Glitch-free discretely programmable clock generation on chip

DOI:
10.1109/ISCAS.2005.1464968

Document status and date:
Published: 01/01/2005

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Glitch-free Discretely Programmable Clock Generation on chip

M. Meijer*, F. Pessolano^ and J. Pineda de Gyvez*
*Philips Research Labs.
^Philips Semiconductors
Eindhoven, The Netherlands

Abstract—**In this paper we describe a solution for a glitch-free discretely programmable clock generation unit (DPGC). The scheme is compatible with a GALS communication scheme in the sense that clock gating and clock pausing are possible. Besides, the proposed scheme does not require waiting for a new clock as the frequency change is seen as almost instantaneously. A prototype has been designed for a 0.13μm triple-well CMOS process technology to also study the properties of the scheme with respect to voltage scaling.**

I. INTRODUCTION

Due to shrinking technologies and increasing design sizes, clock distribution and synchronization in modern chips (being either processors or a Systems-on-a-Chip) is becoming a rather complex task. Furthermore, the clock network is also becoming a power sink in the sense that it demands a sensible percentage of the total power budget. In the past years, this trend has resulted in a new life for fully asynchronous design solutions. However, a paradigm shift from synchronous to asynchronous is unlikely to happen in the industry due to practical problems such as CAD availability as well as lack of trust from the engineering community. Therefore, a middle-path solution has arisen that tries and combine the asynchronous strategy with a more standard synchronous design: Globally-Asynchronous Locally-Synchronous or GALS [1]. In a GALS solution, a system is partitioned in islands. Each island is internally synchronous and, thus, can be realized by means of standard tooling. The composition of islands is done by means of asynchronous communication and, thus, reducing the complexity of clock distribution and synchronization.

The GALS approach is lately being further studied towards addition controllability and programmability in order to be able to merge GALS with voltage scaling techniques so as to minimize the power budget [2]. In such a scenario, the on-chip clock generator for each island is a critical component. A suitable clock generator should be programmable from one frequency to any other one, while the frequency change should be fast and free of spurious glitches. Besides, pausing and gating the clock must be possible. In this paper we describe a solution for a glitch-free discretely programmable clock generation unit (DPGC), which meet the above requirements. Besides, the proposed scheme does not require waiting for a new clock as the frequency change is seen as almost instantaneously.

II. PREVIOUS WORK

In our approach, we discuss a solution for changing the frequency anytime during the operation of the clock generator without spurious glitches on the clock signal itself. Unlike prior-art, this is a unique feature that makes the proposed solution suitable for cases in which the same core has to be operated at different frequencies (i.e. with power management techniques). In [3] Oetiker et al. describe solutions for programmable ring oscillators, where the frequency is finely tunable over a wide range. They discuss and compare most available topologies, which are based on one ring oscillator. Programmability is here considered only as a one-time event and it is not glitch-free. In [4] Taylor et al. present a solution for programmable ring oscillator with a special delay line. The purpose of programmability is here to calibrate the ring oscillator with respect to a reference frequency. Run-time frequency changes are not possible. In [5] Elboim et al. replace the multiple local oscillator with a central PLL and local frequency multipliers. In this case, changing the local frequency involve programming the local multiplier. The operation is to be performed by explicitly gating the clock as for standard multi-output PLL solutions.

