Appears in Proc. ASYNC 2009.

Reducing Power Consumption with Relaxed Quasi Delay-Insensitive Circuits

Christopher LaFrieda and Rajit Manohar
Computer Systems Laboratory
Cornell University
Ithaca, NY 14853, USA
{ccl28,rajit}@csl.cornell.edu

***Abstract*—This paper introduces novel circuits to mitigate power consumption in asynchronous logic. By exposing a preexisting timing assumption in quasi-delay insensitive (QDI) circuits, we develop a set of circuit templates that reduce dynamic power consumption while maintaining the robustness of QDI circuits. We refer to these as relaxed quasi delay-insensitive circuits (RQDI). Power consumption is reduced in four ways. First, we present a circuit template that saves power by reducing the logic required to generate enable/acknowledge signals. Second, we develop voltage converters for asynchronous channels that allow non-performance critical components to be moved to lower voltage domains. Third, we propose a circuit template that improves upon the use of multiple voltage domains by keeping the data logic in the high voltage domain, but moves the enable/acknowledge logic to the low voltage domain. Fourth, we utilize a novel 2-phase buffer to half the switching in global routing and static switching networks. Experiments show that we can reduce energy by 30-50%, with a minimal impact on area and performance.**

I. Introduction

Technology scaling through the deep submicron has greatly increased the importance of minimizing power consumption in circuit design. Transistor threshold voltage, V_{th}, no longer scales well [3] resulting in a slower scaling of the supply voltage, V_{dd}. Hence, performance improvements come at an increasing cost in energy. As performance per watt becomes more critical than frequency alone, circuit designers may be forced to abandon the highest performing circuits for more energy efficient alternatives.

Asynchronous circuits have a couple of advantages over synchronous circuits in terms of low power design. The lack of a clock network is a substantial advantage. High-speed clock networks have been known to account for 30-35% of total power in microprocessors [2]. In addition, asynchronous circuits have the equivalent of perfect clock gating. High performance asynchronous circuits are composed of many parallel processes (fine-grain pipelined circuits). These processes communicate over channels using handshakes. Processes that are not involved in the current computation do not burn dynamic power.

Unfortunately, asynchronous circuits lose some of their power savings in orchestrating handshakes between processes. Four phase handshakes, used extensively in quasi delay-insensitive (QDI) circuits, charge and discharge wires in their data channels four times per cycle. The power dissipated in channels is significant since wires there tend to be longer than wires that are local to a process. A significant amount of power is also lost in generating enable/acknowledge signals. This is particularly frustrating since those signals are not directly involved in computing the function of a particular process, but rather in detecting the validity and neutrality of inputs and outputs.

In this paper we present the following circuit techniques to reduce power consumption:

- **HCHB Template:** We define a new circuit template that reduces the logic needed to generate enable/acknowledge signals by applying an easily satisfied timing assumption.
- **Voltage Scaling:** We implement voltage scaling in two ways. One, we design efficient voltage converters that operate on data channels to support multiple voltage domains. Two, we present a circuit template that operates with its forward path (data logic) in a nominal voltage domain and its return path (enable/acknowledge) in a lower voltage domain, thus keeping latency constant.
- **Two Phase Static Switching Networks:** We propose an efficient two phase buffer and protocol converters for global communication and static switching networks, which are particularly important in FPGAs [12].

This paper is organized as follows. Section 2 reviews QDI logic, defines our conservative timing assumption and presents the resulting logic template. Section 3 introduces circuits to perform voltage scaling. Section 4 introduces a two phase buffer for static routing and its associated converters. Section 5 discusses the setup details of our experiments. Section 6 presents our results. Section 7 addresses previous work. Section 8 discusses future work and we conclude in Section 9.

II. Relaxed QDI Logic

QDI circuits are quite robust in terms of process variations and design tolerances. In this work, we expose a timing assumption used in staticizers for QDI logic and apply it to other parts of circuits. Our goal is to optimize circuits with respect to area and power while maintaining the robustness

of QDI. The resulting circuits are no longer strictly QDI. We refer to them as relaxed QDI (RQDI).

A. QDI Circuits

Quasi delay-insensitive circuits are designed by decomposing a high level description of an asynchronous system into production rules (pull-up and pull-down networks) through numerous steps [6]. For our purposes, we will focus on handshaking expansions (HSE) and preestablished circuit templates. A commonly used QDI circuit is the weak condition half buffer (WCHB) shown in Figure 1 and described with the following HSE:

$$*[[R^e \wedge L^f \longrightarrow R^f\uparrow [] R^e \wedge L^t \longrightarrow R^t\uparrow]; L^e\downarrow; \\ [\neg R^e \wedge \neg L^f \wedge \neg L^t]; R^f\downarrow, R^t\downarrow; L^e\uparrow]$$

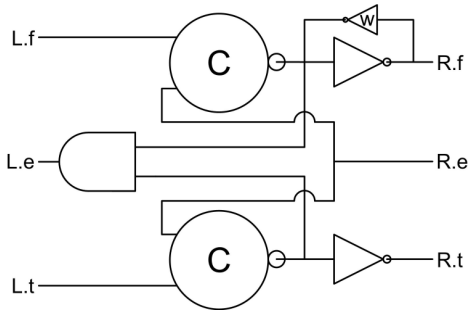

Fig. 1. A WCHB with one staticizer shown.

The WCHB takes a dual-rail (four phase) input and produces a dual rail output. There are two distinct phases: i) an evaluation phase (the first line of the HSE) where inputs arrive and the output becomes valid, and ii) a reset phase (the second line of the HSE) where the inputs and output reset. It is considered a half buffer because it takes a pair of them to store a single data token. It has a forward latency, or simply latency, of two transitions and a cycle time of ten transitions. In high performance asynchronous circuits, we strive to design circuits with a maximum latency of two transitions and a maximum cycle time of 18 transitions. A WCHB is handy for buffers, but the precharge half buffer (PCHB) is preferred for buffered logic [4]. The PCHB template for two inputs and one output is shown in Figure 2 and the HSE for the PCHB is as follows:

$$*[[R^e \wedge L^f \longrightarrow R^f\uparrow [] R^e \wedge L^t \longrightarrow R^t\uparrow]; L^e\downarrow; \\ [\neg R^e]; R^f\downarrow, R^t\downarrow; [\neg L^f \wedge \neg L^t]; L^e\uparrow]$$

The PCHB template has a latency of two transitions and a cycle time of 14 transitions. The main difference from the WCHB is that the neutrality of the inputs is detected on $L^e\downarrow$ rather than $R\downarrow$. As a result, input neutrality can be detected in multiple transitions without impacting latency, which allows for a greater number of inputs. In a WCHB, neutrality is detected in the pull-up stack of the data rails. This limits the number of inputs possible in a WCHB because more inputs means more series PMOS transistors in the pull-up stack. Generally, we limit ourselves to three series PMOS in any stack. Any more and the rise time will be poor and charge sharing in dynamic nodes becomes problematic.

