Reducing Dynamic Power Consumption in FPGAs Using Precomputation

Chi Chiu Tsang and Hayden K.-H. So

Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong
{cctsang,hso}@eee.hku.hk

***Abstract*—This paper studies the effectiveness of employing precomputation in reducing dynamic power consumption in commercial off-the-shelf (COTS) FPGAs. Precomputation is a high-level logic optimization technique that lowers power consumption of a design by disabling part of the circuit based on a few relatively simple precomputation conditions. With careful design considerations, the increased logic utilization and its associated power consumption can be justified by the power saving resulted from disabling a much larger part of the design. This fundamental trade-off benefits particularly well from the tile-based structures of modern FPGAs that consist of large number of redundant logic cells. Using the design of a comparator as an example, we study the trade-offs and unique opportunities provided by modern FPGA architectures in employing precomputation as a technique to reduce dynamic power consumption. In our example, 83% of dynamic power from logic, or 43.1% of total dynamic power including routing is reduced with negligible increase in resource consumption.**

I. Introduction

Reducing power consumption of a design running on field programmable gate arrays (FPGAs) is of prime concern not only for embedded systems, but also for high-performance computing systems where power consumption is quickly limiting their performances. A number of factors contribute to the total power consumption of an FPGA when it is configured. First, because of leakage current in deep sub-micron technologies, static power is consumed as soon as the device is powered. To tackle this problem, a number of researchers have already proposed solutions such as the use of high threshold (V_t) devices in FPGAs [1].

Another major contributor to the power consumption of a configured FPGA can be broadly classified as its dynamic power consumption that results from signal switching activities. As a first order estimation, the dynamic power consumption, P_d, of a device can be calculated as

$$P_d \propto \alpha \cdot C \cdot V^2 \cdot f_{clk} \tag{1}$$

where α is the probability that a signal switches, C is the load capacitance associated with the switching activities, V is the voltage swing of the signal, and f_{clk} is the clock frequency. In [2], dynamic voltage scaling was used to reduce power consumption, which tackled the factor V in (1). In this work, we aim to reduce dynamic power consumption of commercial off-the-shelf (COTS) FPGAs by reducing the effective load capacitance, $C_{eff} = \alpha \cdot C$. In particular, we study the effectiveness of the use of precomputation on FPGAs to reduce dynamic power consumption.

Precomputation [3] is a well-known logic optimization technique originally developed to reduce power consumption of application specific integrated circuits (ASICs). The basic idea of precomputation is to examine the input of a circuit module on each clock cycle to detect for conditions under which part of the complex circuit module may be disabled. In these cases, the output of the circuit module will be computed using a smaller sub-module or only part of the original module instead of the complete circuit.

Depending on the actual disabling mechanism, both static power [4] and dynamic power consumption may be reduced as a result. In this work, we limit our focus to reducing dynamic power consumption of COTS FPGAs by conditionally disabling switching activities in part of the design, thereby reducing α. This, however, comes at an expense of increased C. When designed correctly, an optimal effective capacitance C_{eff} can be achieved that results in much lower dynamic power consumption.

The contribution of this work can be summarized as follows:

- Identified precomputation as an effective dynamic power reduction technique for current generation COTS FPGAs using unmodified vendor software;
- Studied trade-offs in employing precomputation in current FPGA devices and unique opportunities provided by FPGA architectures that are absent in ASIC designs;
- Proposed architectural enhancements in next-generation FPGA devices that may further benefit from precomputation techniques in lowering power consumption.

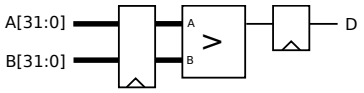
In the next section, using the design of a comparator as a case study, we study the use of precomputation on FPGAs. In Sec. III, we discuss the trade-offs in using such technique in current generation FPGAs and propose future enhancements. Finally we conclude the paper in Sec. IV.

II. Case Study: A Comparator

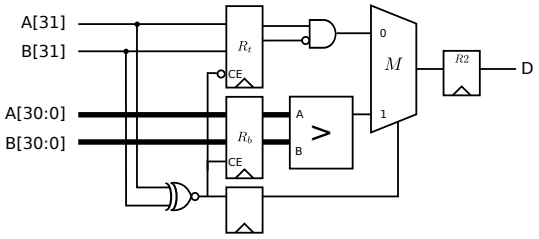
In this section, we present a detail case study of implementing a simple comparator using precomputation on a commercial off-the-shelf FPGA.

A. Base Case

The simple comparator of two 32-bit *unsigned* numbers shown in Fig. 1(a) forms the base case for our comparison. It compares the values of A and B and output 1 in D if $A > B$, 0 otherwise. The design was specified using behavioral VHDL, synthesized and implemented on a Xilinx



(a) A comparator.



