CHARGE RECYCLING FOR POWER REDUCTION IN FPGA INTERCONNECT

Safeen Huda, Jason Anderson

Dept. of ECE, University of Toronto
Toronto, ON, Canada

Hirotaka Tamura

Fujitsu Laboratories Limited
Kawasaki, Japan

ABSTRACT

We propose charge recycling (CR) to reduce power consumption in FPGAs. We take advantage of the property that many routing conductors are left unused in any FPGA implementation of an application. Charge recycling via the unused conductors reduces the amount of charge drawn from the supply, lowering energy consumption. We present a routing switch that operates in two modes: normal and CR, and describe the CAD tool changes needed to support CR at the routing and post-routing stages of the flow. Results show that dynamic power in the FPGA interconnect can be reduced by ∼15-18.4% by the proposed techniques, depending on the performance constraints.

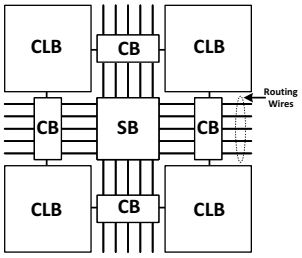
1. INTRODUCTION

Recent years have seen explosive demand for low power computing systems; from increased battery life in mobile systems to reduced electricity and cooling costs in large scale compute-farms, power reduction is a much pursued goal at all ends of the computing spectrum. While field-programmable gate arrays (FPGAs) are widely used in diverse applications, their high power consumption stands in the way of their widespread use in mobile and cloud settings. Indeed, a recent study showed that an FPGA design implementation consumes 7-14× more power than a functionally-equivalent ASIC design [1]. Power consumption in FPGAs is dominated by interconnect [2] – the area overhead required to realize programmability contributes to long metal wires with high capacitance, as do the programmable switches attached to each wire.

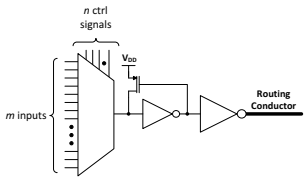
In this paper, we attack FPGA interconnect power through a charge recycling (CR) approach. Our approach leverages a unique aspect of FPGAs, namely, that a considerable portion of an FPGA's wire segments are unused in any given design, even in designs that use a high fraction of the available look-up-tables (LUTs). In modern commercial FPGAs, the unused routing conductors are wasted and pulled to a constant voltage. We propose instead that (some) conductors be programmably available as *charge reservoirs* (when they are unused), wherein they are each paired with a used conductor in close proximity. Given a used conductor with an available reservoir, charge from the used conductor is transferred to the reservoir on a falling transition; that is, instead of dumping all charge to the ground rail. Conversely, charge from the reservoir is transferred to the used conductor on a rising transition, thereby reducing the amount of charge that needs to be drawn from the supply rail to bring the used conductor to rail V_{DD}. We propose a dual-mode interconnect switch design capable of CR. In the first (high-speed) mode, the switch functions as a traditional switch; in the second (low-power) CR mode, special circuitry within the switch realizes the charge recycling behavior as described. Note that this power reduction technique is complimentary to other dynamic power reduction techniques, such as operating with multiple power supplies, or V_{DD} scaling. As such, combining charge recycling with other dynamic power reduction techniques will lead to further power reduction.

As with other architectural features, CAD tool support is required for CR to be applied successfully. We altered the routing algorithm within the publicly-available VPR framework [3] to encourage high-activity non-timing-critical signals to be routed using CR-capable switches that have available (free) reservoirs. We also implemented a post-routing *mode selection* pass that chooses the mode for CR-capable switches, subject to user-supplied speed performance constraints. Our mode selection is implemented as a mixed-integer linear program (MILP). The CAD tools are informed by models of the power, speed and area of CR-capable routing switches, derived from HSPICE simulation results for a $65n$m commercial process.

We studied the power reductions afforded by the proposed techniques, as well as impacts to speed and area. Results show that power in the FPGA interconnect can be reduced by up to 15-18.4%, depending on the architectural parameters and speed-performance constraints. While a number of circuit-level approaches to FPGA power reduction have been considered, including dual V_{DD} [4], power-gating sleep mode [7], low-swing signaling [8], subthreshold techniques [9], and body-bias leakage control [10], to the authors' knowledge, ours is the first work to consider charge recycling. The remainder of this paper is organized as follows: Section 2 reviews FPGA interconnect hardware and describes prior work on charge recycling in the custom ASIC context. The proposed charge-recycling interconnect switch is described in Section 3, including the circuit-level details and simulation results. The necessary CAD tool modifications are described in Section 4. The results are presented in Section 5. Conclusions appear in Section 6.



(a) Top level organization of CLBs, CBs, SBs, and routing wires.



(b) Circuit-level structure of routing switch.

Fig. 1: Conventional FPGA circuit structures.

2. BACKGROUND

2.1. FPGA Routing Hardware

Electrical connectivity between logic blocks (CLBs) in an FPGA is facilitated the *programmable routing network*. The routing network is comprised of fixed-length routing wires, Connection Boxes (CBs) which connect CLBs to routing wires, and Switch Boxes (SBs) which are used to stitch together routing wires to form paths. A simple organization of routing wires is shown in Fig. 1(a). The switch box consists of buffered routing switches, an example of which is shown in Fig. 1(b). The first stage of this circuit consists of a multiplexer with m inputs; these are the m tracks in the switch box that can be connected to the track being driven by this switch. The pull-up PMOS on the multiplexer output is present because the mux is typically implemented using NMOS transistors (which are poor at passing logic "1").