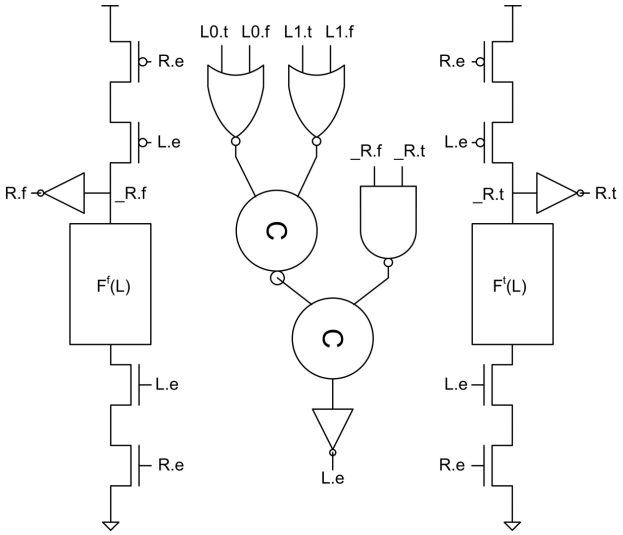

Fig. 2. A two input and one output PCHB template.

An extension to the PCHB is the PCEHB. In the PCEHB, R^e and L^e are combined in a separate c-element. This improves the latency by reducing the number of series PMOS and NMOS in the data rail stacks by one. However, the PCEHB increases the cycle time by four transitions (a non-inverting c-element is two transitions and it needs to be set and reset in one cycle) making the total 18 transitions.

In general, we try to stick to the following guidelines for high performance QDI logic:

1) Avoid three transition or less cutoff paths so that signals are full swing.
2) Keep the latency of each stage to two transistions.
3) Keep the cycle time of each stage within 18 transistions.
4) Dynamic nodes cannot be directly shared between stages. These signals must be buffered first.
5) The output of all state holding logic is staticized (held by weak feedback).
6) Wires in channels are fully shielded.

Guideline 3 may be flexible depending on the application. We often find that even with a worst case cycle time of 18 transitions the frequency of the system is limited by the latency of loops with a suboptimal number of tokens. For this reason we place greater priority on optimizing latency over cycle time. Guideline 4 is meant to prevent bit flips on dynamic nodes. Dynamic nodes are more susceptible to crosstalk because there are times that they are only driven by weak feedback.

B. Half Cycle Timing Assumption

The WCHB, PCHB, and PCEHB circuit templates (and QDI circuits in general) are highly tolerant of process variations because each up and down transition is sensed. The only timing assumption allowed in QDI design is the isochronic fork assumption [7]. This timing assumption states that the

difference in delay between branches of a wire is insignificant compared to the gate delays of the logic reading their values.

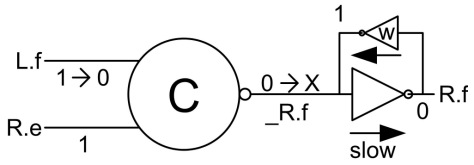

Fig. 3. An error that can occur in QDI logic without the half cycle timing assumption.

Upon closer inspection, however, there is a second timing assumption that is quite common in QDI circuits. Observe the false rail of a WCHB shown in Figure 3. In order for this circuit to work properly a timing assumption is made with respect to its staticizer. Let us assume that the inverter driving R^f is incredibly slow. $\_R^f\downarrow$ has fired, due to $L^f\uparrow$ and $R^e\uparrow$, but $R^f\uparrow$ has not. When $L.f\downarrow$ fires, the c-element becomes state holding and the only active current is the weak feedback. Even though the inverter is weak, it can flip $\_R^f$ because there is no opposing current. The resulting error is due to an actual analog problem and not an isochronic fork.

To avoid such timing errors with staticizers we introduce the half cycle timing assumption. The half cycle timing assumption (HCTA) is a local timing assumption (internal to a process) that assumes a small amount of combinational logic (one or two transitions) will always switch within one half cycle of a process. With cycle times of 10-18 transitions, this assumption has a timing margin of 2.5x-4.5x. In addition, during the half cycle communication occurs across the channels where wires tend to be longer than wires internal to a process. Transitions across these wires will be slower, making the half cycle even longer compared to the two transition logic.

In QDI, the HCTA is only needed to guarantee the correct operation of staticizers. We can reduce logic and design new valid circuits by extending the HCTA for general use. We refer to the resulting logic as relaxed quasi-delay insensitive (RQDI). RQDI logic has a robustness similar to QDI logic because they both use the same timing assumptions and have the same timing margins. Table I compares the timing margins across different circuit families. QDI and RQDI exhibit extremely large timing margins without any impact on their latency or cycle times. A tremendous increase in latency and cycle time is required to get similar timing margins with synchronous or bundled data logic.

C. HCHB Circuit Template

We can reduce the logic needed to compute the neutrality in logic templates by applying the HCTA. Consider the following HSE for the half cycle half buffer (HCHB):

$$*[([R^e \wedge L^f \longrightarrow R^f\uparrow [] R^e \wedge L^t \longrightarrow R^t\uparrow]; L^e\downarrow;), N\downarrow; [\neg R^e], ([\neg L^f \wedge \neg L^t]; N\uparrow); R^f\downarrow, R^t\downarrow; L^e\uparrow]$$

We've introduced a variable N, for neutrality, with the intention of only sensing the $N\uparrow$ transition. N detects the neutrality of L and it can be implemented as the nor of L^f and L^t. $N\downarrow$ can fire at the beginning of the evaluation phase (first line of HSE) when L becomes valid, but doesn't need to fire until the beginning of the reset phase (second line of HSE) before $R^e\downarrow$ arrives, a half cycle later. We make the assumption that $N\downarrow$ will fire before the second half of the cycle and add no logic to detect this transition.

Circuit Family	Timing Margin	Tradeoff
Synchronous	m	mL, mC
Bundled Data (two phase)	m	mL, mC
Bundled Data (four phase)	m	mL, $2mC$
QDI (staticizers)	$2.5x - 4.5x$	none
RQDI	$2.5x - 4.5x$	none

TABLE I
TIMING MARGINS ASSOCIATED WITH VARIOUS CIRCUIT FAMILIES. SYMBOLS m, L, AND C ARE THE TIMING MARGIN FACTOR, LATENCY, AND CYCLE TIME RESPECTIVELY.

