Towards PVT-Tolerant Glitch-Free Operation in FPGAs

Safeen Huda and Jason Anderson
Dept. of Electrical and Computer Engineering,
University of Toronto
Toronto, ON, Canada
Email: {safeen,janders}@ece.toronto.edu

ABSTRACT

Glitches are unnecessary transitions on logic signals that needlessly consume dynamic power. Glitches arise from imbalances in the combinational path delays to a signal, which may cause the signal to toggle multiple times in a given clock cycle before settling to its final value. In this paper, we propose a low-cost circuit structure that is able to eliminate a majority of glitches. The structure, which is incorporated into the output buffers of FPGA logic elements, suppresses pulses on buffer outputs whose duration is shorter than a configurable time window (set at the time of FPGA configuration). Glitches are thereby eliminated "at the source" ensuring they do not propagate into the high-capacitance FPGA interconnect, saving power. An experimental study, using Altera commercial tools for power analysis, demonstrates that the proposed technique reduces 70% of glitches, at a cost of 1% reduction in speed performance.

1. INTRODUCTION

In recent years, field-programmable gate arrays (FPGAs) have become increasingly popular platforms for the implementation of digital systems, as reflected in the increased market share FPGA vendors have enjoyed in the semiconductor industry. However, it has previously been shown that there is a large gap between FPGAs and the alternative medium for the implementation of digital systems, ASICs [1]. While FPGAs (vs. ASICs) suffer from deficiencies in area efficiency and performance, it is the large power consumption – 7-14× as claimed in a recent study [1] – that has particularly inhibited the adoption of FPGAs in a wide variety of current and emerging applications that require strict power budgets. In this paper, we propose a technique to reduce a component of FPGA dynamic power, namely, power dissipated due to *glitches*.

Underscoring the importance of reducing FPGA power, the vendors have adopted a variety of techniques to tackle power consumption at the device, circuit, and architectural levels, and through CAD techniques as well [2, 3, 4]. One particular power optimization, Altera's *Programmable Power Technology* [5], makes use of the fact that a transistor's threshold voltage, V_T can be altered by application of a bias voltage at the base terminal. An increase in $|V_T|$ (by applying a base-terminal (body) bias) results in reduced static power of a device, at the expense of increased delay. However, since designs implemented on FPGAs typically have a large number of paths with considerable timing slack [6], this technique can be used to reduce the static power of circuitry on such paths. Unfortunately, with the vendors transitioning to FinFETs [7, 8] which do not permit independent body bias control, the future of this technique appears to be limited. Nevertheless, the notion of using excess timing slack to trade-off overall power with delay appears to be a very effective means of power reduction – and is one which we exploit in this work.

We take aim at glitch power in FPGAs, which has previously been shown to account for a significant portion of the total dynamic power dissipated, with one study finding that glitches account for ∼26% of total core dynamic power [9]. We propose novel glitch filtering circuitry which serves to completely eliminate glitches of a given pulse-width. The circuitry is incorporated into the buffers present at logic element outputs. Glitches are eliminated immediately after they are generated, and most importantly, before they can propagate into the high-capacitance programmable interconnection network, where they would otherwise result in significant energy waste. We also propose an optimization algorithm to maximize the glitch power reduction (by applying appropriate settings on each glitch filter), subject to timing constraints. We present a full CAD flow which we use to assess the merits of our power reduction technique. Experiments show that glitch power can be reduced by up to ∼70% at an area cost of < 3%, with an average critical-path degradation of ∼1%. We provide an overview of glitch power in FPGAs and previous techniques proposed to reduce glitches in Section 2. Section 3 describes our proposed glitch filtering circuit. Section 4 provides an overview of our CAD flow and glitch optimization algorithm. Section 5 describes our experimental study and presents results. Finally, Section 6 concludes the paper.

2. BACKGROUND

2.1 Glitch Power Dissipation in FPGAs

Glitches commonly occur in digital circuits, as a consequence of unequal arrival times at the inputs of combinational logic gates, such as the scenario depicted in Figure 1, where input *A* transitions after input *B*. The period of time between the two input transitions shown in the figure can potentially give rise to spurious transitions at the output – i.e. glitches – which have no functional value, and are a waste of energy. Each transition consumes CV_{DD}^2 joules of energy, where *C* is the capacitance of the net being driven by OUT.

Glitches are especially troublesome in FPGAs. Whereas in ASICs there exists the freedom to minimize the disparity between the delays of different paths (and thereby ensure that the arrival times of the input signals to a combinational circuit are well matched), there is no such freedom with FPGAs. For example, consider the circuit shown in Figure 2, which shows an inherent potential mismatch between the delays of the two paths of combinational logic, named *path 1* and *path 2*, which converge at the XOR gate. The delay mismatch is inherent because of the structure of this circuit: *path 1* has

FPGA'16, February 21-23, 2016, Monterey, CA, USA

DOI: http://dx.doi.org/10.1145/2847263.2847272

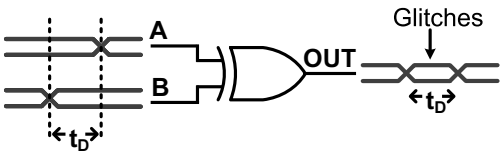


Figure 1: Example showing conditions which result in glitches at the outputs of combinational gates.

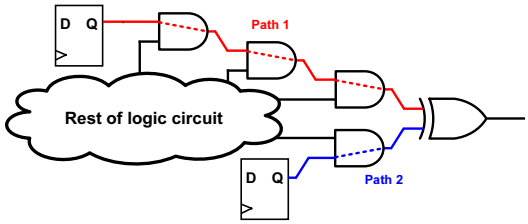


Figure 2: Example showing inherent mismatch in arrival times due to structure of circuit.

more logic gates than *path 2*. If this circuit were implemented in an ASIC, given sufficient freedom in the ability to trade-off power with area or speed, the delay of the single logic gate in *path 2* could be increased (for example, by reducing the sizing of its transistors), or the delays of the gates along *path 1* could be decreased (by increasing the sizing of their transistors), with the objective to equalize and align the arrival times at the inputs to the XOR gate. In contrast, if this circuit were to be implemented in an FPGA, each of the logic gates and inteconnects would be mapped to prefabricated circuits, whose delays cannot be optimized as freely, thus making the problem of equalizing arrival times difficult. Recent work has shown that glitches account for, on average, 26% of total core dynamic power for the MCNC circuits, and up to 50% for specific circuits [9].

2.2 Previous Work on Glitch Power Reduction

Several prior works have addressed glitch power in FPGAs [9, 10, 11, 12]. One recent approach [9] proposed to make use of the prevalent don't-care states in logic circuits to minimize glitches. The authors proposed to set the *don't-care* states to specific values such that when the input vector to a logic circuit momentarily assumes an intermediate (don't-care) state (while it transitions between two states), the output does not make a spurious transition. This technique is a light-weight approach to glitch reduction, as it has zero performance or area overheads, and offers reasonable glitch power reduction: an average of 13.7% over a set of benchmark circuits.

More direct approaches to glitch reduction were proposed in [12] and [13]. While these two works offered different approaches to glitch reduction, the overall strategy was similar to that depicted in Figure 3. The figure shows two signals, A and B, that are inputs to an XOR gate, and again, B arrives earlier than A. In theory, glitches can be eliminated if we equalize the delay along the input paths to ensure all signal transitions arrive at the same time at the different inputs to the logic circuit. This is shown in the figure, where a delay, Δt, is added to B, and so now the difference in the arrival times of the two signals to the XOR gate is $t_D - \Delta t$, which is also the width of the resulting glitch at OUT. Clearly, if the added delay can be calibrated such that it is equal to t_D, then the two signal transitions arrive at exactly the same time to the inputs of the XOR gate, and the glitch is eliminated.