Due to the large capacitances of the routing wires in a modern FPGA, the power dissipated in the routing network is a dominant component of the total power dissipated, as much as 62% [7]. As such, any effort to reduce the power consumption in the routing network will potentially allow for an appreciable reduction in overall power.

2.2. Charge Recycling

Charge recycling is a technique that has been explored in the ASIC domain, particularly in the application of on-chip busses [13]. The idea is as follows: In conventional logic circuits, when a conductor makes a transition from logic "1" to "0", the stored energy on the wire – equal to $CV_{DD}^2/2$ where C is the wire capacitance – is completely dissipated. In contrast, charge recycling stores some of that (normally) dissipated charge in secondary nodes (capacitors). This charge can then be delivered to a wire which is making a transition from logic "0" to "1", reusing some of the energy that would have been otherwise lost in a conventional circuit.

The charge recovery and recycling process is illustrated in Figs. 2 and 3, respectively. The figures show two capacitors, C_L, the load capacitor, and C_R, a reservoir capacitor used to store recovered charge. Switches connect C_L to either V_{DD} or GND, and a third switch can be used to connect the two capacitors together. In Fig. 2, we see that C_L is

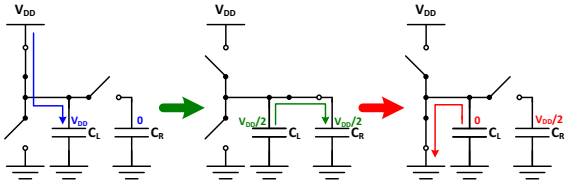


Fig. 2: Charge recovery during a "1" to "0" signal transition.

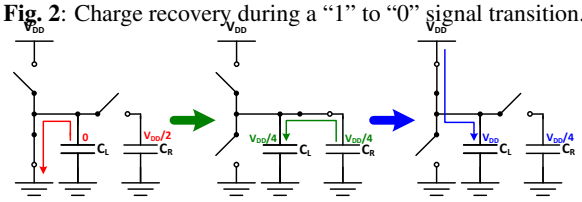


Fig. 3: Charge recycling during a "0" to "1" signal transition.

initially connected to V_{DD}, and there is no charge stored in C_R. During a logic "1" to "0" transition at C_L, initially the circuit undergoes a *charge recovery phase*, where C_R and C_L are connected (both V_{DD} and GND are disconnected from C_R). Given that the two capacitors have equal capacitance, through charge sharing, half of the charge initially stored in C_L will be transferred over to C_R, and thus the voltages of these capacitors will settle to $V_{DD}/2$. After the charge recovery phase, the two capacitors are disconnected, and C_L is connected to GND, and thus fully discharged.

In Fig. 3, C_L is initially connected to GND, and the voltage at C_R is equal to $V_{DD}/2$ as a consequence of the charge recovery phase of a preceding logic "1" to "0" transition. In a "0" to "1" transition, the circuit initially undergoes a *charge recycling phase* where C_R and C_L are connected to one another while both the V_{DD} and GND rails are disconnected. Through charge sharing, the voltage at C_L will rise (while the voltage at C_R will fall) to $V_{DD}/4$. After the voltage at C_L has settled, C_L and C_R are disconnected from one another, and the C_L is connected to V_{DD} to complete the full transition to logic "1". Note that in this example, the amount of charge actually drawn from the V_{DD} rail equal to $0.75 \cdot C_L \cdot V_{DD}$ as opposed to $C_L \cdot V_{DD}$. That is, there is an immediate 25% reduction in energy.

3. CR-CAPABLE FPGA INTERCONNECT

To reduce power consumption in the routing network, we propose an FPGA architecture with a routing network containing charge reservoirs (i.e. spare capacitors) which can be used to store the charge from a signal wire which is being discharged, and then reuse the stored charge when the signal wire is later being charged to V_{DD}. In our case, the charge reservoirs are actually unused routing resources within the routing network. That is, we are not proposing to create "new" capacitors to be used as reservoirs; rather, we propose to use unused routing resources as reservoirs.

Figure 4 shows waveforms for the falling and rising transitions of a signal wire whose charge is being recovered and recycled. During a rising transition, we see two dis-

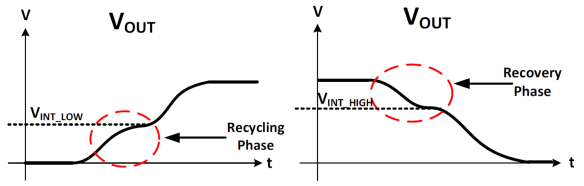


Fig. 4: Charge recovery/recycling on transitions: output waveform during a rising transition (shown on left); output waveform during a falling transition (shown on right).

