A DYNAMIC RECONFIGURABLE CLOCK GENERATOR

Radu M. Secareanu, David Albonesi[1]*, and Eby G. Friedman*[1]

Motorola, Inc.,
Semiconductor Products Sector, Digital DNA^{TM} Laboratories, Tempe, AZ 85284

[1]University of Rochester,
Department of Electrical and Computer Engineering, Rochester, NY 14627-0231

Abstract— **A circuit to dynamically reconfigure the clock frequency of a synchronous digital system according to the changing needs of the application is described in this paper. The circuit changes the clock frequency with a minimal time penalty and offers glitch free, reliable operation.**

I. Introduction

The 1.5 GHz barrier is presently being exceeded by leading microprocessor design houses [1–4]. The latest microprocessors incorporate tens of millions of transistors organized into highly complex computer architectures. Both the speed and number of transistors are predicted to continue to grow exponentially according to Moore's law. In these complex systems, power dissipation, thermal management, power supply design, and a variety of mechanical and electrical related issues, such as packaging, are becoming factors that limit increasing integration densities.

The complexity of microprocessor-based applications varies greatly. One application may require full use of the available hardware resources, while another application may require only a small percentage of the available hardware. One application may have to operate at maximum attainable speeds, while another application may need to operate at only a fraction of the available speed. One application may require a small high speed memory, while another application may require a large memory operating at much slower speeds. A possible solution for these different situations and constraints is an efficient reconfigurable architecture.

One approach to a reconfigurable architecture is to change the clock frequency to the specific needs of the application, leading to a significant power savings and optimal performance for the target digital system. There is, however, a major drawback to changing the clock frequency. The clock signal is the primary reference signal for a synchronous digital system. Dynamically changing (or reconfiguring) the frequency of the clock signal being distributed to each register may lead to complex problems such as glitches, race conditions, clock jitter and instability, and uncontrollable skew. These effects may lead to conditions such as loss of synchronicity, loss of data, and possibly complete failure of the synchronous system.

A *reliable* dynamically reconfigurable clock system is referred to in this paper as a clock system in which the aforementioned problems are virtually eliminated. Also, an *efficient* dynamic reconfigurable clock system is referred to as a clock system in which the time penalty for dynamically changing the clock frequency according to the needs of the digital system is minimized.

A solution for an *efficient* and *reliable* reconfigurable clock generator is presented in this paper. Alternative solutions are discussed and compared with the proposed solution in Section II. Circuit details describing the proposed solution are presented in Section III. A summary of this technique as well as selected switch level simulations are briefly reviewed in Section IV. Some conclusions are outlined in Section V.

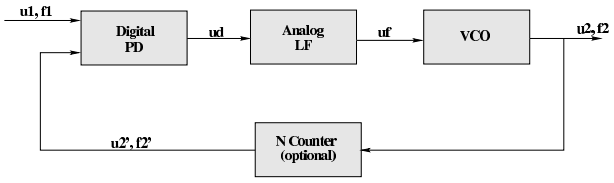

Fig. 1. A digital phase locked loop (DPLL)

II. Comparative study of dynamic clock generators

A fixed clock with high frequency stability is typically implemented by a phase locked loop (PLL) [5, 6]. A digital PLL (DPLL) is shown in Fig. 1. PD is the phase detector, LF is the loop filter, and VCO is the voltage controlled oscillator. The optional divide by N counter is used to generate an output frequency

This research was supported in part by NSF grant CCR-9811929 and by DARPA/ITO under AFRL contract F29601-00-K-0182.

which is N times the reference frequency.

The key parameters of a PLL are [7]:

- The hold range, defined as the frequency range in which a PLL can statically maintain phase tracking
- The pull-in range, defined as the frequency range within which a PLL will become locked
- The pull-out range, defined as the dynamic frequency limit for stable operation of a PLL
- The lock range, defined as the normal operating frequency range of a PLL for a fast lock.

An immediate solution to obtain a dynamic reconfigurable clock generator is to introduce a programmable divide by N counter as shown in Fig. 2. The output frequency is chosen according to the programmable scale factor N. Based on the aforementioned parameters of a PLL, note the following with respect to this simple dynamic reconfigurable clock system: a change in frequency may be a slow process (the pull-in range), the PLL may never lock if the frequency change is large (the pull-out range), and the output frequency may vary over a wide uncontrollable range while locking (all parameters). This behavior is unacceptable for a digital processing system where the output signal of the PLL must be both highly accurate and stable.