III. PRINCIPLES OF THE DPCG

We have devised a micro-architecture for a glitch-free discretely programmable clock generator (DPCG) based on the usage of two identical mutual-exclusive ring oscillators (see Figure 1). The DPCG is based on the simple idea of using one oscillator to produce the clock at the current operating frequency, while the other is activated to generate the clock at the next frequency. When the new clock is stable, the two ring oscillators are synchronized on the low phase of the clock. The new clock is connected to the output, while the old one is stopped. Therefore, only one oscillator is actually oscillating at any given time (except from when the frequency is being changed). The interface to the DPCG consists in two four-phase channels for frequency programming (*Req_f, Ack_f, Data_f*) and clock gating (*Req_g*, *Ack_g*) as depicted in Figure 1. The final scheme includes a test and reset interface that is not made explicit here. Each channel is composed by a request (*Req_f*, *Req_g*) and an acknowledgment signal (*Ack_f*, *Ack_g*). The frequency channel also includes a multi-bit data signal (*Data_f*) to specify the desired output clock frequency. The DPCG functions as follows. When the clock is to be gated, a request is raised on the gating channel (*Req_g*). At this point the switch control gates the currently running internal oscillator and allows the raising of the acknowledge signal (*Ack_g*). The clock gating is removed only after the acknowledge signal is released (not the request signal). When the clock frequency is to be changed, the code representing the new frequency is put on the data signal (*Data_f*). Once this signal is stable a request is raised on the frequency channel (*Req_f*). At this point, the event controller takes care that the sequence of action is correct and as follows. At first, the content of signal *Data_f* is stored (in the MUX flip-flop) and routed to the currently non-oscillating programmable ring. This also results in the raising of the acknowledge signal (*Ack_f*) followed by the retirement of the request (*Req_f*). The non-oscillating ring is then made oscillating at the new frequency and, within one clock cycle, synchronized on the low pulse with the other ring oscillator. At this point, the acknowledgement (*Ack_f*) is retired and the newly oscillating ring is transmitted to the output clock, while the other is stopped.

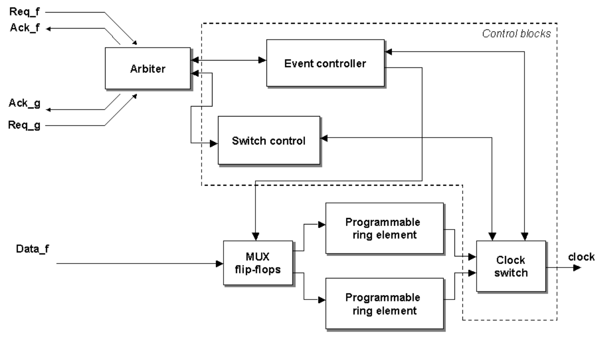

Figure 1: Micro-architecture of the DPCG unit

It is important to notice that even if the clock is transmitted after the acknowledgment is retired, the event controller will ensure that any other request is pending until the clock has not been transmitted to the output. Since there is not relation between the two requests, it might happen that they happen concurrently (or very near in time) thus potentially generating a conflict in determining the order in which the requests should be serviced. This situation is solved by means of a standard arbiter module as normally done in asynchronous systems (see Figure 1).

IV. Design of the Control Block

The correct sequence of actions as described in the previous section is implemented by the control blocks (see Figure 1): event controller, switch control and clock switch. In this section we will give a description of these blocks together with a first logic implementation. A more detailed view on the control blocks is given in Figure 2. The event controller is a finite state machine. The switch control is nothing else than a multiplexer for handshake signals. The clock switch is basically composed by a multiplexer and clock-gating modules use as closing head for each of the ring oscillators (ring head - Figure 2). The event controller essentially takes care that the ring oscillator are synchronized on the clock low phase when frequency is being programmed and that only one ring oscillator is running at a given time. We can formally describe its operation mode with the STG (Signal Transition Graph) in Figure 3. We can see that handshaking on channels (*gate0*, *gated0*) and (*gate1*, *gated1*) is used to synchronize the two ring oscillators. Signal *f_mux* is instead used to select the ring oscillator, which will supply the next clock frequency.