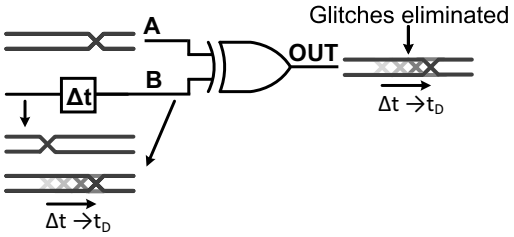


Figure 3: Glitch reduction through input delay balancing.

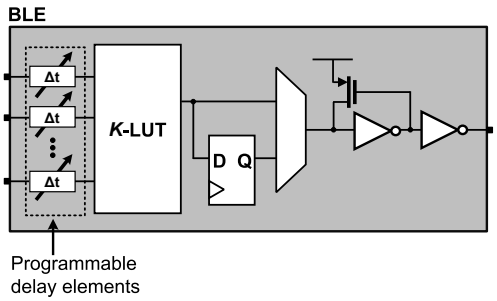
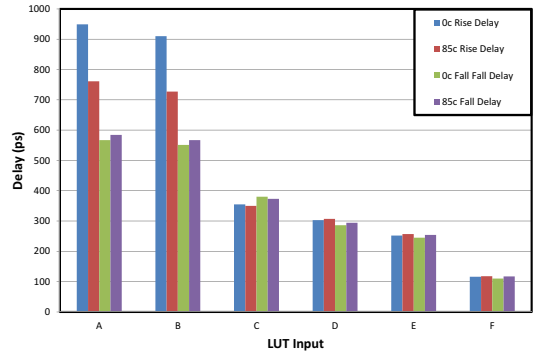


Figure 4: Proposed glitch-free BLE from [12].

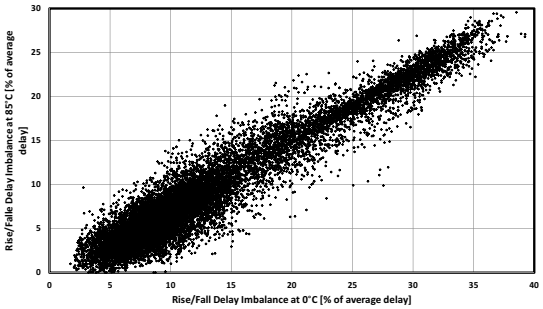
In [13], path delay equalization was proposed by inserting additional routing conductors along paths with early arrival times – the additional delay of each routing conductor slows the path down. While this approach requires no additional changes to the FPGA architecture or circuitry, and does not result in an area penalty (since routing conductors are often underutilized to begin with), the power reductions arising from the elimination of glitches are offset by the increased dynamic power due to the use of additional routing resources. In contrast, [12] proposes a modified logic element (LE) with programmable delay lines at the input pins, shown Figure 4. The programmable delay lines are used to adjust the arrival delays at each input pin so that they may be equalized and glitch power eliminated. While this approach does incur an area overhead associated with programmable delay lines on each input of a LUT (which may not be insignificant for large LUT sizes), the authors claim the ability to completely eliminate glitch power with this approach.

2.3 The Limitations of Path-Delay Balancing

One fundamental problem with the above "path delay equalization" approaches is that in general, circuit delays are a function of temperature and in some process corners, are also a function of logic state, because of unequal rise and fall delays. The reason for these effects is that modern commercial FPGAs are comprised of both CMOS gates and NMOS pass transistors. Multiplexers, which form the core of both programmable logic elements and routing switches, are effectively large trees of NMOS pass-transistors, along with CMOS level-shifters and buffers. These two different styles of circuitry respond differently to changes in operating parameters, such as temperature. Recall that the rise/fall delay of a CMOS circuit is approximately inversely proportional to both μ, the mobility parameter of the gate's transistors, and $(V_{DD} - V_T)$ (this is also true for the *fall* delay of an NMOS pass-gate). While $(V_{DD} - V_T)$ is an increasing function of temperature (by virtue of V_T decreasing with temperature), μ generally decreases with temperature, and so generally, the combined effect results in an overall increased delay with increasing temperature [14]. On the other



(a) Temperature dependence and rise/fall delay imbalance of LUT delays for each LUT input.



(b) Correlation of routing path rise/fall delay imbalance at 0°C and 85°C.

Figure 5: Temperature consequences on FPGA logic and routing delays.

hand, the *rise* delay of an NMOS pass-gate, assuming an ideal transistor model, is approximately:

$$t_{delay} = \frac{C_L V_{DD}}{\mu_N C_{OX}(W/L)(V_{DD}/2 - V_T)(V_{DD} - V_T)} \quad (1)$$

which exhibits an inverse-quadratic relationship between delay and $(V_{DD} - V_T)$. This equation is derived from a differential equation for the voltage at the source terminal of an NMOS transistor when its drain and gate terminals are held at V_{DD}. The increased sensitivity to $(V_{DD} - V_T)$ in this case typically results in an inverse temperature-dependence characteristic – i.e. delay *decreases* with temperature. This characteristic of NMOS pass-gates leads to two observable characteristics: (1) paths which are dominated by pass-transistors have inverse temperature dependence on delay, while paths which are dominated by CMOS gates will more likely experience delay degradation with increased temperature and (2) the inherent asymmetry in rise and fall behaviour of pass-transistors means that the rise and fall delays can only be balanced at a single PVT corner. Under other conditions, the rise and fall delays of pass-transistors are unbalanced. LUT and routing delay data from a design placed and routed in a Stratix III FPGA illustrate these trends, as shown in Figure 5.

Figure 5(a) shows the rise and fall delays for each of the six inputs to a Stratix III LUT at 0°C and 85°C. Note that inputs A, B and C, which connect to pass-transistors at deep levels of the LUT's pass-transistor tree, exhibit an inverse temperature dependency (likely owing to the fact that these paths are pass-transistor dominated), while inputs D, E, and F, which connect to pass-transistors closer to the output of the LUT exhibit slight delay degradation with increased temperature. Note also the large rise and fall delay imbalance (∼45% of the rise delay) for inputs A and B. Figure 5(b) highlights the unpredictable and temperature-dependent rise/fall delay imbalance of routing circuitry. The plot shows the relationship between the rise/fall delay imbalance, expressed as a percentage of the average delay, of each routing path in a sample circuit at 0°C and 85°C. In the figure, it is apparent that most routing paths exhibit a delay imbalance greater than 5%, with some approaching 40% at 0°C. Qualitatively, the plot also shows the unpredictability of the delay-imbalance over temperature, since a poor correlation between delay-imbalances at the two temperature points is apparent.

These characteristics of circuit structures in FPGAs means that the relative arrival times of signals at the inputs to a gate is a strong function of both temperature and logic-state (due to rise/fall delay imbalance). This implies that if delay equalization is used to remove glitches at a particular temperature and for a particular state, the glitches may reappear or new glitches may form at a different temperature/logic state. Circuitry to compensate for these variations would be prohibitive from an area point of view, since aside from the required additional circuitry needed to sense different temperature/logic states, these variations are a function of the specific mapping, placement, and routing of a design, thus making the variations highly unpredictable.

As opposed to prior path-delay-balancing techniques, we propose a glitch reduction technique with low area overhead which has the ability to eliminate all glitches whose pulse-widths are bounded across different process, voltage, and temperature conditions. The following sections detail the proposed circuitry and associated CAD support.

3. PROPOSED GLITCH FILTERING CIRCUITRY

3.1 Circuit Overview

At the core of our proposed glitch power reduction technique is the circuit shown in Figure 6, which we call a *glitch filter*, as it has the ability to suppress glitches. This circuit bears some resemblance to the circuit shown in [15], although there are several subtle differences.