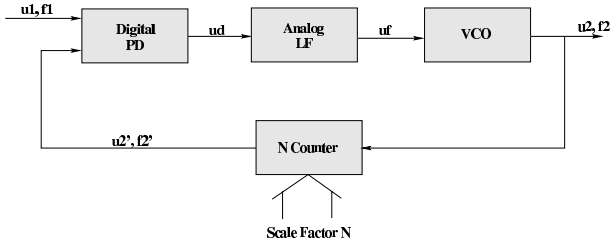

Fig. 2. A variable frequency DPLL (frequency synthesizer)

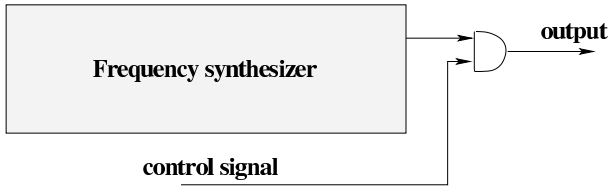

Fig. 3. Gated frequency synthesizer

Two solutions can be considered to solve these problems. One solution, shown in Fig. 3, is to simply shut off the output signal of the PLL while locking, turning the output signal back on once the locking process is over. Two drawbacks are noted:

- The off-time varies depending upon the range of the frequency change,
- The control signal is asynchronous with the clock signal, therefore glitches may propagate together with the clock signal, possibly inducing metastability within the digital system.

To reduce the off-time to zero, two PLLs, as shown in Fig. 4, can be used. If a frequency change is desired, one PLL, with the output signal disabled, locks onto the new frequency. When the locking process is over, the output is enabled and the locked frequency becomes the system clock frequency. The other PLL, with the output enabled, provides the system clock. This PLL is disabled when the first PLL locks, and is available to change the system frequency to a new clock frequency. Similar to the first approach, as the PLL changes, glitches can be generated within the system clock since the two PLL outputs operate asynchronously with respect to each other.

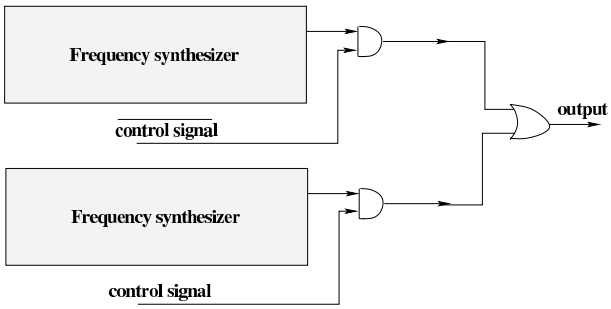

Fig. 4. Selectable gated frequency synthesizers

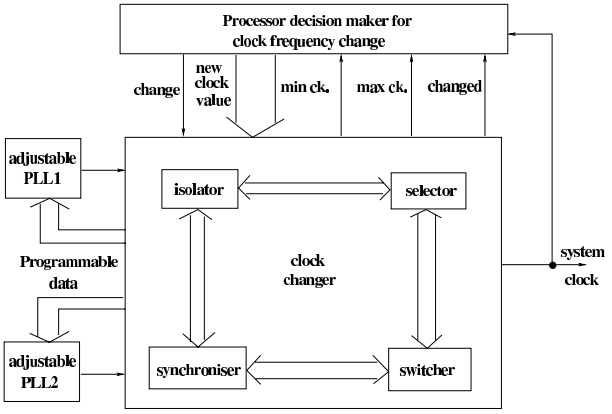

Fig. 5. Circuit solution with two adjustable PLLs

In this paper, two related circuits, shown in Figs. 5 and 6, are proposed to produce a dynamic clock. In both of these versions, note two common blocks: the *processor decision maker for clock frequency change* which, depending on some established algorithms, predicts the clock frequency requirements to achieve the optimal performance of the digital system, and the *clock changer* which *reliably* and *efficiently* changes the system clock from the original

frequency to a new clock frequency. Due to the complexity of the proposed circuits, a block level description of the circuits is provided for the purposes of this paper.