(b) Comparator with simple precompuation on MSBs of both input.

Fig. 1. Reduce power consumption by precomputation on most significant bits of the input.

Virtex-5 device using standard vendor tools. The vendor's tools were constrained to utilize only resources from within one clock region on the chip to avoid power consumption due to unnecessary activations of regional clock buffers [5].

To obtain realistic dynamic power consumption figures, the design was fully placed and routed. The input A and B were then driven by a sequence of pseudo-random data for 1000 cycles. The resulting simulation trace was then used by the Xilinx XPower Analyzer tool to estimate power consumption.

B. Precomputation Transformation

Our first attempt in lowering dynamic power consumption of the comparator is based on the following observation: If the most significant bit (MSB) of A, A_{31}, is larger than the MSB of B, B_{31}, then $A > B$. Similarly, $A_{31} < B_{31} \Rightarrow A < B$. However, if $A_{31} = B_{31}$, then the values of the remaining bits of the input, $A_{30...0}$ and $B_{30...0}$, must be used to determine the final result. This observation leads to the implementation shown in Fig. 1(b).

Comparing to the original circuit in Fig. 1(a), the two 32-bit input registers, which is depicted as a single 64-bit register R to simplify the drawing, are spilt into a combination of a pair fo single bit register (R_t) and a 2×31-bit register (R_b). The MSB of the input are connected to the top register R_t while the remaining bits are connected to the bottom register R_b.

The loading of input data into the input registers are controlled by the result of comparing the values of A_{31} and B_{31}. If their values are different, then R_b is disabled while A_{31} and B_{31} are loaded into R_t. In the next cycle, the registered values of A_{31} and B_{31} are used to compute the final result of the comparison, D, i.e., $D = A_{31}\overline{B_{31}}$. This result is selected by the final multiplexor and the result is registered into the output register.

On the other hand, if the values of A_{31} and B_{31} are identical, then the bottom halves of the input registers, R_b is enabled. The remaining bits of A and B are loaded and a full 31-bit comparison must be performed in the next cycle to determine the value of D.

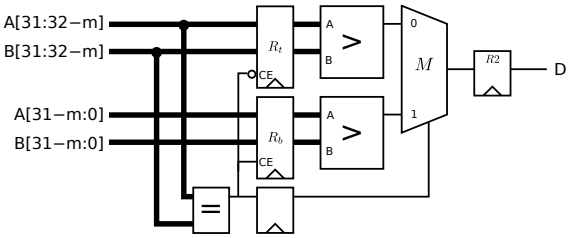


Fig. 2. Comparator with m most significant bits precompuation on the input.

To estimate the potential power saving from such transformation, note that in each cycle, internal signal switching activities occur only in either the top or the bottom half of the circuit, which is determined by the values of A_{31} and B_{31}. The expected dynamic power consumption is therefore:

$$E(power) = P(A_{31} \neq B_{31}) \cdot W_t + P(A_{31} = B_{31}) \cdot W_b + overhead \quad (2)$$

where W_t and W_b are power consumption of the top and bottom half of the circuit respectively. Since the input A and B are random, $P\big(A_{31} = B_{31}\big) = P\big(A_{31} \neq B_{31}\big) = 1/2$. As a first order estimation, we may assume that the 31-bit comparator in the bottom part consumes approximately the same power as the original unmodified 32-bit comparator, while the top half requires relatively negligible power. As a result, excluding the overhead of precomputation transformation, the expected dynamic power consumption should be reduced by approximately 50% when compared to the original circuit.

C. Multi-bit Precomputation Transformation

In general, instead of examining just the single most significant bit of an input, the precomputation logic presented above can be applied to the general case where the m most significant bits of an input are compared. In other words, observe that:

$$A_{31,...,32-m} > B_{31,...,32-m} \Rightarrow A > B \quad \text{and} \quad (3)$$

$$A_{31,...,32-m} < B_{31,...,32-m} \Rightarrow A < B. \quad (4)$$

Based on this observation, a multi-bit percomputation implementation of the comparator can be realized as shown in Fig. 2. In contrast to the design shown in Fig. 1(b) where only the MSBs are examined, comparing the m most significant bits decreases the probability that the power-consuming comparator in the bottom must be used. In particular,

$$P(A_{31,...,32-m} = B_{31,...,32-m}) = \frac{1}{2^m}.$$

However, as m increases, the logic complexity of the top half of the circuit increases. When compared to the architecture depicted in Fig. 1(b), the circuit in Fig. 2 must employ a small, m-bit comparator in the top-half of the circuit. Furthermore, an m-bit equality comparison must be performed as part of the precomputation. Both of these additional circuitries incur extra power consumption that will inevitably offset the power conserved by disabling the bottom half of the circuit. As a result, there is an optimal m such that the saving in power consumption is maximal.