The proposed circuitry is effectively a buffer with a first stage inverter, shown as $B1$ in the figure, followed by a second stage inverter, formed by transistors $M5$ and $M6$. Transistors $M1$ through $M4$ form gating circuitry which can disconnect the input stage from the output stage, and this enables the glitch filtering functionality of the proposed circuit, as will be described. The circuit also comprises a programmable *inverting* delay line, shown as $D1$ in the figure, whose delay determines the glitch pulse-widths which are filtered out by the circuit. When the glitch-filtering mechanism of this circuit is not needed (i.e. for timing critical paths), the multiplexer shown in the figure allows the input transition to bypass the delay-line and directly drive transistors $M2$ and $M3$. The SRAM configuration cells shown connected to the delay line, will be discussed below.

To understand the operation of this circuit, we first begin with a description of the dynamic behaviour of the circuit in response to a single transition at the input, and then describe the response to a glitch (i.e. multiple consecutive transitions). Immediately following a transition at the input to this circuit – say from logic-"0" to logic-"1" – the output of $B1$ transitions from logic-"1" to logic-"0", but because of the delay t_D of $D1$, the output of the delay line remains at logic-"0". The combination of signal values (the input to the circuit at logic-"1" while the outputs of $D1$ and $B1$ at logic-"0") means that $M1$ will turn on, thus discharging the gate of $M5$ and turning it off. Since $M3$ is also off during this time, the gate of $M6$ will be briefly floating, but because it was previously driven to V_{DD} (since the input was previously at logic-"0"), $M6$ will remain turned off. As such, the output remains at logic-"0" (despite both $M5$ and $M6$ being momentarily off). After a delay t_D, the transition

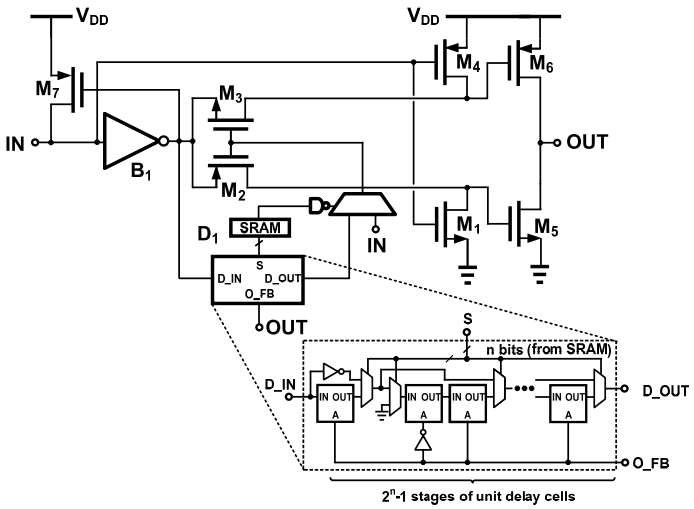
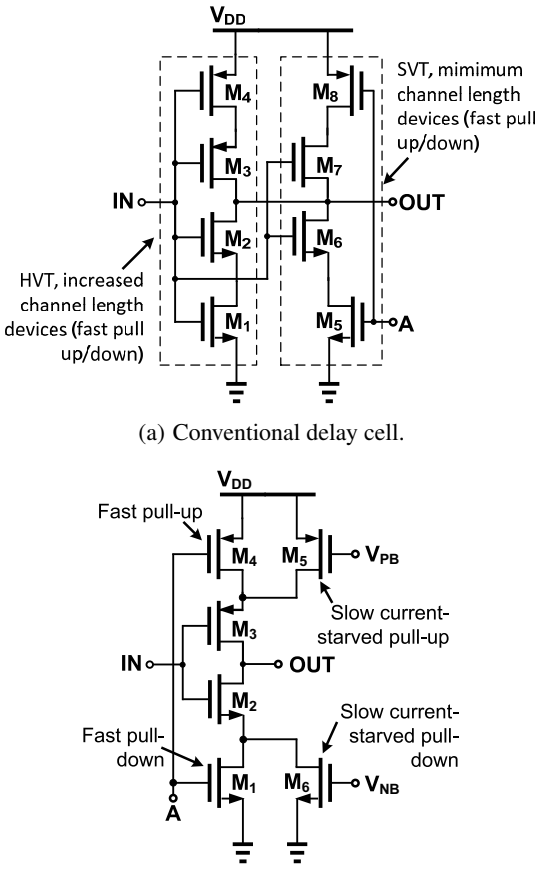


Figure 6: Proposed glitch-filter circuit.

at the input of the circuit is seen at the output of $D1$, and this turns $M3$ on. Since the source of $M3$ (which is driven by $B1$) is at this moment logic-"0", the gate of $M6$ is discharged, which turns $M6$ on and results in the output transitioning from logic-"0" to logic-"1".

This analysis of the response of this circuit to a transition at the input highlights an important property: immediately following a transition at the input, the output of the buffer is prevented from following suit and propagating the transition at the input. Instead, the buffer is forced to wait a delay of t_D before the output of $B1$ is connected to the gates of the transistors forming the output stage of the circuit. If however, during this delay t_D, the input transitions back to the previous value (i.e. a glitch has occurred), then the data value during the course of the spurious transition is not "seen" by the output stage when it is reconnected to $B1$. As such, the spurious transition does not propagate to the *output*, and the glitch input to the circuit is prevented from propagating and dissipating power in other areas of the chip. While this discussion highlights how lone pulses may be filtered out, consider what happens when a train of pulses, $x(t)$ appears at the input to the glitch filter: assuming the glitch filter contains an *ideal* delay line, it follows that the delay-line output is equal to $x(t-t_D)$, where t_D is the delay of the delay line. If for example $x(t)$ is a periodic function with a period equal to t_D, then by definition, the output of the delay line will in fact just be $x(t)$, and as such, all glitches will be allowed to pass through from the input to the output. In a more general sense, if the temporal separation, t_S, between glitch i and $i+1$ is less than t_D, glitch $i+1$ will only be partially filtered. If $t_S+t_W=t_D$, where t_W is the pulse-width of a particular glitch, then the glitch will pass from the input to the output of the circuit without any attenuation or pulse-width reduction. Thus, in an analogy to passive electronic filters, while the proposed glitch filter has a low-pass characteristic, in that all glitches with pulse-widths less than t_D may be filtered, it also has a *resonance characteristic*, where the glitches can pass through the filter without any attenuation for certain values of t_S. It can be shown that a mechanism to effectively flush the delay line of its contents following the arrival of a *restoring transition* is required to rectify this problem, and this can be achieved by ensuring that the delay of the delay-line is asymmetric and dependent on the state of its output: if the output of the glitch filter is at logic-"1", then the delay line is to have a slow output-fall delay and fast output-rise delay, while if the output of the glitch filter is at logic-"0", the delay line is to have a slow output-rise delay and fast output-fall delay. This allows a restoring transition to quickly "flush" the delay line, to ensure previous transitions at the input do not continue to stay in flight in the delay line – this helps to significantly mitigate



(a) Conventional delay cell.

(b) Current-starved delay cell.

Figure 7: Delay-cell implementation options.

the resonant effects. It can be shown that a glitch of pulse-width t_W seconds will be filtered without any resonant effects if $t_W < t_D - t_{Df}$ (where t_{Df} is the time required for a restoring transition to propagate through the delay line) with this new topology.

3.2 Delay Line Design

A number of options exist for the implementation of the programmable delay line $D1$. As will be discussed in Section 5, from our experiments we determined that each stage of our delay line had to provide, in the worst case, up to 600 *p*s of delay (some benchmarks required significantly less delay per stage as will be discussed in Section 5). Given that the FO4 delay in 65*n*m CMOS (for standard-V_T) is just over 20 *p*s, achieving such a large delay per stage in an area and power-efficient manner can be challenging. Since the delay of a CMOS gate is approximately a linear function of the product of the gate's drive resistance and its output load capacitance, increasing either of these will result in an increase in delay. However, increasing delay solely through increasing a gate's output capacitance would result in an unacceptably large power overhead, as such we considered two different approaches to effectively increase the drive resistance of the delay cells comprising the delay line, so that our delay targets could be achieved with minimal overhead.