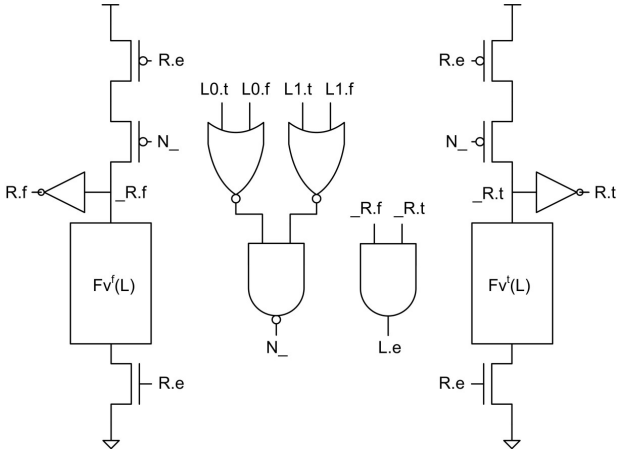

Fig. 4. A two input and one output hchb template.

Applying the half cycle timing assumption results in the HCHB template shown in Figure 4. Validity and neutrality are checked in the data rails similar to the WCHB. This reduces the logic for L^e and gets rid of one series NMOS in the pull down stack. It takes two transitions to detect the neutrality of the inputs. This does not affect the two transition latency of the circuit, but makes the cycle time 14 transitions. An additional requirement of the HCHB over the PCHB is that the pull down networks of the data rails need to wait for all the inputs to become valid before firing. In some cases the pull down stacks already wait for validity. In other cases the pull down stacks need to be augmented to wait for input validity.

Figure 5 shows the false rails of a PCHB and HCHB and2 process and their corresponding transistor widths. The false rail of the HCHB has been extended to wait for the validity of both $L0$ and $L1$. The HCHB pull down stack has two more transistors than the PCHB, but area saved elsewhere in the circuit more than makes up the difference. In some circuits, e.g. a full adder, the pull down stack will already guarantee input validity with no additional effort. In circuits with many inputs, the validity can be checked in a separate single rail

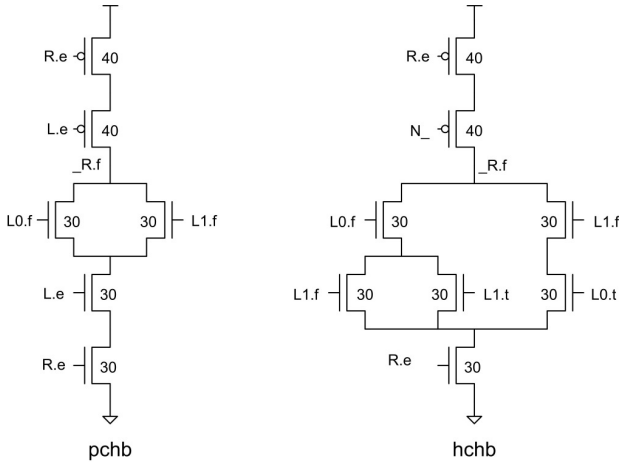

Fig. 5. The false rail stacks of an and2 process for a PCHB(left) and a HCHB(right). The numbers are the transistor widths in lambda units (half minimum gate length).

pseudo output.

III. Voltage Scaling

A high level discussion on voltage scaling in asynchronous architectures can be found in [5]. Energy efficient pipelines are discussed in [13]. In this section we aim to facilitate voltage scaling in asynchronous circuits by: 1)introducing a pair of efficient voltage converters, and ii) proposing a dual voltage circuit template that has constant latency.

A. Efficient Voltage Converters

The standard low to high voltage converter is shown in Figure 6. The input signal in has a voltage range from GND to V_{DDL}. This signal is not directly used in a pull up network because if V_{DDL} is less than $V_{DD} - V_{th}$ then in cannot turn off PMOS transistors in the V_{DD} domain. Instead, in and its inverted version are fed into the pull down NMOS transistors of a cross coupled PMOS structure. When one of the NMOS transistors becomes active it begins to discharge its output node. A short circuit then exists between the NMOS and PMOS transistors. If the NMOS transistor is sized correctly, it will win the fight with the PMOS transistor and eventually both cross coupled nodes will switch. Higher voltage signals can be used freely in lower voltage domains, therefore a high to low converter is not needed.

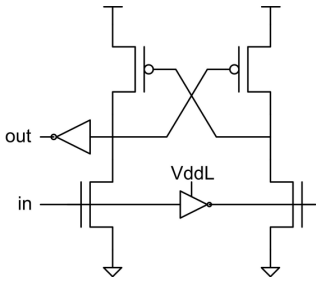

Fig. 6. The standard voltage converter.

In asynchronous circuits, we will be converting voltage across channels rather than across simple signals. For dual rail codes, we have three signals to convert: the two data rails and the enable rail (acknowledge). In channels going from a lower to higher voltage domain, the data rails need to be converted to the higher voltage. In channels going from a higher to lower voltage domain, the enable rail needs to converted to the higher voltage.

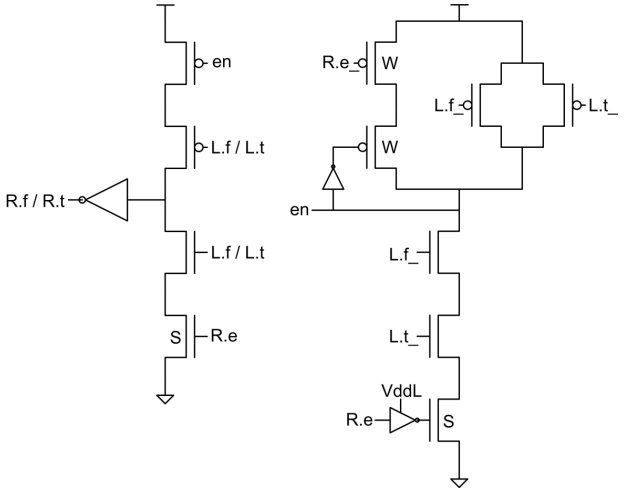

Fig. 7. A pipelined high voltage to low voltage dual rail converter. The pull up feedback on the right stack is explicitly shown.

The short circuit that occurs in the conventional voltage converter can be avoided by guarding the conversion with high voltage signals that are available in the handshake. The following is the HSE for high to low voltage converter:

$$*[([R^e \wedge L^f \longrightarrow R^f\uparrow [] R^e \wedge L^t \longrightarrow R^t\uparrow]; L^e\downarrow;), en\downarrow; \\ [R\_^e \wedge L\_^f \wedge L\_^t]; en\uparrow); R^f\downarrow, R^t\downarrow; L^e\uparrow]$$

This HSE is similar to the HCHB. The main difference is that we use inverted versions of the R^e, L^f, and L^t. Conveniently, the half cycle timing assumption allows us to use inverted versions of signals without having to check each transition on the inverted and non-inverted version of the signal (this is not possible in pure QDI circuits).

