Low-Latency, HDL-Synthesizable Dynamic Clock Frequency Controller with Self-Referenced Hybrid Clocking

Robert M. Senger,
Eric D. Marsman
University of Michigan
Ann Arbor, MI 48109-2122, USA
{emarsman, rsenger}@eecs.umich.edu

Gordon A. Carichner, Sundus Kubba,
Michael S. McCorquodale
Mobius Microsystems, Inc.
Detroit, MI 48226-1686, USA
{carichner, kubba, mccorquodale}@mobiusmicro.com

Richard B. Brown
University of Utah
Salt Lake City, UT 84112-1109, USA
brown@coe.utah.edu

***Abstract*—A low-latency, HDL-synthesizable dynamic clock frequency controller is presented as a time-efficient alternative to full-custom implementations. Frequency division of a fully integrated hybrid temperature-compensated LC oscillator (TC-LCO) and ring oscillator clock reference avoids PLL locking delays to enable low-latency, hazard-free frequency selection on an actively running CPU. Fabricated in 0.18µm CMOS as part of a low-power SoC microsystem, the circuit dissipates 480µW at 1.8V.**

I. INTRODUCTION

The ability to dynamically scale CPU clock frequency and power supply voltage with workload has become an important technique for reducing active and standby power consumption in nanoscale embedded systems. Dynamic frequency scaling (DFS) and dynamic voltage scaling (DVS) have been used successfully to reduce power in portable embedded applications such as PDAs and cell phones. Recently, even high-performance microprocessors such as Intel's dual-core Montecito have adopted complex dynamic voltage and frequency scaling (DVFS) control circuits to maintain power and temperature within acceptable limits [1].

Many DFS circuits are implemented with a PLL, which is used to multiply a low frequency reference signal that is typically derived from an external crystal oscillator (XO). The PLL prescaler can be changed to generate a new clock frequency, however, even relatively fast-locking PLLs incur a delay on the order of microseconds to regain lock after the prescaler has been changed [2][3]. During this time the system clock is typically unusable, which can easily translate into thousands of missed CPU cycles. To avoid stopping the CPU, dual-PLL DFS architectures such as [4] use one PLL to produce a usable system clock while the second PLL's prescaler is changed. After the second PLL has locked onto the new frequency, the system clock can be switched. This approach is problematic when transitioning from low to high frequency to handle sudden, unexpected increases in workload as might occur in interrupt-driven embedded systems. For real-time applications, the delay to lock the second PLL on a higher frequency can result in unacceptable performance degradation. DFS architectures proposed in [2] and [5] avoid this problem by running a single PLL at high frequency and dividing the clock down. Using a frequency divider on the output allows for nearly instantaneous frequency switching without incurring the lock penalty associated with changing the PLL prescaler. However, by maintaining the PLL at a high frequency and using clock division, the PLL dissipates more power than if the prescaler were reduced during times of low CPU activity.

Many software algorithms have been developed to manage voltage and frequency scaling policy in processors [3][6]-[8]. Such algorithms can be divided into two main categories, course and fine grain scaling. Fine grain algorithms are typically more effective in reducing the energy-delay product because they scale frequency/voltage more often to match changes in workload. The main concern when deciding whether to scale voltage and frequency is the DVFS switching delay. This is basically the transition/switching time and typically includes some amount of idle time for the processor waiting for PLL relocking. Because of this DVFS switching cost, adaptive DVFS action should only be triggered for large workload changes such that the net energy-delay product improvement is greater than the switching

This work was supported primarily by the Engineering Research Centers Program of the National Science Foundation under award number EEC-9986866.

cost. Therefore, for adaptive DVFS control, the choices of the triggering condition and the amount of voltage/frequency adjustment should be based on the switching cost [6]. According to [7], the minimum quantum of time for scaling frequency/voltage must be at least two to three orders of magnitude larger than the switching latency. The aim of this work is to reduce the switching latency to enable very fine-grained DFS.

II. DFS Architecture

Most DFS circuits require full-custom design with careful transistor level simulation to avoid glitches on the clock during frequency transition. In this work, we present an HDL-synthesizable, low-latency, glitch-free dynamic clock frequency controller that switches frequencies without halting the CPU. Rather than a traditional bottom-up approach using a low frequency external XO reference that is multiplied by an on-chip PLL, our implementation employs an all-silicon hybrid TC-LCO to eliminate the need for both the PLL and external reference. The TC-LCO produces a high-accuracy, low-drift, and low-jitter 1GHz reference that is frequency divided and squared. A ring oscillator is also provided for low power standby mode. The proposed DFS circuit has been integrated on an SoC with a microcontroller unit (MCU) and DSP core and fabricated in *TSMC*'s 0.18µm MM/RF CMOS process. This SoC was designed to control Wireless Integrated Microsystems (WIMS) as would typically be found in remote sensor or biomedical applications [9]. A previous generation of the WIMS Microsystem is described in [10].