The two alternative delay-cells are depicted in Figure 7. The first delay-cell shown in Figure 7(a), which we call a *conventional delay cell* is effectively a conventional inverter comprising of transistors with increased channel-lengths; a combination of a stack of series transistors (needed when the maximum modeled length by the foundry is less than the length necessary to meet a target delay)

and transistors with increased length leads to degraded drive resistance, allowing us to meet target delay. Note that while this technique also increases the input capacitance of the cell (which is the dominant load on the preceding cell in the delay-line), and thus will increase power, the power-overhead is still smaller than a delay-cell which achieves the same delay strictly through increased load capacitance. The second delay-cell shown in Figure 7(b), which is a *current-starved delay cell*, achieves increased delay by restricting the maximum pull-up/down current, by applying a bias voltage which is less (greater) than V_{DD} (GND) on the gates of transistor $M6$ ($M5$). This technique allows us to degrade drive-resistance to meet our delay targets by finding suitable voltages V_{PB} and V_{NB}.

These two techniques have different costs and trade-offs. For the conventional delay cell, an increase to its constituent transitors' channel lengths results in increased area, and as mentioned previously, increased input-capacitance and thus power. In contrast, for the current-starved delay cell, as long as suitable bias voltages V_{BP} and V_{BN} can be generated and distributed throughout the chip in a reliable and cost efficient-manner, the transistors comprising the delay cell can be set to minimum length and width, thus minimizing area and power overheads. However, given than $V_{PB} > GND$ and $V_{NB} < V_{DD}$, transistors $M5$ and $M6$ have degraded overdrive voltage, thereby making them more sensitive to V_T variation, although in this case sensitivity to variation and area overhead can be traded-off since increasing the size of $M5$ and $M6$, will reduce their σ_{VT}. In addition, there may be considerable area/power costs of distributing the voltages V_{PB} and V_{NB}. In this work, however, we assume that both the costs of the bias generation and the current-starved delay-cell's sensitivity to variation are insignificant, as this serves as a *lower-bound estimate* on the area and power overheads of the proposed glitch filtering circuitry. In contrast, the conventional delay cell allows us to form an *upper-bound estimate* on the are and power overheads of the glitch filtering circuit. Rigorous PVT analysis of the current-starved delay cell, as well as design and cost/benefit analysis of the bias generation circuitry is left for future work. Finally, observe that both of the delay cells shown in Figure 7 contain fast pull-up/down paths, which are activated by input A for both cells; these paths enable the cells to have asymmetric, state-dependent delay, which serves to eliminate resonant effects as discussed previously.

3.3 PVT Sensitivity

Given the nature of the way in which glitches are suppressed using this technique, we may draw some conclusions about the ability of this circuit to suppress glitches in the face of variation in PVT. Section 2.3 highlighted the principal shortcoming in preventing glitches using delay balancing: it requires precise knowledge and control of path delays. As discussed previously, varying PVT conditions makes this difficult to guarantee in modern processes, particularly in FPGAs because of the unique combination of various circuit structures which they employ. However, the exact *bounds* of path delays, and thus the bounds on the pulse-widths of resulting glitches, can be determined, and to some degree guaranteed.

The proposed glitch reduction technique in this work relies solely on the expected bounds of the glitches to be suppressed. If it is known in a circuit at a particular node which dissipates a large amount of power, that the vast majority of glitches at that node have pulse widths which are less than some bound W_{max} over all PVT corners, then simply by setting the delay of the delay line of the glitch filter at that node to a value $\geq W_{max}$ ensures that those glitches will be eliminated for all possible operating conditions of the circuit. Moreover, we can always ensure that the delay line delay is greater than W_{max} over all process corners; given that the delay line delay can be expressed as $t_D + \delta_D$, where t_D is the nominal delay of the delay line and δ_D is some random deviation from the nominal delay due to PVT variation, we can guarantee that the delay line delay is greater than W_{max} by setting $t_D \geq W_{max} + |\delta_D^{(max)}|$, where $\delta_D^{(max)}$ is the worst case variation of the delay line's delay over all PVT corners (which can be modelled and bounded). There may be a resulting timing penalty in this case, but for energy critical applications, this technique provides an ability to guarantee glitch suppression. While the inherent robustness to PVT makes this approach even more appealing, in this paper, we perform glitch suppression using statistics gathered from a single corner. Multi-corner glitch reduction and optimization using this technique is therefore left for future work. In addition to improved robustness compared to delay-balancing glitch reduction approaches, the proposed circuit also offers reduced area overhead, since only a single glitch filter is required at the output of the BLE, whereas a BLE with input delay balancing will need a delay-line and associated circuitry at each input to the BLE, as shown previously in Figure 4.

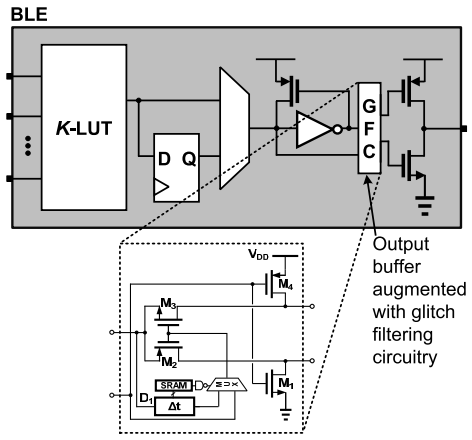


Figure 8: Proposed BLE architecture.

3.4 Proposed Architecture

Figure 8 shows a proposed Basic Logic Element (BLE) incorporating the proposed glitch filtering circuitry. We assume a conventional BLE will have an output buffer consisting of the inverter and transistors shown in the figure; this buffer exists to restore the voltage at the output of the bypass multiplexer to full CMOS levels, and is necessary to drive the BLE's output load. We propose to augment this buffer with glitch filtering circuitry consisting of transistors M_1-M_4, delay-line D_1, SRAM configuration cells, and additional auxiliary circuits as shown in the figure. The SRAM cells are used to configure the delay of the delay line. Recall that if the delay line is set to a delay t_D, then all glitches input to the glitch filter of a pulse width less than t_D will be filtered out. Thus, given information about the glitch statistics at the input to the glitch filter (which may be profiled through timing simulations of a given design mapped to the FPGA), the delay line can be configured accordingly to maximize glitch suppression and save power. However, the ability to filter glitches comes at a cost, namely an added delay of t_D. The configuration of the glitch filter delays is thus non-trivial. While it is desirable to filter out all possible glitches in the circuit, it may not be desirable from the performance angle. Moreover, the configuration of each glitch filter must take into account effects of an added delay on downstream nodes, since the additional delay introduced by a glitch filter may in fact result in the inception of new glitches downstream. If these downstream glitches propagate through very capacitive interconnect network, there may be a resulting net *increase* in glitch power. The additional delay introduced by a particular glitch filter setting may also leave less room for downstream nodes to filter out glitches due to reduced slack. As such, the optimization of glitch filter settings is a combinatorial optimization problem, wherein *global* glitch power must be reduced

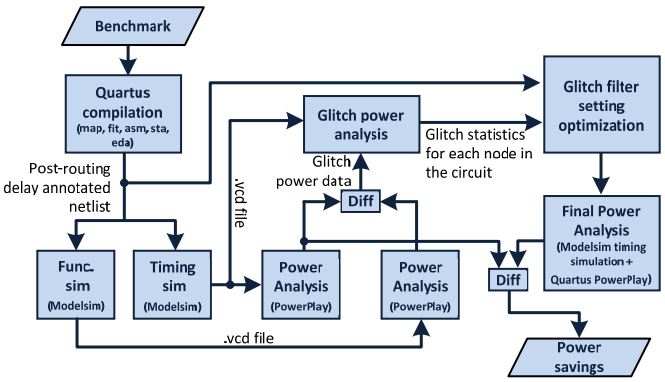


Figure 9: CAD/experimental flow for proposed glitch reduction technique.

under the presence of timing constraints. The CAD flow and glitch optimization approach are described in next section.