The high to low converter is shown in Figure 7. $R^e\downarrow$ is sensed in the en stack and $R^e\uparrow$ is sensed in the R^f and R^t stacks. The short circuit found in the conventional converter is avoided by guarding the stacks with L^f and L^t and their inverted versions. This only leaves the short circuit caused by the weak feedback that is common to all state holding gates. We can mitigate the impact of the weak feedback by adding a weak series PMOS for $R\_^e$ in en's feedback and R^e in R^f and R^t's feedback. An example of this is shown in the pull up feedback for en. When $R\_^e$ goes to V_{DDL}, and $L\_^f$/$L\_^t$ are high, the bottom transistor partially turns on. At the same time, the top weak PMOS partially turns off which reduces the weak short circuit current. The transistors marked with a S are made strong by using a low threshold voltage transistor and oversizing it. The low to high converter is similar as shown

in Figure 8.

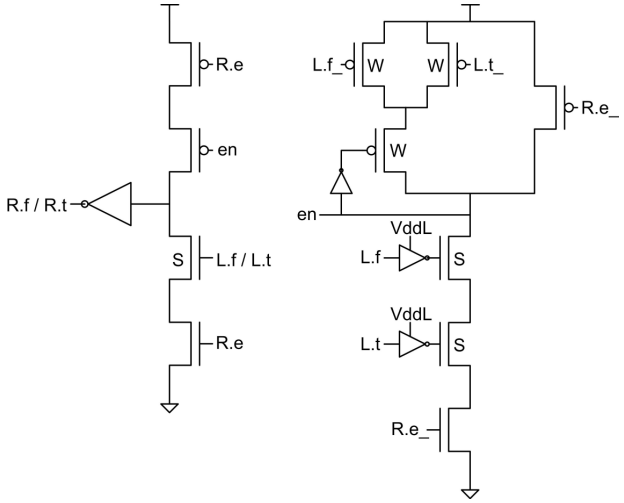

Fig. 8. A pipelined low voltage to high voltage dual rail converter. The pull up feedback on the right stack is explicitly shown.

B. DVHB Circuit Template

Simply scaling the voltage in an asynchronous circuit will increase both its cycle time and latency. The latency can be kept constant by keeping the logic in the forward path in the high voltage domain and moving only the logic in the return path to the low voltage domain. In other words, the data rail stacks use V_{DD} and the enable/acknowledge logic uses V_{DDL}. A voltage conversion is needed whenever the data rail logic uses a signal from the enable logic.

To minimize the forward latency, we start with a PCEHB reshuffling:

$$*[[R^e]; en\uparrow; [L^f \longrightarrow R^f\uparrow [] L^t \longrightarrow R^t\uparrow]; L^e\downarrow; \\ [\neg R^e]; en\downarrow; R^f\downarrow, R^t\downarrow; [\neg L^f \wedge \neg L^t]; L^e\uparrow]$$

Both the R^e and L^e signals are in the low voltage domain. The PCEHB reshuffling allows us to remove these signals from the data rails and do the voltage conversion in the logic for en. Similar to the voltage converters in the previous subsection, we need to find a high voltage signal to use as a guard in the en logic. The only choice is to use the validity of the data rails, Rv. The dual voltage half buffer circuit (DVHB) is depicted in Figure 9. The shaded logic is in the low voltage domain. The voltage conversion occurs in the $en0$ and en stacks. We use the same technique as before to lessen the weak feedback during the conversion.

Although the shaded logic seems to be a small part of the circuit, this logic switches every cycle compared to the data rails where only one rail switches per cycle. In addition, by making L^e and R^e low voltage we have reduced the switching in the channels where wires tend to be longer and more capacitive. Moreover, the en stack, $en0$ stack and the data rails only have one series PMOS transistor which helps to limit their output capacitance (the weak feedback PMOS transistors

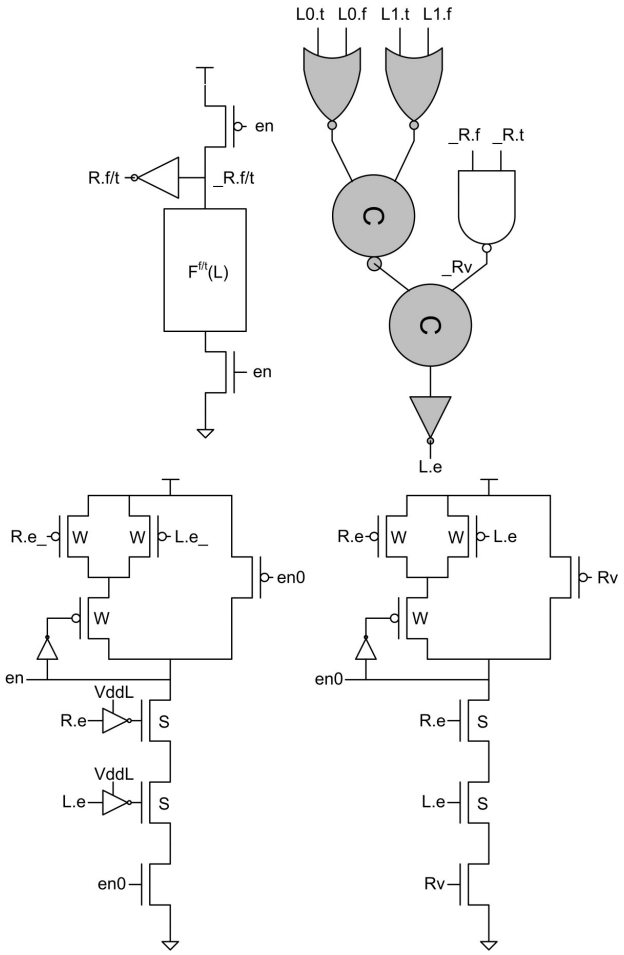

Fig. 9. The DVHB circuit template. The shaded logic is in a lower voltage domain.

are minimum size and do not contribute greatly to the output capacitance).

IV. Two Phase Static Switching Networks

Two phase handshake protocols are often suggested to reduce power and increase frequency in asynchronous circuits. The main difficulty with two phase protocols is that they are very inefficient in performing logic functions, as has been noted by others [8]. However, a simple two phase buffer with similar characteristics to a WCHB (two transition forward latency and ten or less transition cycle time) would be useful in two specific applications. The first, and most obvious, is global communication. The second application is static switching networks.

Static switching networks are especially important in FPGAs [12]. Logic clusters in FPGAs are surrounded by statically configured switching networks. In asynchronous FPGAs, these statically configured switching networks are built up out of the switches shown in Figure 10. Programming bits are set to select which set of input data rails are the input to the WCHB via the MUX. The output data rails of the WCHB fanout to other switches and the associated

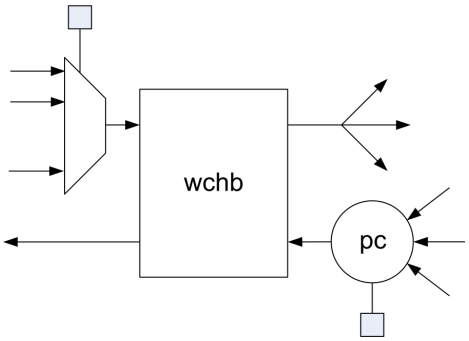

Fig. 10. A statically programmed n-way switch.

enables/acknowledges must be combined via a programmable c-element (depicted as pc in the diagram). We can replace the WCHB with a two phase buffer and the switch will work without any modification.