Fig. 1 shows the novel DFS circuit. A simple flip-flop chain divides the monolithic clock reference ($2f_0$) to produce eight frequencies ranging from f_0=100MHz to f_7=781kHz if the TC-LCO is enabled and from f_0=10MHz to f_7=78.1kHz if the ring oscillator is enabled. The eight selectable frequencies supply two software controlled 8-to-1 multiplexers that provide separate clocks to the MCU and DSP cores. The multiplexers' outputs (f_{MCU}, f_{DSP}) are not hazard-free and require parallel chains of synchronizing flip-flops (FF0, FF1)

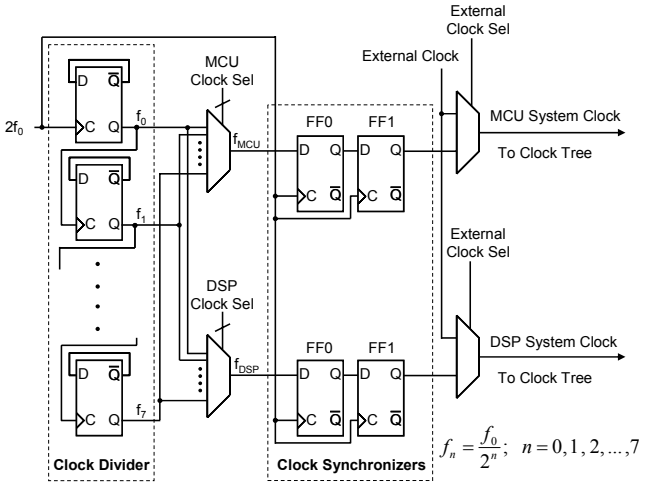


Figure 1. HDL synthesizable glitch-free dynamic frequency controller.

to remove glitches and eliminate metastability that might occur when switching between frequencies. 2-to-1 multiplexers select between the on-chip clocks and an external clock, however, these multiplexers cannot be changed dynamically. This DFS circuit can easily be expanded to provide additional, independently selectable clocks if required.

Glitches on the clock signal can result in logic failure through two primary mechanisms. If the glitch is small enough that some downstream clocked elements treat the glitch as if it were a clock pulse and some do not, this can cause a failure. Alternatively, if the glitch occurs too close to an adjacent clock edge, this can lead to a timing failure. By clocking the synchronization flip-flops with $2f_0$, the delay between consecutive rising (falling) and falling (rising) edges on FF0.Q is forced to be greater than or equal to the half-period of f_0. Assuming the digital logic is designed to operate at a maximum frequency f_0, this implementation will prevent both of the aforementioned glitch-related timing errors even in latch-based designs. Fig. 2 shows examples of how the circuit suppresses glitches on the f_{MCU} multiplexer output during frequency transitions. As the software dynamically switches the clock selection multiplexer between f_0, f_2, and f_1, glitches occur on f_{MCU} at the instant of the switch. These glitches are restricted to frequency f_0 by FF0 and will not cause logic malfunction in the MCU core.

Although only FF0 is required for glitch suppression on the selected clock, a metastable value could be latched into FF0 if f_{MCU} were sampled while in the process of transitioning. This is possible if the output of the clock multiplexer violates setup or hold time on FF0. Such timing problems can be avoided through careful design and simulation of the various timing paths through the DFS circuit. However, in a fully synthesized design where there is limited control over the synthesized result it can be difficult and time consuming to guarantee the necessary timing accuracy to completely avoid metastability in FF0.