The circuit shown in Fig. 5 uses two adjustable PLLs. Similar to the circuit shown in Fig. 4, when one PLL locks to the new target frequency, the system operates at the original clock frequency. Note that the new clock frequency becomes effective after a delay equal to the lock time of the PLL (from the time the request for a frequency change is decided by the *processor decision maker* to the time when the requested frequency becomes operational). This delay may be considerable, depending upon the range of the frequency change. A solution to overcome this drawback, which, however, is hardware intensive and offers less flexibility in the selectable frequency range and frequency step variations, is shown in Fig. 6. In this circuit, several fixed PLLs operate simultaneously on predefined frequencies. The *clock changer* selects the predicted frequency, as requested by the *processor decision maker*. Since all of the PLLs are locked, the lock time is eliminated and the frequency change is immediate. The only delay between the frequency change is the processing time of the *clock changer*.

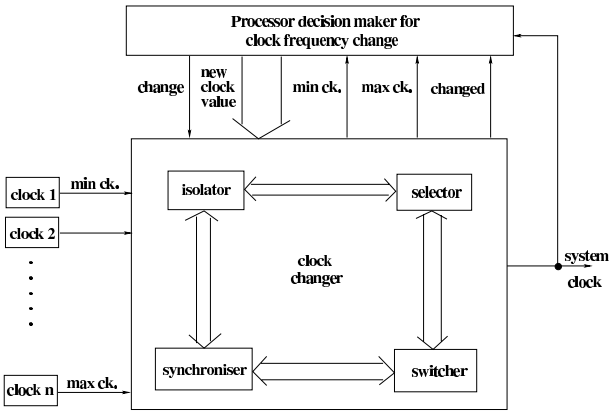

Fig. 6. Circuit solution with multiple fixed PLLs

The key hardware block of the proposed solution is the *clock changer*, which is discussed in the following section. The *processor decision maker* hardware block implements predefined decision algorithms for choosing the optimal frequency of the digital system according to the changing needs of the application.

III. Hardware overview

As mentioned previously, the key hardware block of the proposed circuit solution is the *clock changer*. Due to the circuit complexity of the *clock changer*, only a block description is addressed in this paper. The input and output signals of the *clock changer*, as well as the internal blocks and the global operation, are reviewed in this section. The algorithms implemented by the *processor decision maker* represent the subject of different work.

Note in Figs. 5 and 6 the input and output signals of the *clock changer*:

- The PLL outputs provide the available frequencies for the digital system.
- The *change* signal is generated by the *processor decision maker*. When the *change* signal is enabled, the *processor decision maker* requests from the *clock changer* a frequency change.
- The *new clock value* signal bus, generated by the *processor decision maker*, informs the *clock changer* of the requested frequency determined by the *processor decision maker*.
- The *min ck* and *max ck* signals, generated by the *clock changer*, informs the *processor decision maker* that the *clock changer* has reached the minimum or maximum available frequency limits and therefore a new request for a change of frequency in the same direction is not possible.
- The *changed* signal, generated by the *clock changer*, informs the *processor decision maker* that the *clock changer* has completed the frequency change and therefore a new request for a frequency change can be accepted by the *clock changer*.
- The *system clock* signal is the global clock output signal for the digital system.

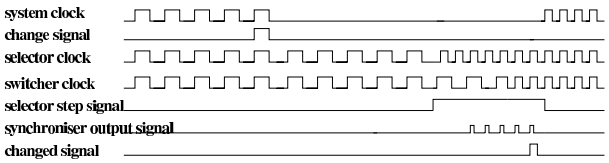

Fig. 7. A timing diagram example describing the operation of the *clock changer*

Note that the *clock changer* consists of four internal blocks: the *isolator*, the *selector*, the *synchroniser*, and the *switcher*. The internal operation of the *clock changer* can be described as follows. A timing diagram example for this description is shown in Fig. 7.

- Both the *selector* and the *switcher* receive as inputs all of the PLL outputs.
- At time zero, all of the internal blocks of the *clock changer* as well as the digital system operate at the *current system clock* frequency.
- The *change* signal is received by the *isolator*. In response to the *change* signal, the *isolator* terminates

the *current system clock* while the clock is at a zero state. Internal to the *clock changer*, the *current system clock* remains active for all of the four internal blocks.