A. HC2PFB Buffer

A simple dataless two phase buffer can be represented by the following HSE:

$$*[[L = R^e]; R := L; L^e := \neg R]$$

The resulting circuit is a c-element and an inverter, shown in Figure 11. The circuit gets more complicated when you add data. We will assume a simple protocol where sending a zero means a transition on the false rail and sending a one means a transition on the true rail (although the resulting buffer can support the LEDR [1] protocol as well). The problem is that R^e changes each cycle and each data rail needs to know which sense of R^e to wait for. We can introduce a state variable to track this, but it would be expensive to manage it. A better solution is for each rail to wait for the XOR of the opposite rail with R^e, instead of R^e. The idea is that if the opposite rail caused R^e to change then the output of the XOR will be unchanged since both of its inputs have switched (in reality the output will switch and then switch back). The HSE for the two phase data buffer is:

$$\begin{array}{l} *[[L^f = XOR(R^t, R^e) \longrightarrow R^f := L^f \\ \quad []L^t = XOR(R^f, R^e) \longrightarrow R^t := L^t] \\ \quad L^e := XOR(R^f, R^t)] \end{array}$$

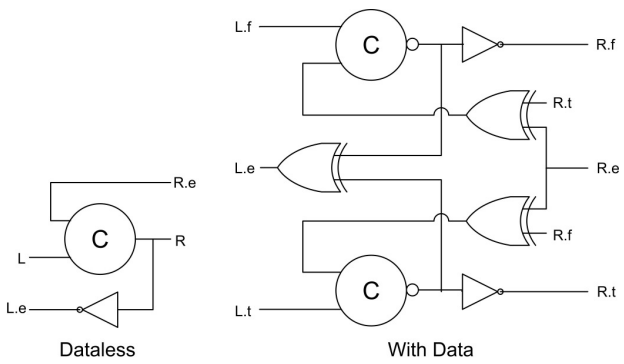

Fig. 11. The HC2PFB buffer: dataless (left) and with data (right).

The HC2PFB (half cycle two phase full buffer) buffer is shown in Figure 11. The HC2PFB has a forward latency of two transitions and a cycle time of seven transitions. The pair of XOR gates that process R^e can be folded into the c-elements, but this makes the data rails more complex and increases the latency. The HC2PFB is 45% larger than the WCHB, however, since the HC2PFB has such a short cycle time the XOR gates can be undersized to reduce the area penalty. In addition, each HC2PFB can replace two stages of WCHBs because it can support twice the number of transitions in a cycle. The slack will remain the same because we are replacing two half buffers with a full buffer. When used in this fashion, the HC2PFB equivalent circuit is 15% smaller than the WCHB. This is an important result because the four-to-two and two-to-four phase converters are significantly larger than the buffer.

B. 4:2 Converter

Consider the following HSE for the four-to-two converter:

$$\begin{array}{l} *[[L^f \longrightarrow R^f := XOR(R^t, en) \\ \quad []L^t \longrightarrow R^t := XOR(R^f, en)]; L^e\downarrow; \\ \quad [\neg L^f \wedge \neg L^t]; en := R^e; L^e\uparrow] \end{array}$$

When one of the input data rails goes high, the corresponding output data rail needs to toggle which is equivalent to setting the rail to the XOR of the opposite rail and R^e. However, using R^e directly can cause an oscillation on the data rails. For example, assume $L^f\uparrow$ causes $R^f\uparrow$ which in turn causes $R^e\downarrow$. If $R^e\downarrow$ arrives before $L^f\downarrow$, then $R^f\downarrow$ might fire. To prevent this scenario, we introduce en to store the value of R^e at the beginning of the cycle.

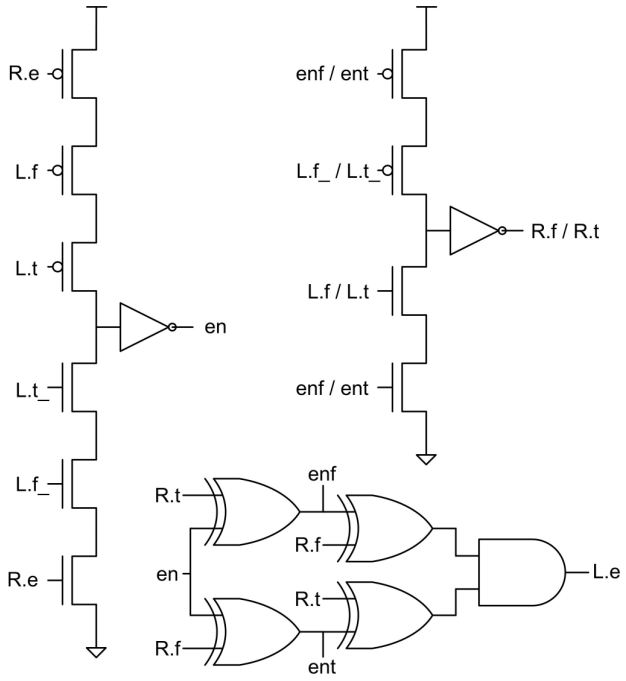

Fig. 12. A four-to-two phase converter converter.

The four-to-two phase converter is shown in Figure 12. The XOR gates that generate enf and ent are equivalent to the ones used in the buffer. Output validity is detected through a pair of XOR gates and an and gate. The assignments in the HSE have been implemented with latches. The timing constraints of the latches can be accounted for by the handshake and the half cycle timing assumption, with the exception of the hold time on enf/ent for the R^f/R^t nodes. This hold time is set by the leftmost pair of XOR gates. Note, this is the equivalent of a three transition cutoff path in QDI circuits (see Guideline 1 at the end of Section 2). The solution is to implement these XORs as XNORs followed by an inverter. The latency of the converter is three transitions in the worst case because of the need to invert L^f and L^t. The four-to-two phase converter is about $3x$ larger than a WCHB.

C. 2:4 Converter

The following is the HSE for the two-to-four converter:

$$
\begin{array}{l}
*[[R^e \wedge L^f = XOR(R^t, en) \longrightarrow R^f\uparrow \\
\quad []R^e \wedge L^t = XOR(R^f, en) \longrightarrow R^t\uparrow]; \\
\quad en := \neg en; L^e := en; \\
\quad [\neg R^e]; R^f\downarrow, R^t\downarrow]
\end{array}
$$

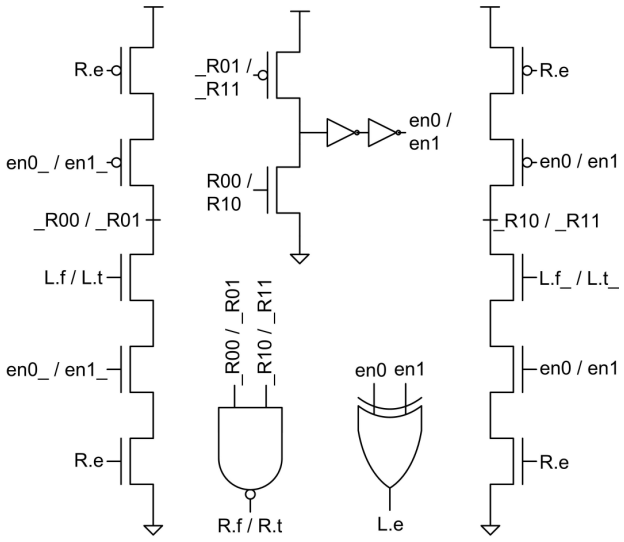

Fig. 13. A two-to-four phase converter.