As a relatively simple solution to this problem, FF1 was added to minimize the probability of metastability on the clock output. To approximate the Mean Time Between Fail-

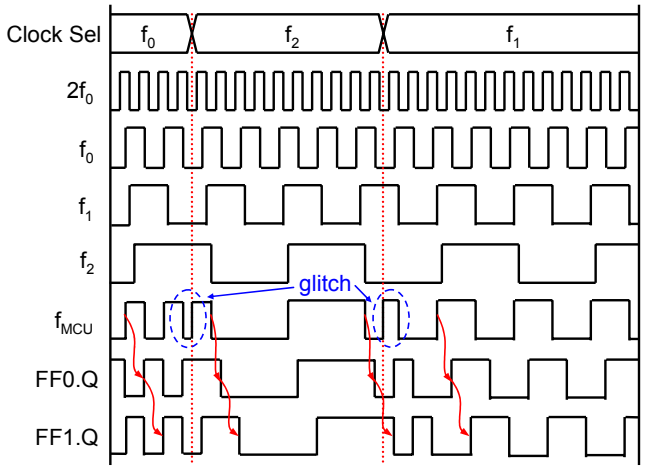


Figure 2. Glitch suppression on f_{MCU} during frequency transitions.

ures (MTBF) caused by a metastable output from FF1, the equations from [12] can be used:

$$MTBF = \left(t_a f_{clk} 2 f_0 \exp\left(\frac{-t_w}{\tau_s} \right) \right)^{-1} \quad (1)$$

$$t_w = \frac{n-1}{2 f_0} \quad (2)$$

From (1) and (2), f_{clk} is the clock frequency currently selected, $2f_0$ is the synchronization flip flops' sampling frequency, t_a is the flip flop aperture time, τ_s is the regeneration time constant, and n is the number of synchronization flip flops. Both t_a and τ_s can be approximated at 200ps for this analysis. For the current configuration where n=2 flip-flops, the worst case MTBF occurs when $f_{clk}=f_0$=100MHz. At this speed, the DFS circuit would be expected to have one metastability failure in 150 years of operation. Adding a third flip flop to the synchronization chain would increase the MTBF to 2.85×10^{18} years at the expense of another cycle of clock switching latency.

The proposed DFS circuit is particularly well suited for integration with high frequency TC-LCOs because the $2f_0$ required for synchronization is already generated when using a FF chain to divide the LCO's high oscillation frequency. LCOs must operate at high frequencies to minimize inductor area [13]. An additional benefit of dividing a high frequency reference clock by N is that the relative period jitter for the divided clock is reduced by a factor of $N^{1/2}$. In contrast, when a low-jitter XO reference is multiplied N times using a PLL, the relative period jitter is increased by $N^{1/2}$ [14]. It is worth noting that our DFS circuit could operate in the GHz range or could be used with a PLL as the clock source, however power consumption would increase in order to produce $2f_0$.

III. Hybrid Clock Synthesizer Architecture

The self-referenced hybrid clock synthesizer, shown in Fig. 3, includes a free-running RF LCO, a low power ring oscillator, a temperature-compensated bias circuit and an arbiter mux [15] for asynchronous glitch-free switching between the two oscillators. The synthesizer supports a reduced-power standby mode in which the TC-LCO is powered down while the system operates from the low power, low frequency ring oscillator. Although well suited for two or three frequencies, the arbiter mux does not scale well as the number of input frequencies increases. Also, the arbiter mux's switching speed is limited by the slowest input frequency making it too slow for most DFS applications.

The RF LCO includes a complimentary cross-coupled negative-transconductance sustaining amplifier, a differential inductor, and a bank of switched capacitors in parallel with the *LC* tank. The LCO generates a 1GHz reference signal that is followed by a frequency divide-by-5 circuit. Frequency deviation due to process variation is corrected by trimming the capacitance in the tank using an 8-bit calibration byte. The measured calibration range is ±10.75%, giving an initial calibrated accuracy of ±420ppm at 25°C. The ring oscillator has five differential stages and outputs a 20MHz signal that can be calibrated via the digital interface to account for process variation. The LCO dissipates 9.62mW and the ring oscillator dissipates 0.82mW at 1.8V. The entire clock synthesizer occupies $0.25 mm^2$ of silicon.

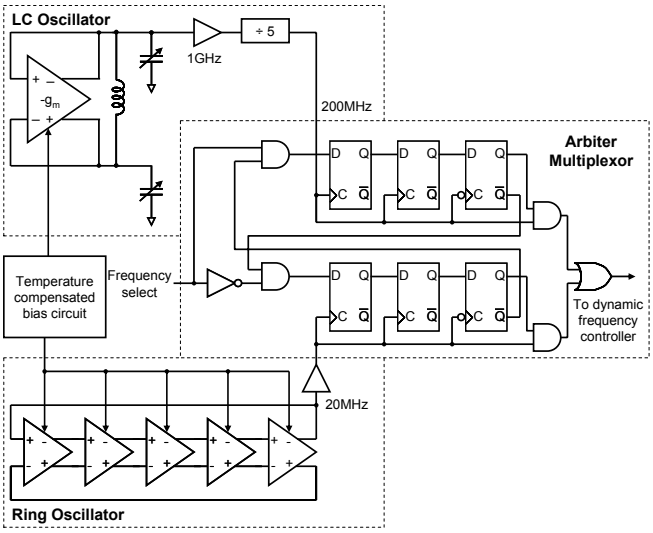


Figure 3. Self-referenced hybrid clock synthesizer.