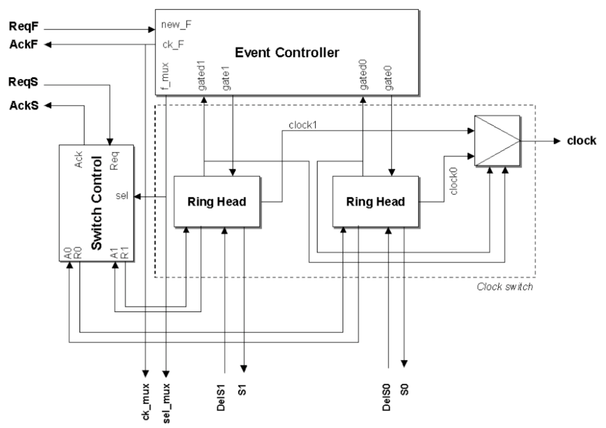

Figure 2: Zooming into the control blocks of the DPCG unit

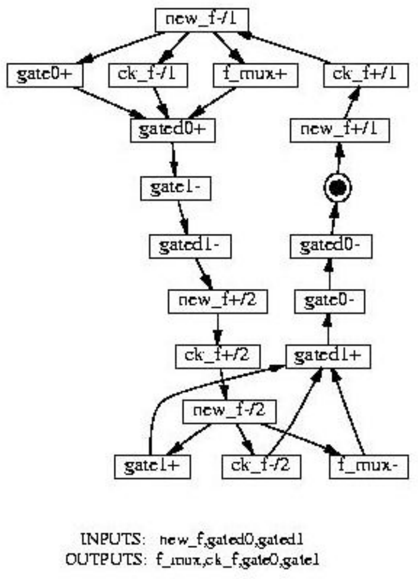

Figure 3: STG description of the Event Control module

The logic implementation for the event controller has been obtained by means of the Petrify tool [6]. The transformation of the obtained logic equations into fully testable circuits has been trivial and done by hand (Figure 4).

In doing so, we have also included standard test shells for every Muller C-gate. The switch control is basically a multiplexer for four-phase handshake signals. Its function is to re-route the gating request to the currently oscillating ring and re-route back the acknowledge signal. It has been designed fully by hand and without support of an STG description so as to minimize its delay (Figure 5). The unit used to perform the clock switching is composed of two ring heads units and one standard multiplexer. Each ring head is used to close the ring oscillator and to implement the gating with handshaking. The circuit for the ring head unit is given in Figure 6. The signal *Si* is sent to the programmable delay element (that constitute the ring oscillator). This delay element provides the ring head with a delayed form of Si, which is signal *DelSi*.

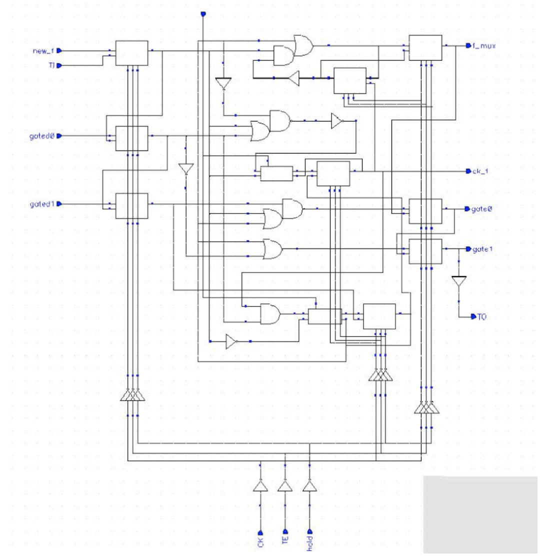

Figure 4: Implementation of the event controller.

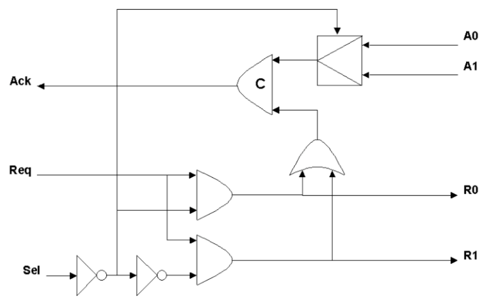

Fig. 1:

Figure 5: Details of the Switch Control module

The NAND gate on *Si* is used in order to stop the ring oscillator, when it is not used to generate the output clock. We can identify two cascaded standard latch+AND gating structures, which are used for gating the clock and synchronization on the low phase of the clock. The *gated* and *A* signals are used to signal when the gating has taken place. There is a timing constraint on these signals, as they have to signal the gating of the clock after the clock is really gated. Obviously the inverter gate is not enough of a delay; however, systematic delays in the event controller make sure that this timing constraint are always met (as long as the latch and the AND gate are kept close to each other).