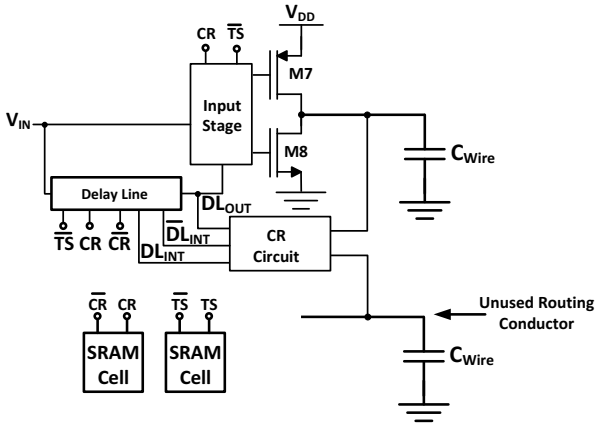


Fig. 5: CR buffer circuit.

tinct phases: a first phase where the routing conductor is initially charged to an intermediate voltage value V_{INT_LOW} as charge is shared between the routing conductor and the charge reservoir, and a second phase where the routing conductor is fully charged to V_{DD}. Similarly, during a falling transition, the output undergoes two distinct phases while discharging. In the first phase, the routing conductor is *partially* discharged to an intermediate voltage, V_{INT_HIGH}, through charge sharing with a charge reservoir. In the second phase of discharging, the routing conductor is fully discharged to GND.

In the steady state after several transitions have occured, V_{INT_LOW} and V_{INT_HIGH} will assume constant respective values. Assuming that the reservoir and load capacitors have equal capacitance, we note that V_{INT_HIGH} must be equal to $\frac{V_{DD}+V_{INT\_LOW}}{2}$, and V_{INT_LOW} must be equal to $\frac{V_{INT\_HIGH}}{2}$. From these two relations, we can therefore show that $V_{INT\_LOW} = \frac{1}{3}V_{DD}$, which therefore implies a theoretical maximum energy reduction of 33% using the proposed charge recycling technique.

The challenge in designing a charge-recycling FPGA interconnect switch is in realizing the dynamic behavior that must happen on rising and falling transitions. The switch must first disable the driving buffer, then it must allow charge sharing between the load and reservoir, and then, after charge sharing is complete, the switch must re-enable the driving buffer to complete the transition, bringing the load to one of the two rails. Note that the reason we must first disable the driving buffer is that otherwise, we may draw charge from the supply rail to charge *both* the load and the reservoir!

To implement the desired behavior, we designed the buffer circuit shown in Fig. 5, which we call a *CR Buffer*. The CR Buffer consists of an inverting input stage which applies the appropriate signals to the gates of $M7$ and $M8$ (which comprise the buffer's output stage), based on the operating mode of the buffer. The buffer circuit is further augmented with a circuit that performs the charge sharing between the load and reservoir, which we call the *CR circuit*. As shown in Fig. 5, two additional SRAM configuration cells produce signals CR (charge recycling) and TS (tristate) which control whether the switch is in normal mode (high-speed), CR mode, or is tristated (when its load conductor is being used as a reservoir by another switch).

A high level description of the operation of the charge recycling buffer is as follows: whenever there is a transition at V_{IN}, the calibrated delay of the delay line leads to a difference in signal levels between V_{IN} and DL_{OUT} (the output of the delay line). This difference results in the output stage of the buffer being momentarily tristated by the input stage, and activates the charge recycling circuit which will connect the output load to the reservoir conductor - thus allowing charge to be shared between the two conductors. After the transition at V_{IN} has propagated to the output of the delay line, and V_{IN} and DL_{OUT} are now equal, the charge recycling circuit will disconnect the output load from the reservoir, while the output stage will be re-enabled , thus allowing the output to completely rise (fall) to V_{DD} (GND).

The delay line (Fig. 6) consists of two stages (as shown in the figure): the first stage is a current-starved inverter which serves as the main delay cell in the delay line, while the second stage (inverter) serves to simply buffer the output of the first stage. The delay line operates in two modes: when CR and TS are both low (i.e. the CR buffer is not operating in CR mode), transistors $MD7$ and $MD8$ area off, $MD9$ and $MD10$ are on, and so DL_{INT} is pulled to V_{DD}. Thus, when a switch is not operating in CR mode, the delay line and CR circuit are both inactive. When CR is high and TS is low (CR buffer is operating in CR mode), DL_{INT} will be directly connected to V_{IN}. As such the delay line is active, and DL_{OUT} will simply be a delayed version of V_{IN}. Note that transistors $MD1$ and $MD4$ act as *clamped current sources*, as their gates are biased with voltages V_{PB} and V_{NB}, respectively. The bias voltages are set to limit the amount of current that may flow through these transistors, intentionally slowing down the inverter. The delay of the delay line can be calibrated by adjusting the values of bias voltages on the clamped current sources and in fact, this delay must be calibrated to match the settling time of the signal and reservoir nodes during charge sharing for maximum power savings.

The description of the input stage of the buffer circuit (Fig. 7), when operating in CR mode, is as follows: When V_{IN} makes a transition from logic "1" to "0", $M1$ will immediately turn on, raising the gate of $M7$ to V_{DD} and thus turning it off. Due to the delay line, node DL_{OUT} will initially continue to hold logic "1". This combination of logic values results in $M5$ turning on, but $M4$ remaining turned

off; this prevents the gate of $M8$ to be raised to V_{DD} immediately following a "1" to "0" transition at the input. Thus, the output stage is momentarily tristated as both $M7$ and $M8$ are off. Note that the gate of $M8$ remains low during this period as it effectively retains the node voltage prior to the transition at V_{IN}. As the transition at V_{IN} propagates through the delay line, DL_{OUT} will make the transition from "1" to "0", and this causes $M4$ to turn on, which results in $M8$ turning on, thus allowing the output to fall to logic "0" (GND rail). A rising transition on V_{IN} results in analogous behavior – temporary charge sharing with the reservoir before connecting the load to the V_{DD} rail.