It is difficult to generate the toggle of en based on the values of R^f and R^t alone. They don't contain any information about which sense of L^f and L^t caused them to fire. To remedy this, we implement each data rail with two state holding gates followed by a NAND gate, as shown in Figure 13. (This technique is sometimes used in QDI circuits to break up the load in a complicated pull down stack.) One node fires when the transition is caused by $L^f\uparrow$ or $L^t\uparrow$ and the other fires when the transition is caused by $L^f\downarrow$ or $L^t\downarrow$. With this information, we generate a pair of signals, $en0$ and $en1$, which act like two phase data rails. Similar to the buffer, L^e can be set based on the XOR of these signals. This converter also has a forward latency of three transitions due to inverting the input data rails. It is roughly $3.25x$ larger than the WCHB circuit.

V. Evaluation Setup

All simulations are done with HSpice using model files for a 65 nm process. Wire capacitances are approximated by adding a 4fF capacitance to each output node. This amount of capacitance is typical of short wires based on our observations of extracted layout in this technology. Gates are sized to have the drive strength of an inverter with its pmos width set to 20 lambda units and its nmos width set to 10 lambda units (lambda is defined as half the minimum gate length). All power and energy numbers are based on total dissipated power.

Name	Inputs	Outputs	Description
and2	2	1	and gate
or2	2	1	or gate
xor2	2	1	exclusive or
fa	3	2	full adder
benc	3	2	booth encoder

TABLE II
Benchmark circuits used in evaluation. Note: these are dual-rail pipelined circuits.

The benchmark circuits used in our evaluations are listed in Table II. Latency and cycle time numbers reported represent the worst case. We measure the worst case by switching the data rail with the slower stack. For example, the true rail in the and2 circuit has one extra series nmos transistor, therefore we exercise that stack in its simulations. The area reported is the total transistor area of a circuit (the sum of $width*length$ of each transistor).

VI. Results

A. HCHB Template

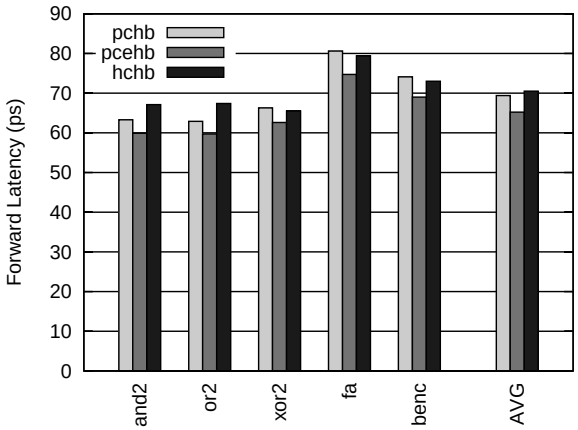

Fig. 14. Forward latency of benchmark circuits across PCHB, PCEHB, and HCHB templates.

The latency of the five benchmark circuits for the PCHB, PCEHB, and HCHB templates are shown in Figure 14. On average, the latency of the PCEHB is 6% less than the other

circuit templates. The PCEHB is generally lower latency because R^e and L^e are combined in a separate c-element, rather than in the data rail stacks. The HCHB has a similar latency to the PCHB except and2 and or2 circuits where it's 6% slower. The pull down stacks in these circuits were augmented to wait for input validity, which makes them slower.

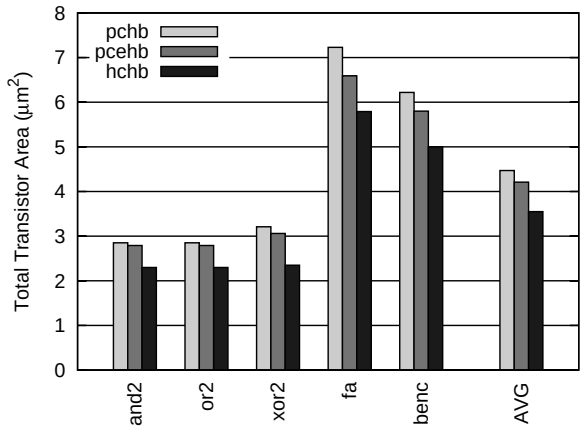

Fig. 15. Total transistor area of benchmark circuits across PCHB, PCEHB, and HCHB templates.

Figure 15 shows a comparison of the total transistor area across the benchmark circuits. An interesting result is that the PCEHB is slightly smaller than the PCHB. Once again, this is attributed to its simpler data rail transistor stacks. The HCHB is about 15% smaller than the PCEHB template and 20% smaller than the PCHB template on average. This is a result of the simplified detection of input neutrality possible with the half cycle timing assumption.

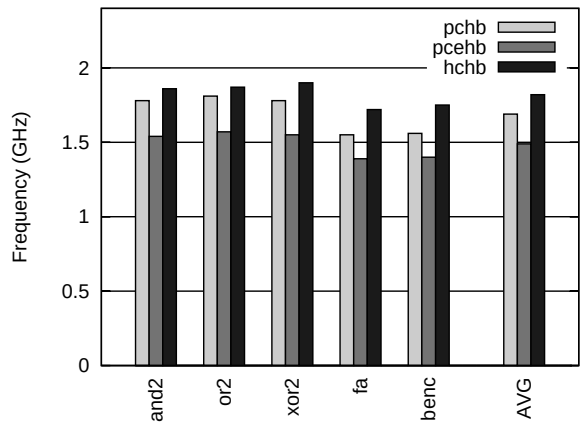

Fig. 16. Frequency of benchmark circuits across PCHB, PCEHB, and HCHB templates.

The HCHB template is consistently higher frequency than the other templates across all five benchmark circuits, as seen in Figure 16. On average, the HCHB is 7% higher frequency than the PCHB. The PCEHB has an 18 transition cycle time and the HCHB and PCHB both have a 14 transition cycle time. However, the HCHB is higher frequency because many of its transitions, especially those that detect input neutrality, are simpler. This suggests that HCHB can use even less area because we can use smaller transistors for these fast transitions to match the frequency of the PCHB.