The circuits shown above were designed and simulated using commercial 65nm STMicroelectronics models to verify functionality, and extract power and delay overheads. All simulations were conducted using Cadence's Spectre simulator, with typical transistor models, 1V V_{DD}, and at a temperature of 28 °C. We assume a transition time of 150 ps at the input to the buffer, and an output load of 20fF (which is to represent the load of multiplexers and wire capacitance at the output of the BLE, similar to the loading considered in [16]). The delay penalty of the cell when the glitch-filtering mechanism is *not* used (and thus the input signal is allowed to propagate directly through the bypass multiplexer) is approximately 30 ps; this delay represents the signal propagation from the buffer's input through the glitch filter's bypass multiplexor to the gate of transistors M_2 and M_3 in Figure 8, and the signal propagation from the output of the first stage of the buffer through transistor M_3 (M_2) to the gate of transistor M_4 (M_1). These two signal propagations have some overlap, thus the total added delay is not simply their sum. For the baseline output buffer (without glitch filtering circuitry), under the aforementioned PVT conditions, we observed a $\sim$90 ps delay in our simulations, which grows to $\sim$120 ps with the addition of the glitch filtering circuitry.

For a glitch filter implemented with current-starved delay cells, the dynamic power overhead is $<<$ 1%, while for a conventional delay cell based delay-line, the glitch filter's dynamic power overhead would increase to $\sim$1.5% (compared to estimated power dissipated in routing, which is typically the dominating component of total dynamic power consumption). The static power overhead is also negligible, since aside from transistors $M1-M6$, and the transistors comprising the bypass multiplexer and input buffer $B1$, all transistors are HVT (high-threshold voltage transistors).

4. GLITCH FILTER SETTING OPTIMIZATION

4.1 CAD Flow

A proposed CAD flow supporting glitch filter-based power optimization is presented in Figure 9. While packing, placement, and routing remain the same as in a conventional FPGA CAD flow [17], we propose to add the following steps post-routing: glitch power analysis, glitch filter setting optimization, and final power analysis. Other than specific implementation details, overall the additional steps in the CAD flow are agnostic to the original CAD flow being augmented, and as such, we chose to use Altera's Quartus II CAD software as the base CAD flow to extend.

Details for each of the CAD flow steps introduced are as follows: the first step post-routing is glitch power analysis, where cumulative distribution functions relating glitch pulse-width and glitch power are generated. Total glitch power is calculated on a node-by-node basis by comparing power reports generated from a functional simulation and a timing simulation. Since a functional simulation contains no glitches, differences in power between two different simulations represent glitch power. The exact distribution of pulse-widths are extracted from the timing simulation, and the relative frequency of glitches of a certain pulse width is used to estimate the dissipated power arising from glitches of that pulse width. These statistics, in addition to the netlist and its timing information (i.e. the timing graph), are used by the glitch filter setting optimization step to find the best settings for each glitch filter used in the circuit. The details of this step will be discussed in the following section. Finally, a timing simulation is performed on the model of the circuit augmented with glitch filters (configured to the settings determined in the previous step), and the results of this simulation are used by power analysis to determine the power savings through glitch filtering.

4.2 Glitch Power Optimization

It is worthwhile to consider some of the challenges faced in optimization of glitch filter settings in a circuit. As mentioned above, reduction of all glitches at a node with pulse-width less than t_W results in an increase in delay at that node of t_W. This trade-off must be considered carefully. To begin with, in the optimization of overall power reduction, we ought to allocate more of the timing slack available on a particular path to nodes on that path with the greatest power reduction opportunity (that is, nodes with the highest glitch power). As such, timing data must be used in conjunction with the power reduction opportunities for nodes along a particular path. In addition, we must be careful in considering the consequences a particular glitch filter setting may have on downstream nodes. When a particular node's glitch filter is set to a setting of t_D, all downstream nodes will experience a *delay push-out* of t_D seconds at one (or more, if there are re-convergent paths) of their inputs. This results in the profile of relative arrival times (that is, the difference in time between the signal arrival times at each input) of the inputs to each downstream node being altered, which may result in *new* glitches (and increased power consumption) arising downstream. Even if the power dissipation of downstream nodes does not change, the glitch statistics of downstream nodes will become "stale", and therefore less useful for subsequent decision making.

In the absence of a model that relates changes in glitch-filter settings to altered glitch power characteristics on downstream nodes of the circuit, the glitch power analysis step (outlined above) would have to be executed frequently to ensure: (1) overall glitch power has not increased following changes to the glitch filter settings for a set of nodes and (2) the glitch power and pulse-width statistics for all affected nodes are updated so that they are always relevant and usable for glitch power optimization. This particular approach seemed to be intractable from a run-time point of view, and as such an alternative approach was pursued. Specifically, we opted to ensure that regardless the glitch filter settings applied to the various nodes in the circuit, the relative arrival times at all *consequential* nodes stayed unchanged. By consequential, we mean a node whose worst-case increased glitch power due to an altering of its relative fanin arrival times is significantly large. This means that nodes of relatively low output capacitance, whose worst-case glitch power is small in comparison to the potential glitch power savings elsewhere in the circuit, would be allowed to have altered input arrival times. This approach therefore ensures that the two main concerns previously discussed are avoided. By ensuring the relative arrival times stay unchanged (for consequential nodes), we are assured that no new (consequential) glitches are created at a given node whenever we attempt to filter glitches at other points (specifically, upstream nodes) of the circuit, while this also ensures that the glitch data at each consequential node is never stale. Whenever a consequential

node's glitch filter settings are changed to some delay t_D, we ensure that for all nodes downstream, the arrival times at each fanin path is similarly increased by t_D (by applying the necessary delay settings on upstream glitch filters). Admittedly, this is a somewhat conservative approach, and leads to compromised power reduction and development of a strategy to more optimally manage the effects of unequal delay push-out among the fanin paths of consequential nodes is left for future work.

4.2.1 Problem Statement

The optimization of glitch power as described in the preceding sections can in general be formulated as a constrained non-linear optimization problem. We are given as input a timing graph for a circuit, $G(V,E)$ where each vertex, $v \in V$, represents an input or output port of a block in the design (i.e. LUT, DFF, RAM, I/O), while each edge $e = (u,v)$, represents a signal path between ports (i.e. routing or a block's internal timing arc). For each node v in the circuit, we are given a *glitch power density function*, $GP_v(t)$ (this is obtained empirically during the glitch power analysis step in the CAD flow described in Section 4.1), which describes the amount of power dissipated at node v by glitches of width t. The total glitch power at node v is therefore:

$$P_v^{(max)} = \int_0^\infty GP_v(\tau)d\tau. \tag{2}$$

At each node v, we have a decision variable d_v, which is the specific glitch filter setting at node v. As described previously, the proposed glitch filtering circuitry is able to eliminate all glitches of pulse width less than d_v from appearing at its output when the glitch filter's delay line is set to a delay of d_v. This means that given a glitch filter setting of d_v, the total glitch power at node v would be reduced to:

$$P_v = \int_{d_v}^\infty GP_v(\tau)d\tau. \tag{3}$$

Therefore, total glitch power in the entire design (which we aim to minimize) is:

$$P_{total} = \sum_{v \in V} \int_{d_v}^\infty GP_v(\tau)d\tau. \tag{4}$$

At each node v, we wish to keep track of the worst-case arrival time, arr_v, and this expressed as:

$$arr_v = \max_{(u,v)\in E} arr_u + d_u + t_{uv}, \tag{5}$$

where t_{uv} is the delay from node u to v in the graph, and d_u is the delay push-out caused by the glitch filter setting at u (i.e. it is u's glitch filter delay setting). Let CO be the set of *circuit outputs* – i.e. primary outputs, flip flop inputs, etc. Given a constrained critical path of T, we have the following constraint:

$$\forall v \in CO : arr_v \leq T \tag{6}$$

As described in the previous section, we also wish to ensure that the delay push-out on each fanin edge of each node v are within some tolerance level of one another. However, this should be a soft constraint, as we only wish to avoid the case of unequal input delay push-out on nodes which are consequential. Similar to arrival time, we can keep track of the minimum and maximum input delay push-out on each node v, $dp_v^{(min)}$ and $dp_v^{(max)}$ respectively, with the following equations:

$$dp_v^{(min)} = \min_{(u,v)\in E} dp_u^{(min)} + d_u \tag{7}$$

$$dp_v^{(max)} = \max_{(u,v)\in E} dp_u^{(max)} + d_u \tag{8}$$

We wish to ensure that $dp_v^{(max)}$ and $dp_v^{(min)}$ are within some acceptable threshold of one another, i.e.:

$$dp_v^{(max)} - dp_v^{(min)} \leq K_{tol} \tag{9}$$

Where K_{tol} is the maximum allowed mismatch between input push-out (which can for example be obtained empirically and set to a constant value). Equation 9 represents a hard constraint, and so we may allow this constraint to be relaxed with the following modification:

$$dp_v^{(max)} - dp_v^{(min)} - K_{eq} \cdot e_v \leq K_{tol} \tag{10}$$

In contrast to Equation 9, we introduce a large constant K_{eq} (which can be set to some value which provably will always be greater than $dp_v^{(max)} - dp^{(min)}v$, such as the critical path delay), and a *slack variable* e_v, which is binary, and allows the constraint in Equation 9 to be violated for certain nodes. In order to objectively trade-off glitch power reduction opportunities available in the circuit with the requirement that nodes have equal delay push-out on input edges, we introduce a penalty term to Equation 4 to form our objective function:

$$P_{obj} = \sum_{v \in V} \int_{d_v}^\infty GP_v(\tau)d\tau + \sum_{v \in V} P_v \cdot K_p \cdot e_v, \tag{11}$$

where K_p is an empirically determined penalty factor. The second term in Equation 11 effectively indicates that whenever a node's input fanin delay push-outs are allowed to be unequal to one another, we must pay a penalty in power consumption pessimistically set to $K_p \cdot P_v$ (i.e. all glitch power reduction at node v is now lost, and some additional power penalty may be incurred if $K_p > 1$). This ensures that we are careful in balancing the input delay push-out on consequential nodes, while allowing us to violate this condition on inconsequential nodes if this would allow for greater glitch power reduction in other parts of the circuit. Equations 5 - 8 and Equations 10-11 together define a constrained optimization problem.

4.2.2 MILP Formulation

While the optimization of Equation 11 may at first appear to be intractable given that $GP_v(t)$ are arbitrary non-linear functions, we can simplify the equation by observing that each glitch filter's delay line has finite resolution and a limit on its maximum delay. In other words, $d_v = di_v \cdot res$, where res is the finite resolution of the delay line, and di_v is an integer decision variable (it is effectively the delay line setting for node v's glitch filter) in the range from $0 \leq di_v \leq 2^B - 1$, where B is the number of configuration bits of the delay line. This means that given the finite number of values for d_v, we also have a finite number of possible values for the glitch power dissipated at node v. This observation allows us to recast Equation 11 as a linear equation coupled with linear constraints. First, let $gp_v[k] = \int_{(k-1)\cdot res}^{k\cdot res} GP_v(\tau)d\tau$ where $1 \leq k \leq 2^B - 1$. We can then rewrite Equation 11 as:

$$P_{obj} = \sum_{v \in V} \sum_{k=1}^{2^B-1} x_v[k] \cdot gp_v[k] + \sum_{v \in V} \int_{t_{DM}}^\infty GP_v(\tau)d\tau + \sum_{v \in V} P_v \cdot K_p \cdot e_v \tag{12}$$

Where $x_v[k]$ are binary decision variables for node v, and indicate whether or not the glitch power corresponding to $gp_v[k]$ can be eliminated (i.e. if $x_v[k] = 0$, $gp_v[k]$ can be eliminated), and $t_{DM} = (2^B - 1) \cdot res$ corresponds to the maximum delay of the delay line. The second term in this Equation describes glitch power that cannot be reduced due to the finite maximum delay of the glitch filter delay line. As such, this term is a constant, and therefore Equation 12 is a *linear* function of $x_v[k]$ and e_v. We may relate this objective func-

tion with the underlying decision variables di_v with the following linear constraints at each node v:

$$\forall_{k\in\{1,\ldots,2^B-1\}} : res \cdot di_v + k \cdot res \cdot x_v[k] \geq k \cdot res \quad (13)$$

This constraint indicates that if $di_v < j$, $x_v[k] = 1$ for $k \geq j$; in other words, the delay line setting is unable to filter glitches of pulse-width greater than or equal to $res \cdot j$, and so the glitch power corresponding to these glitches cannot be eliminated. On the other hand, the constraint is satisfied if $x_v[k] = 0$ for $k \leq di_v$; thus the glitch power corresponding to glitches of width $\leq res \cdot di_v$ can be eliminated.

The *min* and *max* constraints in Equations 5, 7, and 8 also represent non-linearities in our problem formulation, however these too may be recast into a linear form through the following constraints:

$$arr_v \geq \forall_{(u,v)\in E}\ arr_u + d_u + t_{uv} \quad (14)$$

$$dp_v^{(min)} \leq \forall_{(u,v)\in E}\ dp_u^{(min)} + d_u \quad (15)$$

$$dp_v^{(max)} \geq \forall_{(u,v)\in E}\ dp_u^{(max)} + d_u \quad (16)$$

It can be shown that the objective function Equation 12 is minimized whenever arr_v and $dp_v^{(max)}$ are minimized, and whenever $dp_v^{(min)}$ is maximized. Thus, for example, any optimal solution to the objective function will guarantee that arr_v will be suitably minimal, while ensuring that the constraints in Equation 14 are met. As such, for all practical purposes, $arr_v = max_{(u,v)\in E}(arr_u + d_u + t_{uv})$. Similar claims can also be made for $dp_v^{(max)}$ and $dp_v^{(min)}$ to ensure that they are effectively the *max* and *min* of their respective arguments.

Together, the objective function in Equation 12 and the constraints in Equation 6, 10 and Equations 13-16 define a mixed-integer linear program (MILP), with binary variables $x_v[k]$ and e_v, integer variables d_v, and continuous variables arr_v, $dp_v^{(min)}$, and $dp_v^{(max)}$. This MILP can be solved using standard mathematical optimization software. We used the commercial `Gurobi Optimizer` tool, version 6.0.5 [18].

5. EXPERIMENTAL STUDY

To assess the power reductions attainable using our proposed technique, we conducted a set of experiments on the 20 largest MCNC benchmark circuits, as well as the 6 circuits from the UMass RCG HDL Benchmark Collection [19]. Since the glitch analysis step in our CAD flow requires functional and timing simulations, we chose the MCNC benchmark circuits as it is straightforward to generate testbenches for these designs that result in sufficient toggling on their internal nodes without requiring an intimate knowledge and understanding of how each of these benchmarks work. The UMass RCG HDL benchmarks were chosen because ready-made test benches are provided with the benchmark set. We also conducted an architecture study to investigate different trade-offs in area overhead and power reduction corresponding to the parameters of the glitch filter circuit. We considered the impact on power reduction from quantization effects in the reduction in resolution and finite maximum delay of the delay lines. Indeed, a glitch filter whose delay line is infinitely precise and has infinite range would offer the greatest flexibility, and thus the greatest opportunity to reduce power. On the other hand, a delay line with large range and fine granularity would also require many stages and SRAM cells, thus presenting an area overhead. Our experiments shed light on an appropriate choice for these parameters.