Regarding the mode selection, note that transistors $M9$ is controlled by CR, while transistors $M10$ is controlled by $\overline{TS}$. As with the signal CR, TS is also the output of a configuration bit and is used to put the buffer into tristate mode; it is necessary for the buffer driving an unused conductor to be put into tristate mode so that the charge recycling process does not result in short circuits between buffers. When CR is high and TS is low, $M9$ and $M10$ are both off, and so the gating circuit works as described above. When TS is high and CR is low, $M10$ will be on; assuming that an unused buffer will have its V_{IN} held at V_{DD} and given that DL_{OUT} is at logical "0", this ensures that $M8$'s gate will be driven to GND while $M7$'s gate will be driven to V_{DD}, thus placing the buffer in tristate mode.

We now turn to the circuit that performs the charge sharing between the load and reservoir, which we called the *CR circuit*. The details of the CR circuit are shown in Fig. 8. The CR circuit is implemented using pass transistor logic and drives the gates of the transistors connecting the switch output and charge reservoir nodes. The description of the circuit is as follows (assuming CR is at logic "1", and so DL_{INT} = V_{IN}): When V_{IN} makes a transition from "0" to "1", since DL_{OUT} will remain at logical "0" for a brief period of time, $M13$ will be turned on, resulting in transistor $M18$ turning on, which allows charge to be recycled from the reservoir capacitor. After a brief period of time, DL_{OUT} will make the transition to logical "1", and this results in $M14$ turning on and thus $M18$ being shut off. This disconnects the reservoir node from the output. When V_{IN}

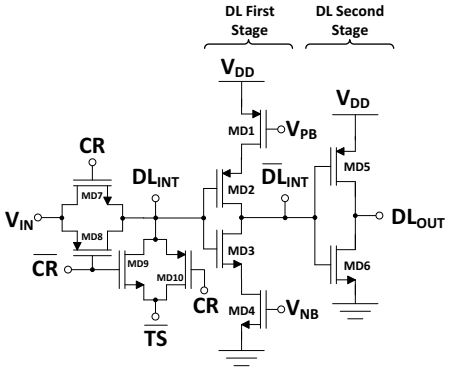


Fig. 6: Circuit implementation of delay line in CR buffer.

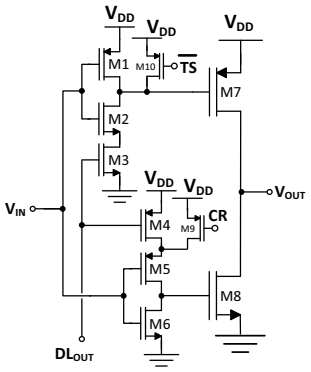


Fig. 7: CR Buffer with input stage circuitry exposed.

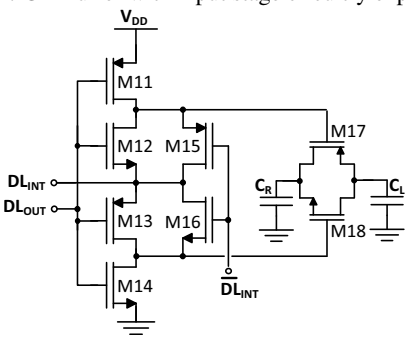


Fig. 8: Charge recycling (CR) circuit.

makes a transition from "1" to "0", since again DL_{OUT} will momentarily remain at logical "1", the gate of $M17$ will be pulled to GND through $M12$, thus turning it on and allowing charge to be recovered from the load by the reservoir capacitor. Again, after the signal V_{IN} has completely propagated through to DL_{OUT}, $M11$ will turn on, $M12$ will be shut off, which results in both $M17$ and $M18$ being shut off, again disconnecting the output from the reservoir capacitor.

We simulated the proposed switch design using commercial 65nm STMicroelectronics models. Fig. 9 shows the simulated waveforms as the output node of a switch and a reservoir node when the input to the switch is driven by a signal with a high toggle rate. Note that, as expected, during the charge recovery/recycling phases, the output and reservoir node voltages converge to the same voltage – this is expected since during these phases the nodes are effectively short-circuited to one another. We estimate that the total load capacitance of a length-4 wire in a commercial FPGA is near 200fF as follows: Based on the tile area data in [15], we estimated a single tile length in each dimension as 170μm for an FPGA in 65nm CMOS which implies a wire capacitance of 170fF for a length-4 wire, assuming a wire capacitance per length of 0.25fF/μm. We anticipate that the additional parasitic capacitance of a length-4 wire (resulting from the diffusion capacitance of the many multiplexors connected to a length-4 wire) will push the total load capacitance close to 200fF in 65nm CMOS. With a 200fF load the simulations show a power savings of 26%. The difference in simulated power savings and the theoretically-expected 33% reduction is because of the additional overhead power

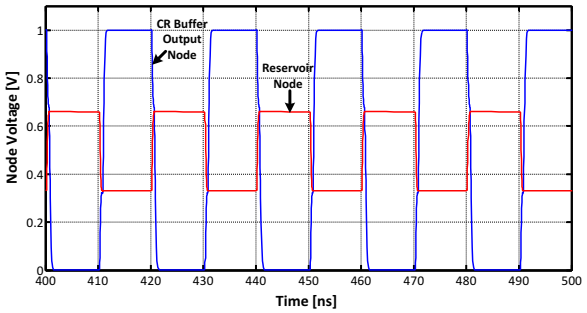


Fig. 9: Simulated waveforms of switch output and reservoir nodes.

consumption of the circuitry needed to facilitate charge recycling.