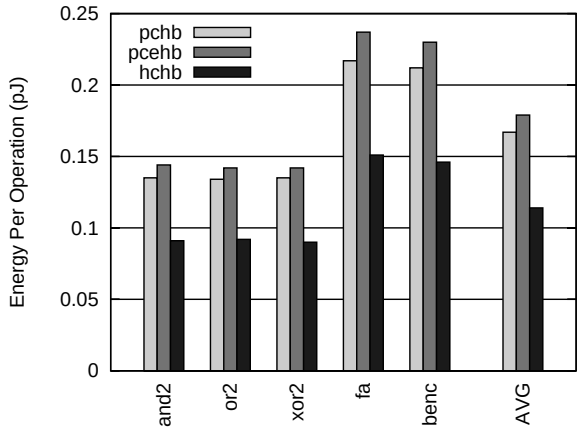

Fig. 17. Energy per operation of benchmark circuits across PCHB, PCEHB, and HCHB templates.

The energy per operation (or per cycle) of the benchmark circuits is reported in Figure 17. The HCHB template consistently uses less energy than the PCHB and PCEHB templates across all five benchmarks. The HCHB template consumes 32% and 36% less energy on average than the PCHB and PCEHB templates respectively. This is a great result because it is accompanied by significant area savings, a slight frequency improvement, and a negligible latency penalty.

B. Voltage Scaling

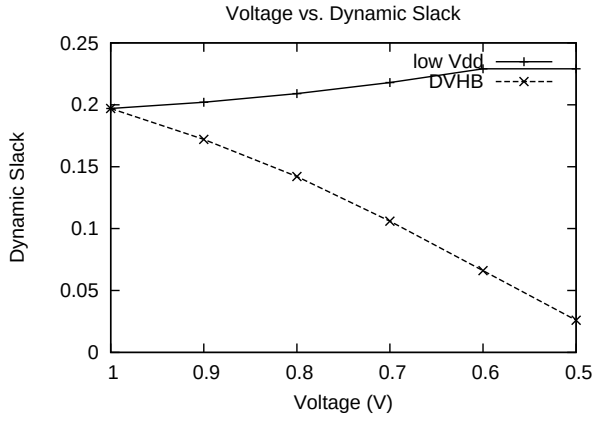

Fig. 18. Average dynamic slack of low V_{DD} and DVHB templates across benchmark circuits.

The average dynamic slack of low V_{DD} and DVHB templates across benchmark circuits is shown in Figure 18. The dynamic slack is calculated as twice the forward latency over the cycle time (for half buffers). The dynamic slack remains relatively constant for V_{DD} scaling. In the DVHB, the forward latency remains constant while the cycle time increases which results in a decreasing dynamic slack. The throughput-optimal number of tokens in a pipeline is proportional to the dynamic

slack. This implies that the throughput of loops with a less than optimal number of tokens will improve with the DVHB template relative to simply scaling V_{DD}. In such loops, the voltage of the enable logic can be scaled somewhat without impacting performance.

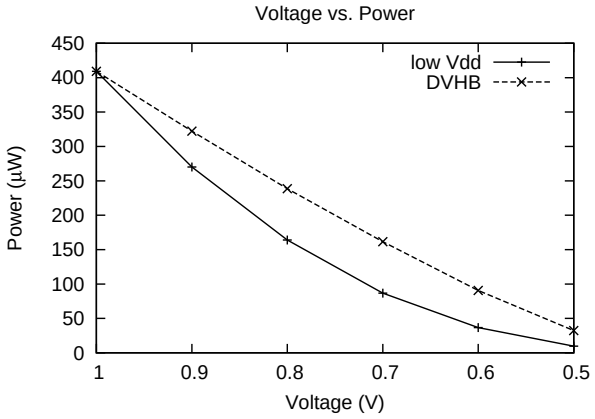

Fig. 19. Average power across benchmark circuits for low V_{DD} and DVHB templates.

Figure 19 displays the average power across benchmark circuits for low V_{DD} and DVHB templates. The V_{DD} scaled circuit exhibits the expected cubic decrease in power with supply voltage. The DVHB template doesn't scale as well as pure V_{DD} scaling. The power doesn't scale as well because its data rails, forward path logic, and voltage converters remain in the high voltage domain.

C. Two Phase Static Switching

In asynchronous FPGAs, programmable routing between logic clusters is made up of stages of the static switch shown in Figure 10. In this experiment, we make this routing use two phase logic by replacing the WCHB with an HC2PFB. In fact, we replace two stages of the WCHB switch with one stage of the HC2PFB switch. This keeps the slack, latency, area, and cycle time roughly constant between the two implementations. There is some overhead incurred from the 4:2 and 2:4 phase converters at the input and output of the routing logic. We vary the number of two phase pipeline stages between these converters and measure the area impact and energy reduction over the original WCHB version (where no converters are needed). In addition, we vary the width of the individual switches.

Figure 20 shows the energy reduction in the two phase static switch with increasing switch width. As the switch width increases, more static muxing and programmable c-elements are need to build the switch. As a result, more capacitance is switching each cycle. Intuitively, one would think that the maximum energy reduction would be 50%. However, each two phase buffer replaces two four phase buffers. Even in this configuration, the two phase buffer is higher frequency. At a switch width of 16 there is over a 52% reduction in energy.

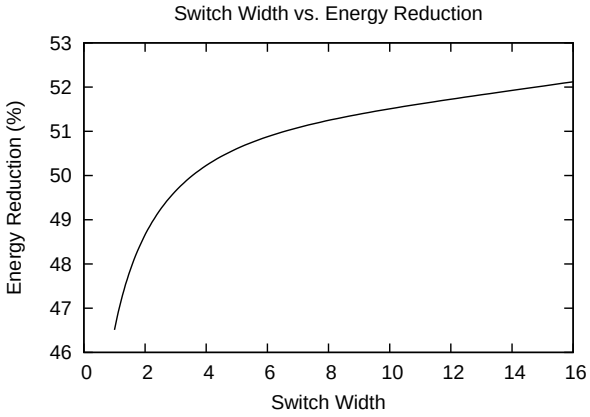

Fig. 20. Energy reduction as switch width increases.

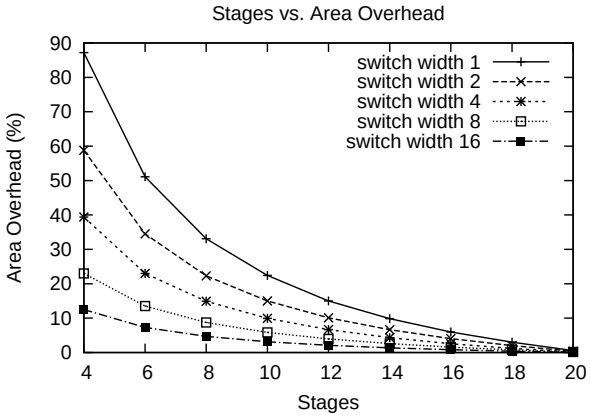

Fig. 21. Area overhead as the number of stages increase.