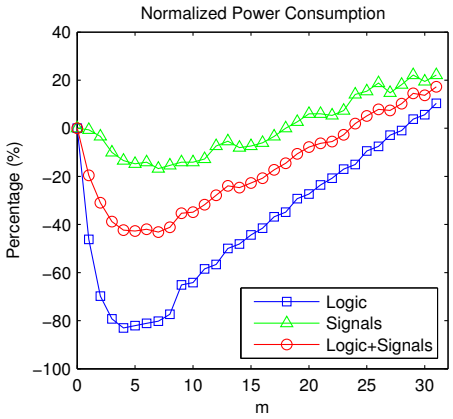


Fig. 3. Power consumption of comparators with precomputation normalized against standard compartor without precomputation.

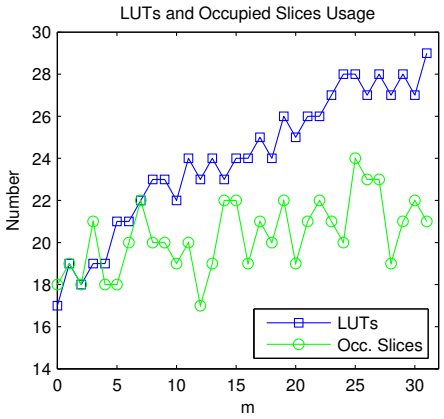


Fig. 4. Logic resource utilization of comparators with precomputation.

Fig. 3 shows the normalized power consumption of comparators with precomputations on the m most significant bits against the base case comparator without any precomputation to highlight power saving due to precomputation. In the diagram, logic power consumption (marked as "Logic") refers to the dynamic power consumption incurs by logic elements such as LUTs. Signal power consumption (marked as "Signals") refers to the dynamic power consumption incurs from switching activities of signal routes.

From the result, it can be observed that the reduced switching activity provides significant savings in logic power consumption. The best saving is obtained in the case of $m = 4$ where logic power consumption is reduced by 83%. Beyond that, the saving in logic power consumption is reduced as the size of the additional equality comparison increases with m. However, the effect of this power overhead is mellow and only out-weight the saving in reduced activities when $m \geq 29$.

The power consumption arising from signal activities in the interconnect network is also reduced. The best saving is obtained in the case of $m = 7$ where the signal power is reduced by 16.8%. When compared to the reduction in lower power consumption, the reduction in signal power is limited by the rapid increase in routing complexities as m increases. Furthermore, when compared to the base case comparator, the large number of clock enable (CE) signals that control all the input registers incur a large amount of dynamic signal power consumption.

Nevertheless, the overall power saving due to precomputation with respect to the sum of logic and signal power consumption remains encouraging. At $m = 7$, a saving of 43.1% is obtained.

Note that when clock and I/O power are also taken into account, the total dynamic power saving is reduced to only 7.68%. This is mainly due to the fact that only one single comparator is implemented in the entire FPGA, making the I/O and clocking overhead relatively high. In a real system where a large number of computational blocks are employed, such overhead is expected to be much lower.

III. FPGA Advantages

In this section, we highlight features unique to FPGAs that make precomputation-based optimization a particularly viable option for lowering dynamic power consumption.

Recall that the fundamental trade-off in employing a percomputation-based architecture is between the decrease in power consumption due to lower signal activities and the increase in power consumption due to the additional logic and interconnect required for precomputation. Because of their architectures, modern FPGAs provide unique opportunities to lower such overhead, making precomputation an appealing technique.

At the lowest level of an SRAM-based FPGA, the core of its reconfigurable fabric consists of an array of lookup tables (LUTs) and flip-flops (FFs) connected through a programmable interconnect network. Often time these reconfigurable fabrics are grouped into larger blocks both physically and conceptually. For example, in a Xilinx Virtex-5 device, 4 LUTs and 4 FFs, as well as a handful of other logical devices are packed physically as a *slice*. By grouping these low-level reconfigurable fabrics into a macro-block, it is possible to implement dedicated routing and configuration options that provide higher performance and lower power consumption. However, also because of such grouping, it is becoming much harder for even proprietary vendor software to pack unrelated logic into the same macro-block. As such, many vendor software would report such a block as being occupied even when not all of the low-level reconfigurable fabric are utilized.