Our methodology is summarized in the CAD flow shown in Figure 9. A circuit is first compiled using Altera's Quartus II software, targetting 65nm Stratix-III devices, to generate a delay-annotated netlist. This delay annotated netlist is then input to ModelSim for timing simulation, and the appropriate input vectors are used for simulation (10000 random vectors are generated for the MCNC circuits, while the UMass RCG HDL benchmark circuits are provided with test benches containing appropriate input vectors). A functional simulation is also performed using the same set of vectors to allow us to characterize the glitch statistics of the circuit. These glitch statistics, along with timing information (also output by Quartus II in Standard Delay Format (SDF)) are then input to the glitch setting optimization framework described in Section 4.

To simulate the glitch statistics after applying our glitch filter settings, we created a behavioural model of our programmable glitch filter circuit – the exact glitch filter settings to be used for this circuit are supplied as parameters. We augment the outputs of combinational logic cells in the original Quartus II-generated netlist with instances of our glitch filter circuit, where each instance would have its glitch filtering parameters set by the previous stage of our CAD flow. This modified netlist is then run with the same set of random vectors used previously, and the output of this timing simulation is used to gauge power using Altera's PowerPlay power estimation tool (via a ModelSim-generated `.vcd` file for switching activity data).

5.1 Maximum Power Reduction Assuming Ideal Delay Lines

The first set of experiments we conducted were to assess the maximum possible power reductions assuming an ideal delay line (i.e. infinite precision and infinite maximum range). The experiments provide an upper bound on the achievable power reduction, prior to our optimization of the range and precision of the delay-line. Table 1 summarizes the power reduction and critical path degradation results for the case where a glitch-filter uses an *ideal* delay line (which has no limits on range or resolution, and does not have any area/power overhead), and for two specific delay-line implementations which will be discussed in the next section. The table lists glitch power reduction along with the resulting reduction to *logic and routing dynamic power* for each circuit in the benchmark set. It should be noted that the amount of dynamic power dissipated in logic (i.e. BLEs and FFs) and routing versus that dissipated in other parts of an FPGA varies from one design to another. For instance, in the UMass benchmark set, the *turboSram* benchmark's core dynamic power is dominated by the power dissipated in memory blocks, while logic and routing power contributes a small percentage to overall power dissipation. In contrast in other benchmarks, such as the *jpeg*, power dissipated in logic and routing power is dominant. The percentage of total dynamic power resulting from glitches is not shown in the table for the sake of brevity, but is similar to previously obtained statistics on the same benchmark set [9]. Note that the virtually no glitches were observed while simulating the *ava* benchmark from the UMass benchmark set, as such the table entries corresponding to glitch reduction for this benchmark are left empty.

Turning our attention now to the first section of the table, we see that with an ideal delay line, glitch reductions ranging from 45-97%, with an average reduction of 75% may be obtained. Logic and routing dynamic power savings range from -1% (for the *ava* benchmark, the 1% power *increase* corresponds to the glitch filter's power overheads, since no glitch power could be reduced) to 33%. Average logic and routing dynamic power reduction is $\sim$ 14.7% over the set of benchmark circuits.

For the other two delay-lines shown in the table, we pessimistically assume that these would be composed of the conventional delay cells shown in Figure 7(a). In our results, we include the simulated power overhead of a delay-line which comprises these delay-cells, in addition to the increased routing power resulting from the area-overheads of the glitch filtering circuitry. As will be

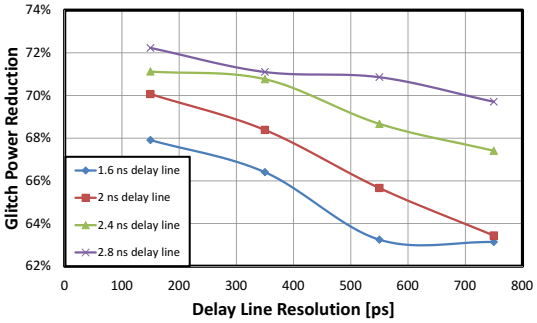


Figure 10: Glitch reductions for delay lines with varying resolution and maximum delay.

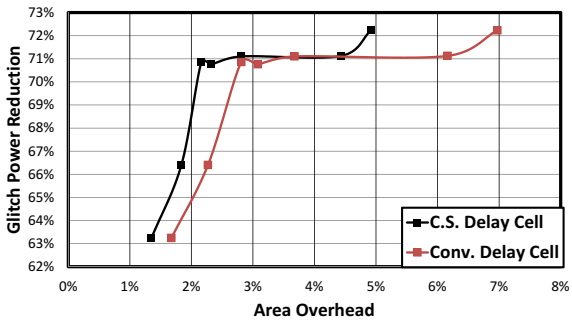


Figure 11: Max glitch reduction vs. area overhead.

described in the subsequent section, for a given delay-line resolution and maximum delay, we may estimate the resulting area overhead. Given an area overhead of x, we estimate that the tile dimensions will increase by a factor of $\sqrt{1+x}$, and thus routing power will (pessimistically) increase by this factor as well. Given the power-breakdown data which can be obtained from PowerPlay's power reports, we are able to estimate the resulting impact to total logic and routing power.

Table 1 also lists the critical path degradation for each circuit. Recall in Section 3 we discussed a delay penalty of 30 ps for the glitch-filter circuit even when the its glitch filtering mechanism is unused (e.g. for timing critical signals). This nominal delay penalty results in a slight impact to the critical path delay for the circuits considered in this work. Observe that the critical path degradation ranges from 0.2% to 2.8%, with an average critical path degradation of 0.9%. Note that the different delay-lines in the table have equal impacts to critical path, since the critical path is unaffected by the range or resolution of the delay-lines.

5.2 Power Reduction with Real Delay Lines

After establishing the maximum possible power savings for each design under ideal circumstances, we experimented with various resolutions and number of bits (i.e. maximum delay line delay) in a bid to identify the delay-line parameters to realize maximum power savings/minimum area overhead. The results of these experiments are shown in Figure 10.

The figure shows 16 different combinations of delay-line parameters, as resolution is varied from 150 ps to 750 ps, while the maximum delay of the delay line is varied from 1.6 ns to 2.8 ns. As expected, the results show monotonic decreases in power consumption as the resolution is increased; reduced resolution is associated with reduced flexibility to eliminate glitches. At the same time, we observe that for the same resolution, the ability to eliminate glitches increases monotonically as the maximum delay of the delay line (and thus, number of configuration bits) increases.

However, the plot reveals an interesting trend: while the delay-line in this study with the finest resolution (150ps) and longest delay (2.8 ns) allowed us to achieve a power reduction of just over 72%, approaching the maximum theoretical glitch reduction of 75%, the other delay lines with coarser resolution and reduced maximum delay were still able to achieve glitch reductions within 5-10% of the theoretical maximum. This compels us to further investigate the area-power trade-offs of these various delay-lines.

We begin by establishing an estimate for the area overhead of the proposed glitch-filter circuit. The number of minimum-width transistors introduced as overhead by this circuit (including configuration memory) for a delay-line constructed with current-starved delay cells is approximately:

$$A_{CS} = 8 + 12n + 6S \quad (17)$$

Where n is the number of bits, and S is the number of stages in the delay line. For a conventional delay cell based delay-line, the area overhead is approximately:

$$A_{Conv} = 8 + 12n + 10S \quad (18)$$

We further estimate that a suitably-sized 6-input LUT has an area of 1110 minimum-width transistors, by using transistor sizing data obtained through area-delay optimization of a conventional island-style FPGA architecture [16] (i.e. similar to Stratix III). Also, assuming that the area of an FPGA tile is broken down as follows: 50% routing, 30% LUTs and 20% for miscellaneous circuitry [20], we can form an estimate for the area overhead of the proposed glitch-filters for varying delay-line resolutions and maximum delay-line delays. For the 16 delay-line parameter combinations explored, we may then obtain the greatest power reduction for a given area overhead. The maximum power-reductions, for the two different delay-lines and over the range of area-overheads considered in this work are shown in Figure 11.