The main drawback to using the two phase switch is the high cost of converting between two phase and four phase protocols. The two phase buffer is about 15% smaller than the two four phase buffers it replaces. The four-to-two phase converter is $3x$ larger than a WCHB and the two-to-four phase converter is $3.25x$ larger than a WCHB. Figure 21 tracks the area overhead of 20 stages of switches with varying widths. In an asynchronous FPGA, neighboring logic clusters are typically separated by 6-8 stages of 4-wide switches. This would put the area overhead at 15-20%. If the FPGA has direct connections between neighboring logic clusters (a common optimization) then the minimum distance between two logic clusters that use the routing network would be 10-12 stages. In this case, the area overhead would drop to about 6-10%. In addition, some of inputs are actually copies. The number of input converters can be reduced by pushing this copying into the two phase logic.

VII. Related Work

Optimizing circuit templates by applying timing assumptions has been explored in previous work. In [9], [11], an additional wire is added to the data channel to share validity and neutrality information between stages. A similar technique can be applied to the HCHB to reduce logic for neutrality

detection. We have chosen not to add an additional wire to the data channels for two reasons. One, successive stages of logic are often not immediately adjacent to one another. When they aren't, channels are longer and the energy consumed in switching them can account for over 30% of the total energy for a stage. An additional wire would increase the switching in the channels by 50%. Two, in an FPGA this wire would have to be statically routed with the rest of the wires in a channel. In this case, the additional muxing required would overshadow any potential area savings.

A set of efficient protocol converters are presented in [8]. Similar to this work, four phase logic is used for computation and two phase logic is used for communication. A major difference from our work is that the four phase logic only supports a single data token at a time. The 2:4 and 4:2 converters share control logic. The input converter will not receive another token until the current token has left the output converter. In addition, more aggressive timing assumptions are used to generate clock pulses used in flops.

A two phase buffer for asynchronous interconnect is proposed in [10]. The latency of that buffer is roughly ten transitions which five times larger than the buffer presented in this work. Again, more aggressive timing assumptions are used to generate clock pulses that trigger flops for the data rails and the acknowledge.

VIII. Future Work

The circuits in this paper were design with the intention of using them to reduce energy in asynchronous FPGAs. The routing logic in an asynchronous FPGA is larger and more power hungry than a typical synchronous FPGA for two reasons. One, the capacitance in the wires is switched four times per cycle with four phase protocols. Two, the copying that is ubiquitous in the routing logic requires programmable c-elements to combine the associated acknowledge/enable signals. As a result, the routing logic can easily account for over 50% of the entire chip area. Using the methods described in this paper to switch to a two phase protocol has the potential to yield rather large power savings.

The voltage scaling templates described in this paper also have the potential to save power in an FPGA. User designs are mapped to an FPGA through a chain of synthesis, place, and route tools. The resulting design contains many reconvergent paths and latency limited loops. These scenarios allow for voltage scaling with minimal impact on performance. We are currently in the process of designing an FPGA utilizing the circuits discussed in this paper.

IX. Conclusions

We have presented a class of circuits that are derived by starting with quasi delay-insensitive circuits and applying a conservative timing assumption, namely the half cycle timing assumption. We refer to these as relaxed quasi delay-insensitive circuits. We used these circuits to help reduce power consumption in a few ways. First, we developed the half cycle half buffer (HCHB) circuit template that reduces the amount logic needed to generate enable/acknowledge signals. The HCHB template reduces area by 15% and energy by 32% on average across our benchmark circuits. Second, we showed how to fold voltage converters into the HCHB buffer. We also proposed the dual voltage half buffer (DVHB) to allow voltage scaling on the enable/acknowledge logic (return path) while keeping the data logic (forward path) in a high voltage domain to maintain a constant forward latency. Third, we presented a two phase buffer for use in global communication and static switching networks. This buffer was shown to reduce energy in static switches by over 50%. The overhead of a four-way switch over ten stages was shown to be about 10%. This overhead results from the four-to-two and two-to-four phase converters, and it decreases as more stages are added.

References

[1] Mark E. Dean, Ted E. Williams, and David L. Dill. Efficient self-timing with level-encoded 2-phase dual-rail (ledr). In *Proceedings of the 1991 University of California/Santa Cruz conference on Advanced research in VLSI*, pages 55–70, Cambridge, MA, USA, 1991. MIT Press.

[2] Michael K. Gowan, Larry L. Biro, and Daniel B. Jackson. Power considerations in the design of the alpha 21264 microprocessor. In *35th Design Automation Conference*, pages 726–731, 1998.

[3] Mark Horowitz. Scaling, power and the future of cmos. In *VLSID '07: Proceedings of the 20th International Conference on VLSI Design held jointly with 6th International Conference*, page 23, Washington, DC, USA, 2007. IEEE Computer Society.

[4] Andrew Matthew Lines. Pipelined asynchronous circuits. Master's thesis, California Institute of Technology, 1996.

[5] R. Manohar and M. Nystrom. Implications of voltage scaling in asynchronous architectures. Technical report, Cornell Computer Systems Lab CSL-TR-2001-1013, April 2001.

[6] Alain J. Martin. Compiling communicating processes into delay-insensitive vlsi circuits. *Distributed Computing*, 1(4), 1986.

[7] Alain J. Martin. Programming in vlsi: from communicating processes to delay-insensitive circuits. pages 1–64, 1990.

[8] Amitava Mitra, William F. McLaughlin, and Steven M. Nowick. Efficient asynchronous protocol converters for two-phase delay-insensitive global communication. In *ASYNC '07: Proceedings of the 13th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 186–195, Washington, DC, USA, 2007. IEEE Computer Society.

[9] Recep O. Ozdag and Peter A. Beerel. High-speed qdi asynchronous pipelines. In *ASYNC '02: Proceedings of the 8th International Symposium on Asynchronus Circuits and Systems*, page 13, Washington, DC, USA, 2002. IEEE Computer Society.

[10] Bradley R. Quinton, Mark R. Greenstreet, and Steven J. E. Wilton. Practical asynchronous interconnect network design. *IEEE Transactions Very Large Scale Integration Systems*, 16(5):579–588, 2008.

[11] Montek Singh and Steven M. Nowick. High-throughput asynchronous pipelines for fine-grain dynamic datapaths. In *ASYNC '00: Proceedings of the 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, page 198, Washington, DC, USA, 2000. IEEE Computer Society.

[12] John Teifel and Rajit Manohar. Highly pipelined asynchronous fpgas. In *FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pages 133–142, New York, NY, USA, 2004. ACM.

[13] John Teifel, Rajit Manohar, David Fang, Clint Kelly, and David Biermann. Energy-efficient pipelines. In *ASYNC '02: Proceedings of the 8th International Symposium on Asynchronus Circuits and Systems*, page 23, Washington, DC, USA, 2002. IEEE Computer Society.