Fig. 4 depicts the logic resource utilizations of the comparators with precomputation on the m most significant bits. As expected from the precomputation architecture, the number of computational logic in terms of LUTs being utilized increases as m increases. However, the number of occupied slices remains fairly stable over the entire range. When examined closely, it can be seen that most of the slices are occupied to provide FFs as input registers, leaving the LUTs unused. As m increases, more LUTs are demanded. As these additional circuits are closely related to the existing design, many of them can easily be packed into the slices that were originally

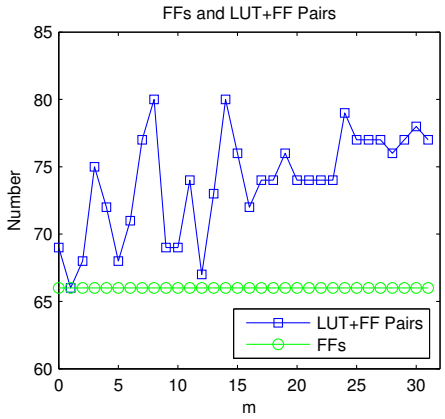


Fig. 5. Flip-flops utilization of comparators with precomputation.

configured with only FFs.

Such increase in packing efficiency is illustrated in Fig. 5 as the number of "LUT+FF pairs" increases with m. In the figure, the number of flip-flop used remains constant at 66 for all cases where $1 \leq m \leq 31$, with 64 of them act as input registers; 1 holds the result of the equality comparison; and 1 stores the output result. A "LUT+FF pair" is formed when both the LUT and its adjacent FF within the same slice are utilized in a design. Based on this notation, the increase in "LUT+FF pair" count in Fig. 5 clearly indicates that more and more logic are packed into previously unoccupied LUTs adjacent to the already utilized FFs within the same slice as m increases.

In other words, these extra functionalities for precomputation appear to have incurred almost no new resources on an FPGA. Although this phenomenon may seem to be a mere accounting discrepancy, it does illustrate a number of important opportunities for precomputation-based techniques to be used on FPGA devices. First, unless FPGA tools can effectively pack unrelated logic into unoccupied LUTs and FFs of a slice, many of them will remain underutilized in the final design. Such packing is particularly difficult when a module-based design flow is employed where most modules were independently pre-placed-and-routed. In contrast, additional precomputation logic can easily be placed-and-routed together with the original logic as a module, making the vendor's tools much easier to take advantage of the unused LUTs and FFs.

Furthermore, as FPGA devices are migrating onto newer technology nodes, new techniques are needed to cope with the ever-increasing leakage power. One way is to employ multi-V_t or multi-V_{dd} techniques in future FPGA devices. By controlling the supply and threshold voltage, the performance and leakage power of different parts of the device can be independently controlled during run-time. Due to cost and technology limitations, blocks of configurable logic are likely to form the basis of control. In these devices, precomputation-techniques will again be able to take advantage of the under-utilized resources.

Finally, apart from reducing dynamic switching activities through the disabling of FFs, precomputation may also be employed to dynamically control the activation of different parts of the device by physical means [4]. Future FPGA architectures may therefore consider having dedicated routings to allow for power-efficient, fine-grained and low-level control of the device during run-time using user-defined precomputation techniques.

IV. Conclusion

In this paper we have presented the precomputation-based optimization technique as an effective option to reduce dynamic power consumption on commercial off-the-shelf FPGAs without any need of special software. The design of a simple comparator has been used as a case study to illustrate the working principle of such technique. By appending a small precomputational logic to the original design, a reduction of 84% in dynamic logic power consumption and 43% combined logic and signal power consumption has been achieved. Such technique is applicable to current commercial FPGAs and may be integrated seamlessly into existing module-based design methodologies.

The fundamental trade-off associated with such technique is between the saving in dynamic power consumption due to lowered signal activities and the increased power consumption due to the addition of precomputational logic. We have argued that the redundancy presented in modern SRAM-based FPGAs provides an excellent platform that favors such trade-off.

Furthermore, similar precomputation-based optimization technique is readily applicable to future FPGA architectures. It is expected that configurable logic resources in these future architectures will be grouped into moderately sized blocks that may be turned on or off dynamically during run-time for sake of reducing leakage power consumption. If suitable low-level control mechanisms are made available by the architecture, it will be possible to employ similar precomputation-based technique as described in this work to control the activation of different regions in these future devices.

References

[1] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using pre-defined dual-Vdd/dual-Vt fabrics," in *FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*. New York, NY, USA: ACM, 2004, pp. 42–50.

[2] C. Chow, L. Tsui, P. Leong, W. Luk, and S. Wilton, "Dynamic voltage scaling for commercial FPGAs," in *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*, Dec. 2005, pp. 173–180.

[3] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 2, no. 4, pp. 426–436, Dec 1994.

[4] A. Abddollahi, M. Pedarm, F. Fallah, and I. Ghosh, "Precomputation-based guarding for dynamic and leakage power reduction," in *Computer Design, 2003. Proceedings. 21st International Conference on*, Oct. 2003, pp. 90–97.

[5] L. Wang, M. French, A. Davoodi, and D. Agarwal, "FPGA dynamic power minimization through placement and routing constraints," *EURASIP J. Embedded Syst.*, vol. 2006, no. 1, pp. 7–7, 2006.