tools support for product commercialization. This paper has shown that it is fairly straightforward to build asynchronous designs simply and easily on commercial reconfigurable devices. Designers can select the appropriate description method depending on their specific environment and circumstance.

However, it is still the case that one of the main difficulties of asynchronous logic design is generating proper timing constraints and analysis to check the overall throughput. Just like conventional synchronous design, asynchronous logic also needs a guarantee that the circuit meets its timing requirements for realistic implementations. In the synchronous case, these timing constraints emerge naturally from clock period considerations between flip-flops and a worse-case analysis of critical paths in the design. By definition, an asynchronous design does not have centralized clock therefore it is hard to guarantee their timing specification especially when the application throughput is timing critical.

Further, it is clear also from our work that when NCL gates are implemented on commercial FPGA devices, the synthesis results are very much larger than the corresponding synchronous case, notwithstanding the higher level of functionality of NCL gates. While it has been shown that any type of NCL gate can be mapped to a flexible 6-input LUT, significant work is still required to reduce the resulting overall area consumed by these gates.

REFERENCES

[1] S. C. Smith and J. Di, "Designing asynchronous circuits using NULL convention logic (NCL)," *Synthesis Lectures on Digital Circuits and Systems,* vol. 4, pp. 1-96, 2009.

[2] K. M. Fant, *Logically determined design: clockless system design with NULL convention logic*: John Wiley & Sons, 2005.

[3] P.-Q. Cuong and D.-D. Anh-Vu, "Hazard-free Muller Gates for Implementing Asynchronous Circuits on Xilinx FPGA," in *Electronic Design, Test and Application, 2010. DELTA '10. Fifth IEEE International Symposium on*, 2010, pp. 289-292.

[4] K. Maheswaran and V. Akella, "Hazard-free implementation of the self-timed cell set in a xilinx FPGA," *Univ. Calif. Davis, Davis, CA, Tech. Report,* 1994.

[5] L. Yijun, X. Guobo, C. Pinghua, C. Jingyu, and L. Zhenkun, "Designing an asynchronous FPGA processor for low-power sensor networks," in *Signals, Circuits and Systems, 2009. ISSCS 2009. International Symposium on*, 2009, pp. 1-6.

[6] D. L. Oliveira, S. S. Sato, O. Saotome, and R. T. de Carvalho, "Hazard-Free Implementation of the Extended Burst-Mode Asynchronous Controllers in Look-Up Table based FPGA," in *Programmable Logic, 2008 4th Southern Conference on*, 2008, pp. 143-148.

[7] Q. T. Ho, J.-B. Rigaud, L. Fesquet, M. Renaudin, and R. Rolland, "Implementing asynchronous circuits on LUT based FPGAs," in *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, ed: Springer, 2002, pp. 36-46.

[8] P. D. Ferguson, A. Efthymiou, T. Arslan, and D. Hume, "Optimising Self-timed FPGA circuits," in *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, 2010, pp. 563-570.

[9] J. Wu, Y. Shi, and M. Choi, "FPGA-based measurement and evaluation of power analysis attack resistant asynchronous s-box," in *Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE*, 2011, pp. 1-6.

[10] E. Brunvand, "Using FPGAs to implement self-timed systems," *Journal of VLSI signal processing systems for signal, image and video technology,* vol. 6, pp. 173-190, 1993.

[11] S. C. Smith, "Design of an FPGA Logic Element for Implementing Asynchronous NULL Convention Logic Circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 15, pp. 672-683, 2007.

[12] K. Meekins, "Delay Insensitive NCL Reconfigurable Logic," 2002.

[13] M. Ligthart, K. Fant, R. Smith, A. Taubin, and A. Kondratyev, "Asynchronous design using commercial HDL synthesis tools," in *Advanced Research in Asynchronous Circuits and Systems, 2000. (ASYNC 2000) Proceedings. Sixth International Symposium on*, 2000, pp. 114-125.

[14] J. Pons, J. Brault, and Y. Savaria, "State-holding free NULL Convention Logic&#x2122," in *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*, 2012, pp. 322-325.

[15] R. B. Reese, "Uncle (Unified NCL Environment)," Technical Report MSU-ECE-10-001 November

2010, available online: http://www. ece. msstate. edu/~ reese/uncle/UNCLE. pdf2011.

[16] D. Edwards and A. Bardsley, "Balsa: An Asynchronous Hardware Synthesis Language," *The Computer Journal,* vol. 45, pp. 12-18, January 1 2002.

[17] D. Edwards, A. Bardsley, L. Janin, L. Plana, and W. Toms, "Balsa: A Tutorial Guide," *The University of Manchester,* 2006.

[18] K. van Berkel, "Beware the isochronic fork," *Integration, the VLSI journal,* vol. 13, pp. 103-128, 1992.

[19] F. A. Parsan and S. C. Smith, "CMOS implementation of static threshold gates with hysteresis: A new approach," in *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, 2012, pp. 41-45.

[20] G. E. Sobelman and K. Fant, "CMOS circuit design of threshold gates with hysteresis," in *IEEE International Symposium on Circuits and Systems*, 1998, pp. 61-64.

[21] Scott Smith. (March 2014). *CCLI Project, Integrating Asynchronous Digital Design into the Undergraduate Computer Engineering Curriculum Throughout the Nation*. Available: http://www.ndsu.edu/pubweb/~scotsmit/CCLI_async.html

[22] Xilinx Inc. (March 2014). *Virtex-6 Configuration Logic Block*. Available: http://www.xilinx.com/support/documentation/user_guides/ug364.pdf

[23] A. Corp, "Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices," February 2011.