IV. Results

The DFS circuit from Fig. 1 has been described entirely in behavioral Verilog HDL, synthesized with *Synopsys* Design Compiler, and placed-and-routed using *Cadence* Silicon Ensemble with *Artisan* standard cells. Fig. 4 shows a die micrograph of the WIMS Microsystem, with the DFS unit contained within the I/O block. No custom design, layout or transistor-level simulation was required to ensure a glitch-free clock when multiplexing between frequencies. This distinguishes our design from other DFS implementations proposed to date [2][4][5][16], and makes it ideal for ASIC design cycles constrained by time and manpower. Although it does not provide the range of frequency and phase options as do spread spectrum clock synthesizers such as the 'flying adder' [16], our design is much simpler, lower power, fully HDL synthesizable, provides ample frequency selections for many DFS applications, and is not restricted to ring oscillator based VCO/PLL clock architectures as is [16].

Figs. 5a and b show oscilloscope traces of the DFS unit switching frequencies while the MCU core is actively running. From the instant that software changes the clock multiplexer select line, there is a latency of only $n/2f_0$ + mux delay until the new frequency has propagated from the multiplexer through the synchronization chain (n=2 flip flops) and is sent to the clock tree. In the WIMS Microsystem, when the LCO is operating, f_0=100MHz so the latency is about 11ns including a 1ns multiplexer delay. When using the low power ring oscillator, f_0=10MHz, so the total latency increases to about

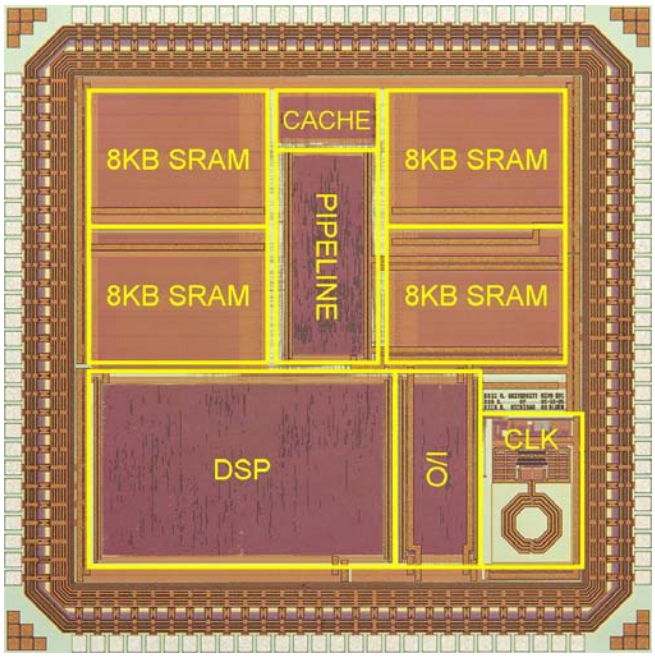


Figure 4. Die micrograph of fabricated WIMS Microsystem in *TSMC* 0.18µm MM/RF CMOS.

101ns. The power dissipated by our DFS unit is a relatively constant 480µW at 1.8V and is much lower than the 150mW reported by [16].

V. Conclusion

This work presents a low latency, HDL synthesizable dynamic frequency controller that reduces DFS latency from microseconds to nanoseconds. The latency is primarily dependent on the maximum operating frequency, f_0, so for higher frequency systems, DFS latency will be further reduced. By leveraging low-latency DFS circuits such as this, DVFS algorithms can employ very fine grained voltage and frequency scaling to optimize the energy-delay product.