- The frequency of the *requested clock* according to the *new clock value* signal bus is changed inside the *selector*, while the *current system clock* remains active inside the *switcher*. At the same time, a *step signal* is generated by the *selector* based on the *current system clock* signal.
- A rising edge of the *requested clock* signal and the generated *step signal* are processed by the *synchroniser*. A glitch free *output signal* with a 25% duty factor, providing the frequency of the *requested clock* signal, is generated at the output of the synchronizer. The rising edge of this *output signal* is synchronous with the rising edge of the *requested clock* signal.
- The *output signal* generated by the *synchroniser* is used to synchronously select the *requested clock* signal inside the *switcher*, thereby eliminating any glitches.
- A state machine prepares the internal hardware for the next clock change by returning all of the internal hardware registers to the initial state. This state machine also enables the system clock, synchronizing the low state of the *requested clock* to match the low state at which the system clock had been left by the *isolator*.

IV. Performance overview

The advantage of the proposed circuit is that the system is completely digital, reducing operational sensitivities to temperature, process, and noise as compared to analog implementations. The primary design sensitive part of the system is the *synchronizer*, where, in order to produce a reliable, glitch free 25% duty factor *output signal*, certain internal delays must be satisfied by the combinatorial logic circuits.

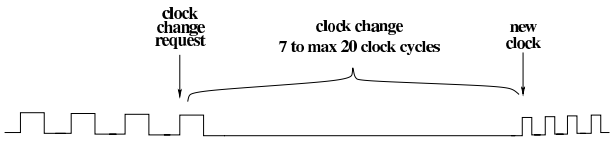

Fig. 8. The system clock signal during a dynamic clock reconfiguration

Several considerations have been accounted for in choosing an appropriate circuit simulation approach: 1) the internal circuit complexity, and 2) the goal to make the results technology independent. Switch-level simulations have been performed to confirm the circuit operation.

The results show that the critical internal delays of the *synchronizer* require a tolerance of up to $\frac{T}{4}$, where T is the period of the current clock signal. The resulting system clock waveform is as shown in Fig. 8. Assuming that all of the PLLs are locked, the time necessary to shift the system clock from the original clock frequency to the requested clock frequency varies between seven and twenty clock periods (at the original frequency); seven clock periods when only two PLLs are used, and twenty clock periods when an unlimited number of PLLs are used.

V. Conclusions

A reliable circuit to dynamically reconfigure the clock frequency of a synchronous digital system is proposed. Switch-level circuit simulations demonstrate reliable, glitch free operation. The internal circuitry is completely digital, minimizing the operational sensitivities as is typical in analog implementations (such as sensitivities to temperature, process, and noise variations). The critical element of the circuit is that specific internal delays of the combinatorial logic circuits must be controlled to within a specific level of accuracy. The proposed clock generator circuit is targeted for reconfigurable processor architectures to enhance the overall power efficiency of these systems.

References

[1] J. Mumbru, G. Panotopulos, D. Psaltis, G. Zhou, X. An, and F. Mok, "Optically Reconfigurable Processors," *Proceedings of the IEEE Southwest Symposium on Mixed-Signal Design*, pp. 22–25, February 2000.

[2] G. Lu, M.-H. Lee, H. Singh, N. Bagherzadeh, F. Kurdahi, and E. Filho, "MorphoSys: A Reconfigurable Processor Targeted to High Performance Image Application," *Proceedings of the International Symposium on Parallel and Distributed Processing*, pp. 661–669, April 1999.

[3] A. C. Cangellaris, "The Interconnect Bottleneck in Multi-Ghz Processors: New Opportunities for Hybrid Electrical/Optical Solutions," *Proceedings of the IEEE International Conference on Massively Parallel Processing*, pp. 96–103, June 1998.

[4] T. Shannon, "Alpha Processor, Inc.: It's not Your Father's Alpha Anymore! Alpha's Last Chance May Be its Best Shot," *Digital Systems Report*, Vol. 20, No. 3, pp. 7–11, Fall 1987.

[5] L. Galic, "Basic Operating Principles of the PLL and Some Problems Encountered in its Design," *Automatika*, Vol. 18, No.3–4, pp. 119–133, April 1977.

[6] G. C. Hsieh and J. C. Hung, "Phase-Locked Loop Techniques. A Survey," *IEEE Transactions on Industrial Electronics*, Vol. 43, No. 6, pp. 609–615, December 1996.

[7] R. E. Best, *Phase-Locked Loops – Design, Simulation, and Applications*. McGraw-Hill, 1997.