Dynamic Clock Management for Low Power Applications in FPGAs

Ian Brynjolfson and Zeljko Zilic
Department of Electrical and Computer Engineering, McGill University
Montreal, Quebec, Canada

Abstract

Low power techniques employing dynamically controlled clock rates offer potentially powerful energy saving capabilities. In this paper, we consider the application of this low power technique to FPGAs, where we reduce energy waste in clock distributions. We show that current FPGA clock managers are inadequate for use in dynamically controlled systems. We provide an architectural block, the dynamic clock divider, that can be added either internally to clock managers or as user logic, to allow dynamic clock management.

Introduction

Until recently, Field Programmable Gate Array (FPGA) designs have focused on the optimization of performance and area (1, 2). Due to their programmability, FPGAs are inherently more energy-inefficient than the equivalent ASIC implementation. The research on reducing the FPGA energy consumption has been directed at interconnect (1), the combinational logic architecture, and the clock distribution network (3). In commercial FPGAs, the effort was directed at operating the FPGA core at a reduced voltage level. The outlined approaches lead to power reduction at the expense of the interconnect speed.

An alternative to these circuit-level techniques is a system-level technique involving dynamic clock management. This technique leads to potentially powerful energy savings (4,5) without necessarily incurring a performance hit.

Low Power Using Dynamic Clock Management

There are four areas to consider when reducing the power consumption of systems implemented in an FPGA. Dynamic power consumption is given by the equation:

$$P_d = kV_{DD}^2 C_L f$$

where V_{DD} is the power supply voltage, C_L is the load capacitance, f is the frequency of switching and k is a switching activity factor. Low power techniques attempt to reduce the power supply voltage, the load capacitance, or switching activity in a circuit. One can also explore the direct proportionality between the operating clock rate and power consumption (6). If the system clock is slowed to the minimum rate that will meet the computational requirements, both the instantaneous power and the overall energy consumption will be reduced without loss of performance.

While the reduction of power consumption happens whenever the switching activity is reduced, only asynchronous design techniques (7) can use that reduction to the full. In synchronous systems, existence of large clock distribution networks diminishes the effects of switching activity reduction in logic gates alone.

This phenomena is especially present in FPGAs due to the large capacitances associated with the routing switches, especially for global signals such as a clock (8). Due to high capacitances associated with global programmable interconnect, approximately 25% of the overall dissipation of energy in an FPGA can be due to the clock signal. This ratio can rise above 50% in highly pipelined circuits (3). Hence, the power reduction in synchronous systems can be realized to its full potential either by clock gating, or by controlling the clock rate based on computational demand. While clock gating has been used widely, demonstrations of the latter technique for reducing power consumption have been presented only recently(9). Studies (4, 5) show that the dynamic clock scaling is an efficient power reduction technique with large potential power savings.

Clock management circuitry capable of performing clock synthesis include programmable frequency multipliers and dividers. As a result, scheduling algorithms (4, 10) can be used to control the division/multiplication of the system clock to meet the demand, without wasting energy. As the computational demand in separate subsystems changes, each subsystem can operate independently at a speed that will allow it to satisfy current individual demands, as shown in Fig. 1.

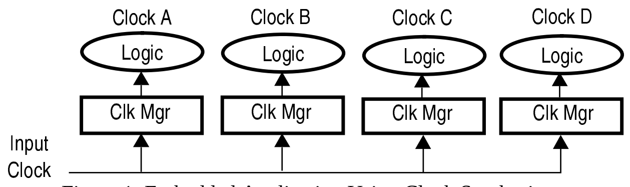

Figure 1: Embedded Application Using Clock Synthesis

Dynamic frequency scaling can be used in conjunction with other methods, such as dynamic voltage scaling. When the clock is slowed, a controlled voltage source can reduce the system voltage level and still maintain the switching speed. For comparison, dynamic voltage scaling does not bring any benefit in conjunction with clock gating.

Dynamic Clock Management Implementations

We propose three scalable implementations of dynamic frequency scaling needed to perform the proposed low power methods. The simplest dynamic clock management circuit is an open-loop implementation with a clock divider inserted into the desired paths, shown in Fig. 2.a. The maximum desired clock rate must be input to the system, to be reduced dynamically using the clock divider. This circuit is inexpensive and simple, but may introduce excessive clock skew.