The simulated delay for our designed CR buffer, when used in CR-mode, is approximately $650ps$, which represents a two-fold delay hit over a conventional buffer, based on our simulations. While the delay penalty may appear high, prior work shows that over 70% of interconnect switches can be slowed down by 75% in a commercial $90n$m Xilinx FPGA, without *any* critical path delay increase [8]. When the CR buffer is not used in CR-mode however, the simulated delay penalty is approximately $25ps$, which is $\sim$8% of the delay of a conventional buffer. This delay penalty is resulting from the fact that in the input stage of the CR buffer, when operating in normal mode, the gate of $M8$ is charged through two series PMOS transistors ($M4$ and $M5$), while the gate of $M7$ is discharged through two series NMOS transistors ($M2$ and $M3$). Nonetheless, since the majority of the total delay through the buffer is given by the delay of the output stage, the delay penalty at the input stage posed by our buffer design results in a modest overall delay penalty.

4. TOOL SUPPORT

One of the premises of the proposed power saving technique is that switches which are in a CR mode will have adjacent tracks which are *unused* and thus can serve as reservoirs. Therefore, special effort must be made during the physical implementation of a circuit to create opportunities for charge recycling, especially for switches carrying signals with high switching activity (as CR will provide the most benefit for high-activity signals). In order to ensure that CR-mode switches have unused adjacent tracks, the first component of our CR-aware CAD flow is a modification to a conventional FPGA router. The router is modified to encourage high-activity signals to be routed using CR-capable switches, and moreover, to see that the reservoirs associated with such switches (in the underlying architecture) remain unused.

The VPR router incorporates the PathFinder algorithm [16]. PathFinder routes individual driver/load connections one at a time. A cost function is used to find a low-cost path through the routing fabric from a driver to a load. The cost function incorporates metrics of timing performance and *routability* (i.e. the ability to legally route all of the connections in the circuit) in order to yield a routed circuit with a favorable quality of results. The cost function implemented VPR is:

$$\begin{aligned} Cost_n &= (1 - Crit_i) \cdot cong\_cost_n \\ &\quad + Crit_i \cdot delay\_cost_n \end{aligned} \tag{1}$$

where n is the routing resource (e.g. wire segment) being considered for addition to a partially-completed route, i is the driver/load connection being routed, $Crit_i$ is the timing criticality of the connection (equal to 1 for connections on the critical path, and decreasing to 0 as the slack of the connection increases), $cong\_cost_n$ is the congestion cost of routing resource n which gives an indication of the *demand* for the routing resource among nets, while $delay\_cost_n$ is the delay of routing resource n.

The rationale for cost function (1) is that when connection i is being routed to a routing resource n, if the connection has high criticality (i.e. $Crit_i$ is close to 1), then the cost of using the routing resource is principally given by the delay of that resource (equal to $delay\_cost_n$). For such timing-critical connections, the principal concern should be to route the connections using the fastest routing resources, with less focus on the demand for such resources by other nets. On the other hand, if the criticality of the connection being routed is low (i.e. $Crit_i$ is close to 0), then the $cong\_cost_n$ is the dominant part of the cost of using a resource. As such, connections with sufficient timing slack are encouraged to use resources with less demand, potentially pursuing more circuitous routes to reduce demand on overcongested nodes.

Turning now to our modification of the VPR router, for a routing segment, n, driven by a CR-capable switch, we use the following cost function:

$$\begin{aligned} Cost_n &= (1 - Crit_i) \cdot [cong\_cost_n \\ &\quad + (1 - \alpha_i) \cdot res\_cost_n \\ &\quad + \alpha_i \cdot (PF \cdot res\_occ_n + GF \cdot not\_cr_n)] \\ &\quad + Crit_i \cdot delay\_cost_n \end{aligned} \tag{2}$$

where α_i is the activity factor for connection i being routed (normalized to a [0:1] range), PF and GF are scalar tuning parameters (determined empirically), and res_cost_n is equal to $(1 - Crit_j) \cdot \alpha_j$, where j is the index of the connection currently occupying the potential reservoir of node n. Note that res_occ_n and not_cr_n are binary variables; res_occ_n is equal to 1 if node n is a CR-capable switch whose reservoir is used and is equal to 0 otherwise, while not_cr is equal to 1 if node n is not CR-capable, and is equal to 0 otherwise.