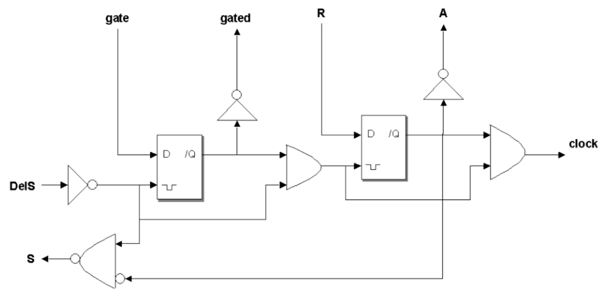

Figure 6: Details of the Ring Head module

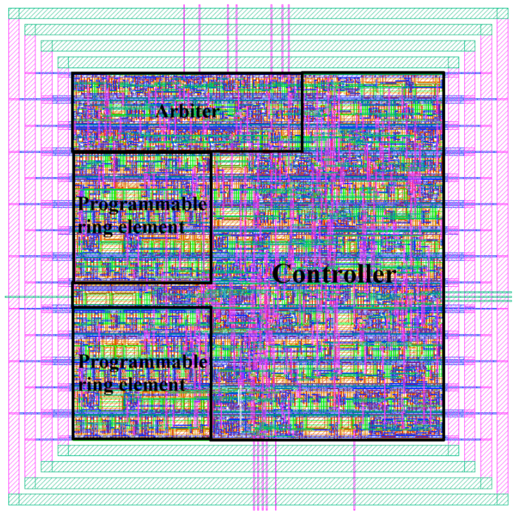

Figure 7: Layout of the complete DPCG unit

V. Realization of a complete DPCG

The physical implementation of the DPCG has been completed in a 0.13μm triple-well CMOS technology with a nominal supply voltage of 1.2 volts. A semi-custom design methodology has been applied, implementing the DPCG unit with standard cells only. Standard cells of a mutual-exclusive and Muller-C element have been designed manually. The DPCG supports adaptive control of power supply voltage, and independent bias control of N-well and P-well respectively. Such adaptive control techniques enable frequency tuning, compensation of inter-die process parameter spread and temperature effects. The design was laid out using a commercial place-and-route tool and a row utilization of 0.8. The DPCG consumes about 70x70 μm^2 excluding the power rings, and 100x100 μm^2 including the power rings. The layout of the complete DPCG is shown in Figure 7. The programmable ring elements are built up out of AND gates, multiplexers, and an inverting ring head. The AND gates are concatenated and serve as delay elements. Multiplexers support the selection of three discrete frequencies by programming the length of the ring oscillator to contain 7, 15, or 30 AND gates respectively. The ring head closes the ring oscillator chain. Each programmable ring element consumes a chip area of 25x25 μm^2. Both programmable ring elements consume about 25 percent of the DPCG core area, while the rest is consumed by the hardware required for the clock control, handshaking, and test support.

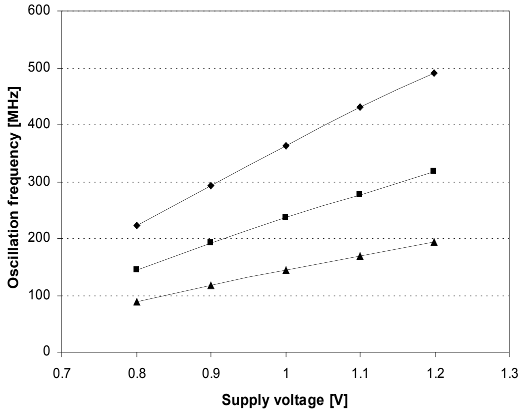

Figure 8: Oscillation frequency versus supply voltage for the programmable ring element.