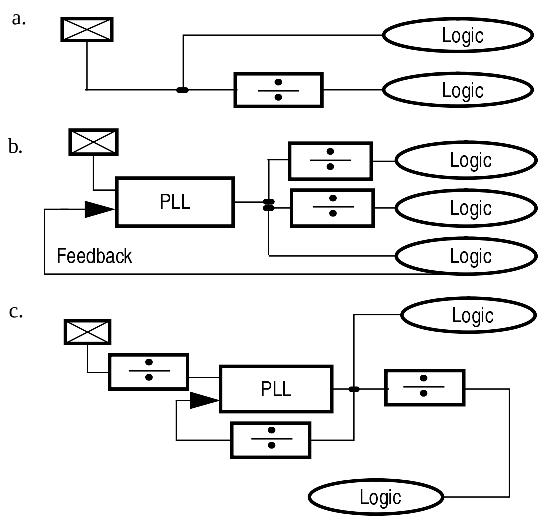

Figure 2: Dynamic Clock Management Implementations

Skew can be compensated by introducing a Phase Locked Loop (PLL) into the circuitry. The simplest dynamically scaled structure is obtained by taking feedback from a point that does not change frequency, as shown in Fig. 2.b. This scheme can successfully apply dynamic clock division. For dynamic multiplication, the signal in the feedback path must be divided, as in Fig. 2.c.

In the case of a large change in input frequency, the output of the PLL may take a long period to settle and regain a lock on the input signal. During this time, the PLL will align the output improperly. The circuitry dependent on the output clock should then be stalled. To avoid an invalid output, the frequency transitions of the input or feedback path must be guaranteed to happen within the lock range (11).

FPGA Clock Management

Dynamic clock management can have a strong impact on the reduction of power consumption in FPGAs. Most modern, FPGAs have dedicated clock managers for solving high speed clock distribution problems in high density designs. They include PLLs and/or Delay Locked Loops (DLL) together with clock dividers and interface circuits. The flexibility and programmability they provide is critical to the scope of applications they can support (12). Several clock managers have been introduced in FPGAs. The three characteristic examples are by Xilinx (13), Altera(14) and Lucent(15). The FPGA clock managers can perform clock buffering, drive the distribution networks, and simultaneously eliminate clock skew. They can also produce phase shifts, and duty cycle adjustment. In addition, clock managers can be used to synchronize several components in a system.

Although they perform their intended function well, current FPGA clock managers are incapable of performing dynamic clock management because their dividers cannot perform dynamic division or multiplication. The Xilinx and Altera clock managers can only be programmed during initial configuration. The Lucent Programmable Clock Manager can be programmed during its operation, but this can lead to dangerous clock outputs, as explained next.

Dynamic Programmable Clock Divider

Changing the settings of existing clock dividers during their operation can lead to metastability and latching errors due to glitches, distortions, asymmetry, transient frequencies and additional clock edges of the output clock signal, as shown in Fig. 3. Even shutting off the system during the change to the new frequency does not help in that case, as the inconsistent duty cycle clocks may be non-transient.

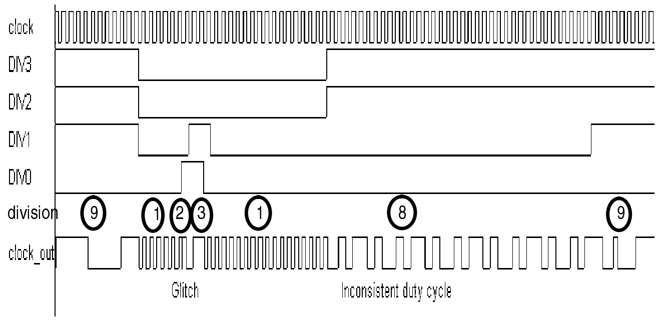

Figure 3: Standard Clock Divider

If dynamic clock dividers are added to the existing FPGA clock managers, they can be used for dynamic clock management. As a result, clean frequency changes can take effect within a clock cycle. The Dynamic Programmable Clock Divider (DPCD), of Fig. 5, is capable of performing dynamic frequency division without undesired effects at the output. Division of the input clock is performed by creating a loop of D-flip-flops {**A**-**D**} driven by the input clock, and feeding the signal back into the loop through an inverter {**D**} to create the necessary clock level inversion. To expand the length of the output clock period, the number of D-flip-flops in the loop is increased by multiplexer {**L**}. In order to perform odd division, flip-flops {**E**, **F**} extend the loop, by half a period, with an asynchronous clear of flip-flop {**A**} on the falling edge of the input clock. For the divider output, multiplexer {**N**} chooses between the original input clock, for a division value of one, and the output of {**A**}.