The motivation for the modification to the cost function is as follows: while routing a circuit, we wish to route connections which have high switching activity and sufficient timing margin (i.e. connections which are favourable candidates to use CR-mode switches) such that they use CR-capable switches whose reservoir tracks are unoccupied. A switch with an occupied reservoir track cannot be put into CR mode, and therefore, such a scenario would lead to a lost opportunity for power savings. In general, when routing connection i, we need to consider two cases: 1) connection i is a favourable candidate to use CR-mode switches,

or 2) connection i is not likely to use CR-mode switches. For the former case, we aim to penalize routes where reservoir tracks are currently occupied, this is given by the $(1 - Crit_i) \cdot \alpha_i \cdot PF \cdot res\_cost_n$ term. For the latter case, we aim to avoid situations where connection i uses a track which could potentially be a reservoir for another connection j (which may be a favourable candidate to use CR-mode switches). As such, we penalize this case as well, and this is given by the $(1 - Crit_i)[(1 - \alpha_i) \cdot res\_cost_n)]$ term. Furthermore, the $(1 - Crit_i) \cdot \alpha_i \cdot GF \cdot not\_cr_n$ term acts to penalize nets which are favourable candidates to be put into CR mode, but are not using CR-capable switches.

4.1. Post-Routing CR Mode Selection

After routing is complete, we perform the final mode selection for each CR-capable routing switch. We formulate the problem of mode selection as a mixed-integer linear program (MILP). Our formulation accepts a user-provided critical path delay constraint as input and optimizes power subject to the constraint by placing an (activity-weighted) maximum number of routing conductors into CR mode. Let $G(V, E)$ be the circuit's timing graph where each vertex, $v \in V$, represents a routing conductor or pin and each (directed) edge, $e = (u, v)$, represents a programmable switch or a combinational path through a combinational logic element (e.g. a LUT). Let C be the subset of V corresponding to CR-capable routing conductors. For each vertex $v \in C$, we introduce a binary 0/1 decision variable, γ_v, indicating whether v is in CR mode. Then, the delay of the conductor is:

$$D_v = DIntrinsic_v + \gamma_v \cdot \delta_v \tag{3}$$

where $DIntrinsic_v$ is v's delay when CR is turned off, and δ_v is the *added* delay when CR is enabled. Vertices corresponding to input pins on combinational logic elements are assigned zero delay. We lump the delay through a combinational logic element onto the vertex corresponding to its output pin, however, it is straightforward to model the case of there being multiple different path delays through the gate. No γ variables are introduced for non-CR-capable vertices – such vertices only have intrinsic delays.

Having defined the delays for logic and routing, we introduce constraints specifying the worse-case arrival time at any vertex, v, Arr_v. Normally, this is done using a max function over vertices that fanin to v in the timing graph. However, as max constraints cannot be handled directly in MILP, we introduce "greater than" constraints for each of v's fanin vertices:

$$Arr_v \geq \forall_{(u,v) \in E} \, Arr_u + D_v \tag{4}$$

where the arrival times for vertices corresponding to primary inputs and flip-flop outputs are initialized to 0. Strict equality is used for vertices having a single fanin vertex in G (as no max function is needed in such cases).

Let CO be the subset of verticies in V corresponding to combinational path end-points, i.e. primary outputs of the circuit and flip-flop inputs. Given a user-provided constraint on the critical path delay, T, we introduce the following constraints:

$$\forall_{v \in CO} \, Arr_v \leq T \tag{5}$$

The objective function of the MILP seeks to maximize the switching activity-weighted number of routing conductors placed in CR mode, and, at the same time, contains terms that cause the arrival time constraints in (4) to realize a max function:

$$\phi = \sum_{i \in C} \alpha_i \cdot \gamma_i - \sum_{j \in V} Arr_j \tag{6}$$

where α_i is the switching activity of vertex i. The first summation counts the number of conductors in CR mode, where each is weighted by its activity. The second summation, whose sign is negative, ensures that the arrival time assigned for each vertex is as small as possible, yet consistent with the constraints of (4), thereby realizing the familiar max function seen in timing analysis arrival time propagation. Note that we scale all delay values to be small enough such that the second summation in (6) does not dominate (6). The objective function (6) and the constraints above constitute an MILP that can be solved with a standard MILP solver (we used the open-source `lpsolve` ver. 5.5 [17]).

5. EXPERIMENTAL STUDY

In order to assess the merits of the proposed circuitry and CAD flow, we used the set of benchmark circuits packaged with VPR 6.0 [3]. Our baseline non-CR-capable architecture contains unidirectional wire segments (direct-drive) which span 4 CLB tiles, uses the Wilton switch block [18], and has logic blocks with ten 6-LUTs/FFs per CLB. All benchmark circuits were routed on the baseline architecture to determine the minimum number of tracks per channel needed to route each circuit successfully (W_{min}). For each circuit, we then computed $W = 1.3 \times W_{min}$ to reflect a medium-stress routing scenario – standard approach in FPGA architecture research. The computed W value for each circuit was used for all experimental runs of the circuit. We use the ACE switching activity estimator tool [19] to compute switching activity for each signal in each benchmark circuit. We make the following assumptions about the architecture in our study: (1) each routing segment is paired with one other routing segment and either can serve as the reservoir for the other, and (2) the paired routing conductors have the same start/end points but run in opposite directions[1].

We assess the available power reductions as the performance constraint on the worst-case critical path is varied in the post-routing mode selection phase of our CAD flow (see Equation 5). Naturally, relaxing the path constraints will allow more switches to be placed in CR mode, yielding improved power reductions at the expense of speed. The power

[1] We considered alternative pairings, however, this scheme produced the best results. The power reduction results for alternative pairings are omitted for space reasons.