Interestingly, the plot shows that for an area overhead of less than 3%, a glitch reduction of $\sim$ 70% is obtained (which is within 5% of the theoretical maximum), even with the larger conventional delay cell based delay line. This corresponds to a glitch filter with a resolution of 350 ps and maximum delay of 2.4 ns (this requires 3 bits for configuration of the delay-line). The total core dynamic power dissipation in this case is reduced by 12.8%. Another candidate delay-line, with resolution of 550 ps and maximum delay of 1.6 ns (requiring 2 bits for configuration of the delay-line) offers slightly reduced glitch reduction of just over 63%, but this is at an area overhead of under 2%, again for both delay-line types. However, the actual reduction in core dynamic power is 12% in this case, thus making this combination of parameters particularly attractive given its low-cost and a net 2.7% decrease in dynamic power savings compared to the theoretical maximum! The detailed power reductions for these two candidate delay-lines are provided in Table 1.

6. CONCLUSIONS AND FUTURE WORK

This paper presented a glitch reduction circuit and the associated CAD flow/optimization framework needed to maximize glitch power reduction in an FPGA architecture comprising the proposed circuitry. In contrast to prior works, the proposed circuitry offers the ability to reduce glitch power over a wide range of PVT corners because of the nature in which it suppresses glitches, while incurring minimal area/delay overheads. Variations of the proposed circuitry allow glitch power to be reduced from 60-71%, which corresponds to a reduction in core dynamic power of 12-13%, at an area cost of 1.5-3%.

As indicated previously, the fanin delay push-out balancing scheme used in this work is a conservative approach to glitch power optimization, and indeed compromises to some degree our ability to remove glitches. Future work will investigate alternative ap-

Table 1: Detailed glitch and dynamic power reductions for glitch filters with three delay line variants.

Circuit	Critical Path [ns]	Ideal Delay Line			Delay Line Candidate #1: 2.4 ns total delay, 3 bits of resolution			Delay Line Candidate #2: 1.6 ns total delay, 2 bits of resolution		
		Glitch Reduction [%]	Logic and Routing Dynamic Power Savings [%]	Critical Path Degradation [%]	Glitch Reduction [%]	Logic and Routing Dynamic Power Savings [%]	Critical Path Degradation [%]	Glitch Reduction [%]	Logic and Routing Dynamic Power Savings [%]	Critical Path Degradation [%]
alu4	22	70	15	1.3	68	13	1.3	65	12	1.3
apex2	26	81	28	1.1	79	27	1.1	76	25	1.1
apex4	24	73	19	1.3	71	17	1.3	70	17	1.3
bigkey	18	63	18	0.5	56	14	0.5	50	13	0.5
clma	22	65	11	2.2	60	9	2.2	50	7	2.2
des	28	63	19	0.5	59	15	0.5	59	15	0.5
diffeq	18	93	5	0.2	92	4	0.2	92	4	0.2
dsip	17	85	18	0.4	74	15	0.4	53	10	0.4
elliptic	17	74	20	0.2	70	18	0.2	60	15	0.2
ex1010	28	82	33	1	79	30	1	78	29	1
ex5p	26	45	19	1.2	42	17	1.2	42	17	1.2
frisc	17	63	7	0.3	60	6	0.3	55	5	0.3
misex3	24	83	20	1	65	12	1	60	12	1
pdc	31	62	19	1	62	17	1	45	12	1
s298	21	59	5	2.8	57	3	2.8	50	3	2.8
s38417	17	96	25	0.7	96	23	0.7	95	23	0.7
s38584.1	24	84	4	1	80	3	1	58	2	1
seq	23	83	21	1	82	20	1	79	19	1
spla	28	71	24	1.1	70	21	1.1	66	20	1.1
tseng	18	75	7	0.2	55	5	0.2	53	4	0.2
ava	4.3	-	-1	1	-	-1	1	-	-1	1
fdct	7.2	97	17	0.7	94	16	0.7	94	16	0.7
fir_filter	12.6	76	6	0.2	72	5	0.2	72	5	0.2
jpeg	7.8	93	20	0.5	85	18	0.5	85	18	0.5
RS_decoder	7.7	73	3	1.2	71	3	1.2	66	3	1.2
turboSram	4.2	75	3	0.8	65	2	0.8	65	2	0.8
mean		**75**	**14.7**	**0.9**	**70**	**12.8**	**0.9**	**63**	**12**	**0.9**

proaches and/or computationally efficient methods to detect situations in which fanin delay balancing can be neglected, even for consequential nodes. Development of compact and accurate models which allow us to predict glitch statistics given the relative arrival times and switching statistics at input signals would serve this purpose well, and so this would be a promising avenue for future research. Finally, since the proposed circuitry allows for glitches to be suppressed under varying PVT, future work will explore the problem of multi-corner glitch power optimization, thus yielding a truly comprehensive glitch power reduction technique.

7. REFERENCES

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. On CAD*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[2] *Virtex-5 FPGA Data Sheet*, Xilinx, Inc., 2012.

[3] S. Gupta *et al.*, "CAD techniques for power optimization in Virtex-5 FPGAs," in *IEEE CICC*, 2007, pp. 85–88.

[4] *Introducing Innovations at 28 nm to Move Beyond Moore's Law*, Altera Corp., 2012.

[5] *Stratix III Programmable Power White Paper*, Altera Corp., 2007.

[6] J. Anderson and F. Najm, "Low-power programmable FPGA routing circuitry," *IEEE Trans. VLSI*, vol. 17, no. 8, pp. 1048 –1060, 2009.

[7] *The Breakthrough Advantage for FPGAs with Tri-Gate Technology*, Altera Corp., 2013.

[8] *Xilinx UltraScale: The Next-Generation Architecture for Your Next-Generation Architecture*, Xilinx Corp., 2014.

[9] W. Shum and J. Anderson, "FPGA glitch power analysis and reduction," in *ACM/IEEE ISLPED*, Aug 2011, pp. 27–32.

[10] S. Wilton *et al.*, "The impact of pipelining on energy per operation in field-programmable gate arrays," in *International Conference on Field Programmable Logic and Applications*, Antwerp, Belgium, 2004, pp. 719–728.

[11] T. S. Czajkowski and S. D. Brown, "Using negative edge triggered FFs to reduce glitching power in FPGA circuits," in *ACM/EDAC/IEEE DAC*, 2007, pp. 324–329.

[12] J. Lamoureux *et al.*, "Glitchless: Dynamic power minimization in FPGAs through edge alignment and glitch filtering," *IEEE Trans. VLSI*, vol. 16, no. 11, pp. 1521–1534, Nov 2008.

[13] Q. Dinh *et al.*, "A routing approach to reduce glitches in low power FPGAs," *IEEE TCAD*, vol. 29, no. 2, pp. 235–240, Feb 2010.

[14] W. Lee *et al.*, "Dynamic thermal management for FinFET-based circuits exploiting the temperature effect inversion phenomenon," in *ACM/IEEE ISLPED*, 2014, pp. 105–110.

[15] K. Chakravarthy, "Programmable glitch filter," Dec. 6 2001, US Patent App. 09/864,946. [Online]. Available: http://www.google.ca/patents/US20010048341

[16] C. Chiasson and V. Betz, "COFFE: Fully-automated transistor sizing for FPGAs," in *IEEE FPT*, Dec 2013, pp. 34–41.

[17] J. Rose *et al.*, "The VTR project: Architecture and CAD for FPGAs from verilog to routing," in *ACM/SIGDA FPGA*, 2012, pp. 77–86.

[18] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com

[19] "UMass RCG HDL benchmark collection," http://www.ecs.umass.edu/ece/tessier/rcg/benchmarks/.

[20] S. Chin and J. Anderson, "A case for hardened multiplexers in FPGAs," in *IEEE FPT*, Dec 2013, pp. 42–49.