To prevent output glitching, D-flip-flops {**G**,**H**,**J**,**K**} latch the new program value on the rising edge of the output from {**A**}. Combinational logic {**Q**,**R**,**S**} also help to prevent glitching, but also prevent transient patterns from being captured and fed back, thus causing irregular oscillations in the circuit (such as in Fig. 3, division by 8).

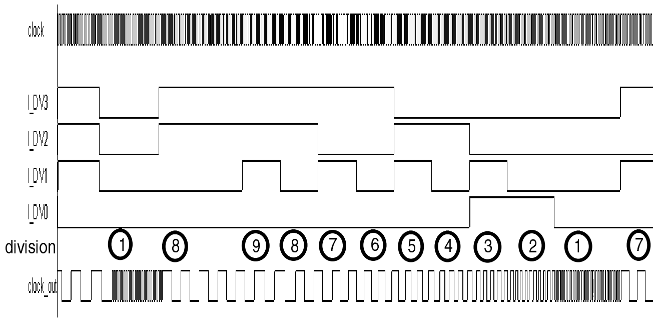

Figure 4: DPCD Operation

Another method of dynamic frequency synthesis is performed with a chain of dividers capable of only even divisions, as in (6). Each divider would statically divide the clock to each successive frequency, and a multiplexer would be used to choose the desired frequency. However, the DPCD, being smaller and more power efficient, is a single circuit capable of performing divisions from 1 to 9. The DPCD design in Fig. 5 is expandable to greater division capabilities by simply adding more D-flip-flops in series.

Case study

An embedded system coprocessor has been designed to speed up digital signal processing functions. It takes advantage of dynamic clocking for reducing its power consumption. The system comprises of a Universal Serial Bus (USB) client communication front-end, and a reconfigurable signal processing unit. Currently, we consider a matrix multiplication circuit. Power to the coprocessor is delivered by the USB cable from the host. Depending on the frame rate and data stream format, the bandwidth requirements of the multiplier will change. Therefore, its required clock rate is based on the bandwidth demand. Also, the standard protocol for USB includes two operating speeds.

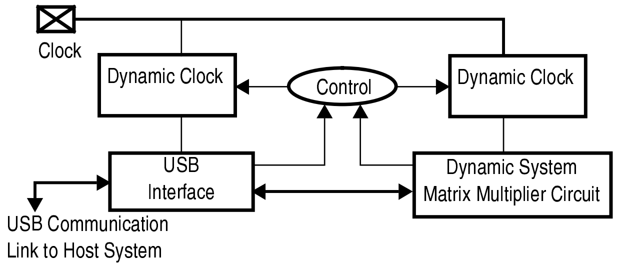
Clock
Dynamic Clock
Control
Dynamic Clock
USB Interface
Dynamic System Matrix Multiplier Circuit
USB Communication Link to Host System

Figure 6: USB/Matrix Multiplier System

Control logic receives information from the system subcomponents and updates the control registers of the clock managers. Other speed-setting control algorithms (10) may be performed locally within the coprocessor. When a transfer from the USB host is preceded by a low speed preamble packet, the USB interface signals the control logic to reduce its clocking speed. The USB host will use the low speed to poll the client and receive information regarding the quantity of data to be transferred. The host then determines the communication speed. Most of the USB circuit is shut off after the final end-of-packet signal, for that transfer, is received. Full speed operation is resumed when communication recommences. Communication between the USB interface and the matrix multiplier may be performed using periodic synchronization methods or FIFOs.

Table 2 shows power consumption of system components running at different clock rates for Altera and Lucent implementations. Fig. 7 and Fig. 8 show power consumption vs.