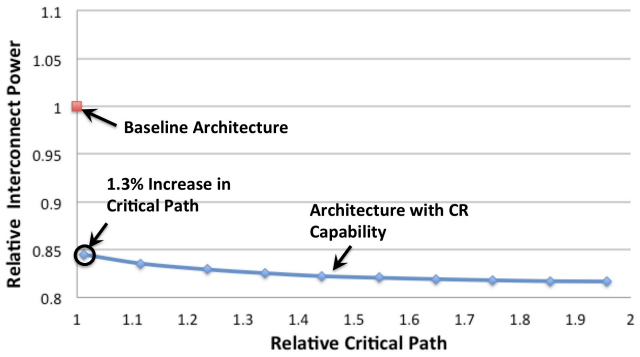


Fig. 10: Power reduction vs. critical path relaxation factor. Shown for an architecture fully populated with CR capable switches.

results correspond to dynamic power consumed in the FPGA interconnect. As discussed previously, we assume that CR mode provides a 26% reduction in switch power consumption and results in a 2× delay hit compared to a conventional buffer. When a CR buffer is not used in CR mode, we assume a 8% delay hit compared to a conventional buffer, as obtained from simulation results above.

Fig. 10 shows the variation in power savings vs. the relative critical path (compared to the baseline). The leftmost point of the blue curve shows the minimum power savings that can be achieved with this technique under minimum relaxation of performance constraints. Here, we see a power savings of approximately 15.5%, and a critical path penalty relative to the baseline of 1.3%. One important point to note is that while the CR buffer circuit reduced interconnect power by ∼26%, only a fraction of all of the routing conductors can be put into CR mode. Moreover, all of the routing conductors which cannot be put into CR mode incur a power *penalty* of around 1%, because of the gating circuitry in the input stage (the CR circuitry and delay line are both disabled when the switch is not in CR mode). This is why the total interconnect power reduction is diminished to 15.5%, while the intrinsic penalty to critical path is due to the delay penalty of CR buffers (not operating in CR mode). However, by relaxing timing constraints, more switches can be put into CR mode. Those switches used for connections with high criticality and thus could not be put into CR mode under strict timing constraints can now be put into CR mode, and thus additional power savings can be achieved. The results show that an additional 3% in power savings can be achieved as the critical path is allowed to increase by a factor of 2. Power savings in the FPGA interconnect reach over 18% in this case, which we believe will be useful in power-sensitive applications that do not need to run at high speed.

The power reductions for an architecture fully populated with CR capable switches and a critical path relaxation factor of 1, are detailed on a benchmark-by-benchmark basis in Table 1 (columns 2 and 3). Note that in the table, **TF** refers to the critical path relaxation factor; a **TF** of 1 implies no relaxation of the minimum critical path achievable in the proposed architecture, while a **TF** of 1.5 refers to a 50% relaxation of the minimum critical path.

In addition to assessing the merits of the proposed techniques from the perspective of power reduction, we also studied the impact on area incurred by including CR-capable routing switches. We first begin with a breakdown of the area, in the number of minimum-width transistors, of all components of a CR-capable switch. We use the number of min-width transistors as the area metric as this is the area metric used in the VPR place/route tool, and thus, it allows us to find the percentage increase in area (over the baseline architecture) of our CR-capable routing architecture.

The total area overhead of the proposed additional circuitry is equal to 45 minimum-width transistors per switch: 39 minimum-width transistors for the CR circuit, gating circuit, and delay line, and an additional 6 transistors for an additional SRAM configuration cell needed for CR-mode selection (note that Fig. 5 shows two SRAM cells, but these are shared between two paired CR-capable buffers). From our benchmark circuits, we obtained the number of CR-capable switches for each circuit as the percentage of CR-capable switches was varied from 10% to 100%, and assessed the total routing area overhead incurred by the additional circuitry. The variation in area overhead and power savings vs. the percentage of CR capable switches is shown in Fig. 11. For the case where the FPGA is fully populated with CR-capable switches (which leads to maximum power reduction), the proposed circuitry leads to an ∼25% increase in routing area with a power savings of approximately 15.5%. Note that as the percentage of CR capable switches is reduced, we see diminished power savings; this is because architectures with reduced number of CR capable switches offer less opportunities to the router to put CR-favourable nets into CR mode. Nonetheless, it is interesting to observe that the reductions in power savings are not directly proportional to the reduction in CR-capable switches. For instance, as the percentage of CR capable switches is reduced to 50%, the total power savings drops from 15.5% to approximately 10%; thus there is a 35% reduction in power savings, but a 50% reduction in area overhead. This encourages us to potentially investigate architectures which have a reduced number of CR capable switches but still over significant power savings. The area overhead for each circuit is shown in Table 1. Column 4 in the table shows the increase in routing area; column 5 in the table shows the increase in overall area.

Table 1 shows that the overall increase in area is 11.3%. With a square tile layout, this corresponds to an increase of ∼5.5% (i.e. $\sqrt{1.113}$) in each of the x and y tile dimensions. We believe this is a loose upper bound on the additional capacitance that would arise from having a larger tile, as the portion of total capacitance contributed by attached switches is unaffected by the tile size increase. The power reductions provided by the proposed technique (15-18.4%) thus exceed the potential increase in wire capacitance (at most 5.5%).