Acknowledgment

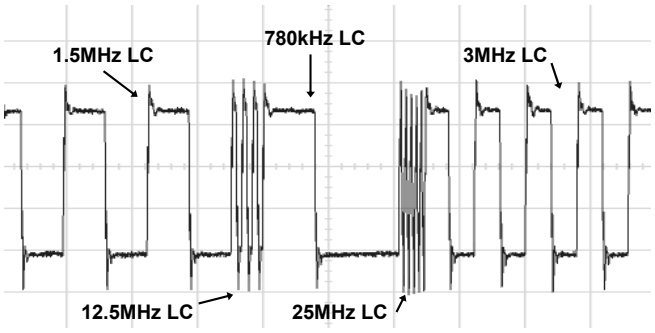
Fabrication of this work at *TSMC* was funded by the *MOSIS* Educational Program. The authors wish to thank *Artisan Components* for supplying digital cell libraries and memory generators and *Synopsys*, *Cadence*, and *Mentor Graphics* for providing CAD tools through their university programs.

References

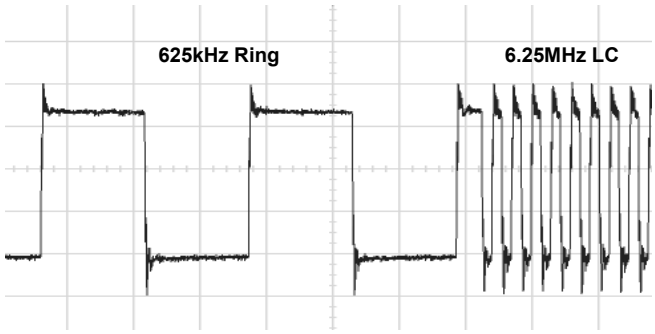
[1] S. Naffziger, B. Stackhouse, and T. Grutkowski, "The implementation of a 2-core multi-threaded Itanium-family processor," ISSCC, pp. 182-183, Feb. 2005.

[2] K. Nowka, et al., "A 32-bit PowerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling," J. Solid State Circuits, vol. 37, pp. 1441-1447, Nov. 2002.

[3] W. Dasssch, C. Lim, and G. Cai, "Design of VLSI CMOS circuits under thermal constraint," IEEE Trans. on Circuits and Systems, vol. 49, pp 589-593, Aug. 2002.



(a)



(b)

Figure 5. Oscilloscope traces showing low-latency dynamic frequency scaling of (a) the TC-LCO clock and (b) the TC-LCO and ring clock.

[4] S. Geissler, et al., "A low-power RISC microprocessor using dual PLLs in a 0.13µm SOI technology with copper interconnect and low-k BEOL dielectric," ISSCC, pp.148-149, Feb. 2002.

[5] S. Akui, et al., "Dynamic voltage and frequency management for a low-power embedded microprocessor," ISSCC, pp. 64-65, Feb. 2004.

[6] Q. Wu, P. Juang, M. Martonosi, and D. Clark, "Voltage and frequency control with adaptive reaction time in multiple-clock-domain processors," HPCA, pp. 178-189, Feb. 2005.

[7] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance trade-off based on the ratio of off-chip access to on-chip computer times," Design Automation and Test in Europe, pp. 4-9, Feb. 2004.

[8] P. Guitton-Ouhamou, H. Ben Fradj, C. Belleudy, M. Auguin, "Scheduling refinement and memory allocation for low power system," 16th Intl. Conf. on Microelectronics, pp. 240-243, Dec. 2004.

[9] E. Marsman, et al., "A DSP architecture for cochlear implants," ISCAS, May 2006.

[10] E. Marsman, et al., "A 16-bit low-power microcontroller with monolithic MEMS-LC clocking," ISCAS, pp. 624-627, May 2005.

[11] R. Senger, et al., "A 16-bit mixed-signal microsystem with integrated CMOS-MEMS clock reference," DAC, pp. 520-525, June 2003.

[12] W.J. Dally and J.W. Poulton, Digital Systems Engineering, 1st Ed., Cambridge University Press, 2000.

[13] K. Yamamoto, M. Fujishima, "4.3GHz 44mW CMOS frequency divider," ISSCC, pp. 104-105, Feb. 2004.

[14] M. McCorquodale, M. Ding, and R. Brown, "Top-down and bottom-up approaches to stable clock synthesis", Intl. Conf. on Electronic Circuits and Systems, pp. 575-578, Dec. 2003.

[15] R. Mahmud, "Techniques to make clock switching glitch free," [Online]. Available: http://www.eetimes.com/.

[16] H. Mair and L. Xiu, "An architecture of high-performance frequency and phase synthesis," J. Solid State Circuits, vol. 35, pp. 835-846, June 2000.