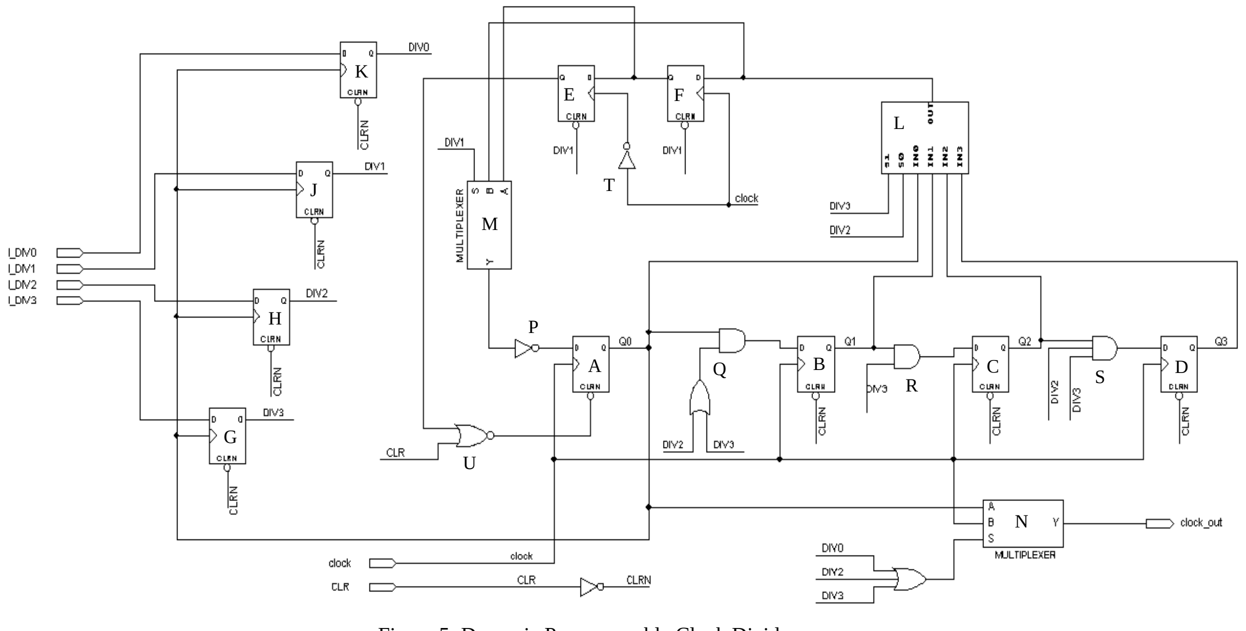

Figure 5: Dynamic Programmable Clock Divider

operating frequency based on manufactuer estimates. The total power consumed for typical scenarios of system use is outlined.

Table 1: Implementation Details and Power Consumption*

	ORCA 3T55-6				
	f_{MAX}	size	P_{fMAX}	P_{12MHZ}	$P_{1.5MHz}$
USB	21.69 MHz	115 PFUs	228.4 mW	126.3 mW	15.8 mW
Multiplier	28.48 MHz	136 PFUs	326.5 mW	114.7 mW	22.9 mW
	Altera EPF10K20-3				
	f_{MAX}	size	P_{fMAX}	P_{10MHz}	P_{2MHz}
USB	30.48 MHz	703 LCs	1241 mW	519 mW	108.6 mW
Multiplier	17.12 MHz	563 LCs	584.14 mW	363 mW	112.6 mW