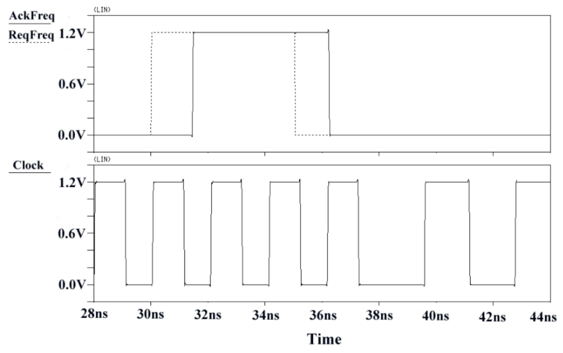

Figure 9: Example simulation trace of frequency change

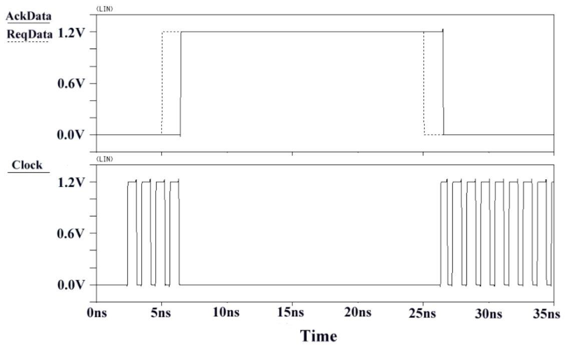

Figure 10: Example simulation trace of clock gating

VI. Circuit simulation results

The layout of the complete DPCG has been extracted by a commercial layout extraction tool, and simulated using an in-house circuit simulator. The nominal supply voltage of the 0.13μm CMOS technology equals 1.2 volts. Simulations have been performed for a temperature of 25 degrees Celsius, and nominal bias of N-well and P-well respectively. At nominal process conditions and at 1.2 volts supply, the implemented DPCG is able to generate a discrete oscillation frequency of 491MHz, 317MHz, and 194MHz respectively depending on the programmed length of the ring oscillator.
Figure 8 shows the oscillation frequency as function of supply voltage for three programmed chain length. From this figure, one can observe that the frequency is reduced by a factor of 2.2 when the supply voltage is lowered to 0.8 volts. On one hand, one can do a coarse grained selection of oscillation frequency by programming the ring oscillator chain length. On the other hand one can do a fine-grained tuning of this oscillation frequency by properly adjusting the power supply voltage value. Next to that, one could also use a proper well bias for the fine-grained frequency control.
Figure 9 shows an example simulation trace of the implemented DPCG in the frequency-programming mode. As it can be noted, the trace follows the mode of operation as described earlier in section 2. No glitch at the output clock is produced when changing the frequency. The change in frequency is only seen as an additional delay on the low phase of the output clock. An example simulation trace in case of clock gating is shown in Figure 10. It can be noticed that the ring oscillator is gated after the gating request (*Req_g*), which is confirmed by raising the *Ack_g* signal. By releasing the *Req_g* signal, the clock gating is removed after the *Ack_g* signal is released.

VII. Conclusions

In this paper we have described a solution for building a programmable on-chip clock generator, which allows discrete changes in the operating frequency without spurious glitches. A prototype design has been completed for a 0.13 μm triple-well CMOS process technology. Circuit simulations confirm that this scheme delivers fast frequency transitions without spurious glitches. Besides, the clock generator showed to correctly work with supply voltage scaling.

References

[1] D. Chapiro, “Globally-Asynchronous Locally-Synchronous Systems”, Ph. D. Thesis, Stanford University, 1984

[2] Iyer, D. Marculescu, “Power and performance evaluation of globally asynchronous locally synchronous processors”, Proc, of the 29th Intl. Symposium on Computer architecture, ISCA2002

[3] S. Oetiker et al, “High Resolution Clock Generators for Globally-Asynchronous Locally-Synchronous Designs", Handouts of the ACiD 2002 workshop, Munich, Germany, Jan. 2002.

[4] G.S. Taylor, S.W. Moore, P. Robinson, “An on-chip dynamically recalibrated delay line for embedded self-timed systems”, Proc. of ASYNC 2000, Eilat, Israel, Apr. 2000

[5] Y. Elboim, R. Ginosar and A. Kolodny, “A Clock Tuning Circuit for System on Chip,” IEEE Transactions on VLSI, 11(4), pp. 616 –626, 2003.

[6] www.lsi.upc.es/~jordic/petrify/petrify.htm