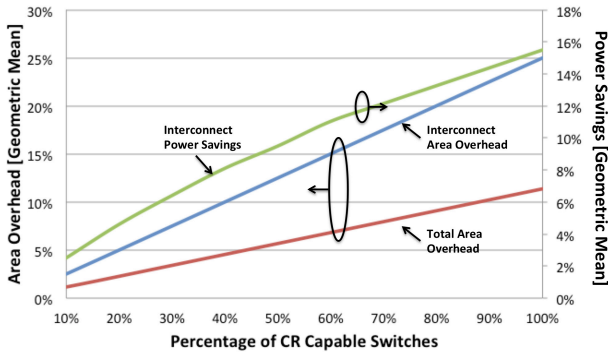


Fig. 11: Area overhead and Power Savings vs. percentage of CR-capable switches.

Circuit	Pwr Red. TF = 1	Pwr Red. TF = 1.5	Routing Area Increase	Total Area Increase
bgm	13.9%	19.9%	24.2%	13.9%
blob_merge	13.6%	14.1%	24.8%	12.5 %
boundtop	16.5%	17.5%	24.5%	10.7%
ch_intrinsics	17%	19.4%	27%	10.4%
diffeq1	17.4%	19.8%	26.8%	11.3%
diffeq2	13.3%	19%	25.7%	10.2%
LU8PEEng	15.9%	16%	24%	13.7%
mkDelayWorker32B	16.5%	20.7%	23.8%	11.9%
mkPktMerge	12.9%	20.3%	22.9%	8.3%
mkSMAdapter4b	16.5%	18.1%	24.7%	10.9%
or1200	18.7%	19%	24.1%	11.9%
raygentop	15.8%	17.2%	25.1%	11.5%
sha	15.9%	16.4%	24.8%	10.2%
stereovision0	16.6%	18.2%	23.1%	9.9%
stereovision1	14.2%	16.9%	24.2%	12.9%
stereovision2	14.9%	21.4%	24.1%	13.8%
stereovision3	15.1%	20.3%	32.4%	9.2%
geomean	**15.5%**	**18.4%**	**25%**	**11.3%**

Table 1: Power reductions and area overhead for each benchmark in an architecture fully populated with CR-capable switches.

6. CONCLUSIONS AND FUTURE WORK

In this work, we presented a novel technique which specifically targets the reduction of dynamic power in the routing network of FPGAs. The idea is to leverage the unused routing conductors in the FPGA interconnect as *charge reservoirs*, which receive charge from neighboring conductors on their falling transitions and which share charge on rising transitions. Ultimately, this reduces the amount of charge that needs to be drawn from the supply rail, saving power. We presented a comprehensive description of the proposed technique, from transistor-level description of necessary circuit structures, CAD flows and algorithms which would optimize the possible power reductions afforded by the proposed technique, and assessed the tradeoffs in area, power, and performance. Results show that charge recycling reduces dynamic power in the FPGA interconnect by ∼15-18.4%, depending on performance constraints.

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] L. Shang, A. Kaviani, and K. Bathala, "Dynamic power consumption of the Virtex-II FPGA family," in *ACM FPGA*, 2002, pp. 157–164.

[3] J. Rose and et al., "The VTR project," in *ACM FPGA*, 2012, pp. 77–86.

[4] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-Vdd," in *ACM/IEEE Design Automation Conference*, 2004, pp. 735–740.

[5] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. Irwin, and T. Tuan, "Reducing leakage energy in fpgas using region-constrained placement," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, USA, 2004, pp. 51–58.

[6] A. Bsoul and S. Wilton, "An FPGA architecture supporting dynamically controlled power gating," in *IEEE FPT*, 2010, pp. 1–8.

[7] T. Tuan, A. Rahman, S. Das, S. Trimberger, and S. Kao, "A 90-nm low-power FPGA for battery-powered applications," *IEEE Trans. on CAD*, vol. 26, no. 2, pp. 296–300, 2007.

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

[9] J. Ryan and B. Calhoun, "A sub-threshold FPGA with low-swing dual-VDD interconnect in 90nm CMOS," in *IEEE CICC*, 2010.

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

[11] G. Nabaa, N. Azizi, and F. N. Najm, "An adaptive FPGA architecture with process variation compensation and reduced leakage," in *ACM/IEEE DAC*, 2006, pp. 624–629.

[12] B. Bishop and M. Irwin, "Databus charge recovery: practical considerations," in *ACM/IEEE ISLPED*, 1999, pp. 85 –87.

[13] P. Sotiriadis, T. Konstantakopoulos, and A. Chandrakasan, "Analysis and implementation of charge recycling for deep sub-micron buses," in *ACM/IEEE ISLPED*, 2001, pp. 364–369.

[14] K. Patel, W. Lee, and M. Pedram, "In-order pulsed charge recycling in off-chip data buses," in *ACM Great Lakes Symp. on VLSI*, 2008.

[15] H. Wong, V. Betz, and J. Rose, "Comparing FPGA vs. Custom CMOS and the impact on processor microarchitecture," in *ACM/SIGDA Intl. Symp. on FPGAs*, 2011, pp. 5–14.

[16] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *ACM FPGA*, 1995, pp. 111–117.

[17] *LP solve reference (http://lpsolve.sourceforge.net/5.5/)*, 2012.

[18] S. J. Wilton, "Architecture and algorithms for field-programmable gate arrays with embedded memory," Ph.D. dissertation, 1997.

[19] J. Lamoureux and S. Wilton, "Activity estimation for field-programmable gate arrays," in *IEEE FPL*, 2006, pp. 1–8.