*Based on results and formulae provided by manufacturers

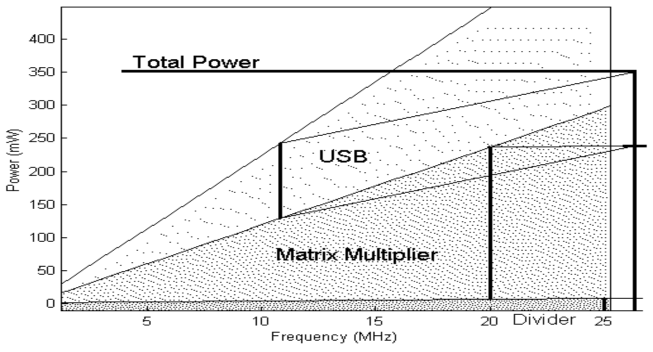

Figure 7: ORCA Implementation

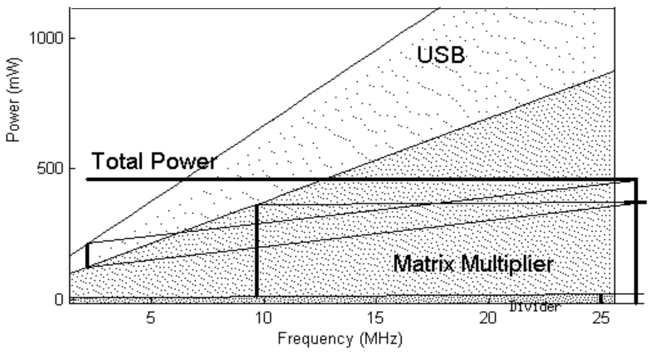

Figure 8: Altera Implementation

Conclusion

Current low-power methods for FPGAs use circuit-based techniques at the penalty of performance. We have introduced a system-level technique that takes advantage of the clock managers present in most modern FPGAs. Dynamic frequency scaling is used to reduce the average power consumption. A dynamic programmable clock divider was presented that enables dynamic operation by eliminating glitches, transient and non-transient divider output errors that are very harmful when introduced in clock signals. The divider can be either included as a part of the FPGA clock managers, or as a user circuit. The results show that this technique can be used efficiently in two FPGA families.

Acknowledgments

Thanks to Stuart McCracken and David Peppy. This work was financially supported by MICRONET.

References

(1) E. Kusse and J. Rabaey, “Low-energy embedded FPGA structures,” *1998 International Symposium on Low Power Electronics and Design*, Aug. 1996, pp155-160.

(2) V. Betz and J. Rose, “Circuit design, transistor sizing and wire layout of FPGA interconnect,” *IEEE 1999 Custom Integrated Circuits Conference*, May 1999, pp. 171-174.

(3) V. George, H. Zhang, and J. Rabaey, “The design of a low energy FPGA,” *1999 International Symposium on Low Power Electronics and Design. Proceedings*, Aug. 1999, pp188-193.

(4) V. Krishna, N. Ranganathan and N. Vijaykrishnan, “Energy Efficient Datapath Synthesis Using Dynamic Frequency Clocking and Multiple Voltages”, *Proc. of 12th International Conference on VLSI Design*, pp. 440-445, Jan. 1999.

(5) T. Pering, T. Burd and R. Brodersen, “The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms”, *Proc. of International Symposium on Low-Power Electronics and Devices,* pp. 76-81, Monterey, Aug. 1998.

(6) N. Ranganathan, N. Vijaykrishnan, and N. Bhavanishankar, “A linear array processor with dynamic frequency clocking for image processing applications”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8. No. 4, Aug. 1998, pp. 435 -445.

(7) S. Hauck, “Asynchronous Design Methodologies: An Overview”, *Proceedings of IEEE*, Vol. 83, No. 1, Jan. 1995, pp. 69-93.

(8) J. Cong, L. Hei, C. Koh, and Z. Pan, “Global interconnect sizing and spacing with consideration of coupling capacitance,” *IEEE/ACM International Conference on Computer Aided Design*, Nov. 97, pp628-633.

(9) M. Olivieri, A. Trifiletti, and A. De Gloria, “A low-power microcontroller with on-ship self-tuning digital clock-generator for variable-load applications,” *IEEE International Conference on Computer Design*, Oct. 1999, pp233-240.

(10) K. Govil, E. Chan, H. Wasserman, “Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU”, *Proceedings of 1st ACM International Conference on Mobile Computing and Networking*, 1995.

(11) Best, Roland E., *Phase-Locked Loops - Design, Simulation, & Applications*, Mcgraw-Hill, New York, 1997.

(12) P. Sasaki, Y. Bobra, W. Cory, A. Ghia, S. Menon, M. Kola, M. Thomas, P. Rau, A. Zaliznyak, “A Fast Predictable FPGA with PLLs, Dual Port SRAMs and Active Repeaters,” *1999 IEEE Custom Integrated Circuits Conference*, May 1999, pp179-182.

(13) Xilinx Inc., *Using the Virtex Delay-Locked Loop (XAPP132 Version 1.31),* Advanced Application Note, October 21, 1998.

(14) Altera Corporation, *Using the ClockLock & ClockBoost Features in APEX Devices*, Application Note 115 (ver. 1.0), May 1999.

(15) Lucent Technologies, *ORCA Series 3C and 3T FPGAs*, Data Sheet, June 1999.