A Symmetric Differential Clock Generator for Bit-Serial Hardware

Mitchell J. Myjak and José G. Delgado-Frias
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA, USA

Abstract

Bit-serial architectures require less area but more sophisticated clocking mechanisms than their parallel counterparts. This paper presents a CMOS circuit that generates high-frequency clock waveforms for bit-serial hardware. Triggered by a master clock input, the circuit outputs a fixed number of pulses followed by a completion signal. The design uses two coupled ring oscillators to produce differential outputs with aligned positive and negative transitions. This feature allows the circuit to drive transmission gates as well as shift registers with complementary enable inputs.

Layout simulations in 180-nm CMOS demonstrate that the clock generator can run at 2 GHz. The circuit thus can drive a bit-serial module with a 9-cycle computation phase at 200 MHz. The power consumption is 1.66 mW in this case. The small size of the layout permits the clock generator to be replicated locally for each bit-serial module.

Keywords: Bit-serial computation, differential clock generator, ring oscillator, symmetric waveforms.

1. Introduction

Bit-serial architectures trade off physical size for time complexity by representing data words as a stream of bits on a single line. This approach has been popular for implementing multiplication and other arithmetic functions where area takes priority over speed. For example, a bit-serial, n-bit multiplier typically computes the product in $2n$ cycles using a linear chain of processing elements [1]. In contrast, a parallel implementation such as a carry-save multiplier operates as a combinational logic block but uses an $n \times n$ array of processing elements. Bit-serial architectures appear frequently in signal processing applications that require long word sizes, such as cryptography [2].

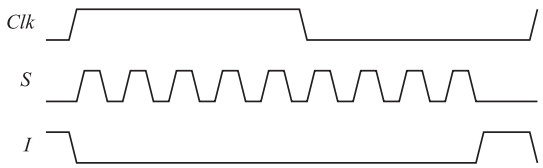

Figure 1. Clocks required by generic bit-serial module.

Another domain in which bit-serial architectures have generated considerable interest is reconfigurable hardware. Representing data words with one line rather than a group of lines conserves the routing resources available on a field programmable gate array (FPGA). One example in the literature describes such an implementation of a modular multiplier for cryptographic systems [3]. Other researchers have proposed new FPGA architectures that directly support bit-serial computations [4]–[6]. In these designs, the memory units within the basic reconfigurable cells can be configured to implement processing elements for the computation itself or a shift register for the data lines.

In general, a bit-serial module requires the clock waveforms shown in Figure 1. The system clock Clk is the common clock signal shared by all components, both serial and parallel. The serial clock S determines the rate at which bits appear on the serial data lines and are processed by the module. During each cycle of the system clock, S contains a fixed number of pulses, corresponding to the number of bits in the data word. After the last pulse, an initialization signal I is asserted to prepare the module for the next data word. Clearly, S must run at high frequency for the module to achieve reasonable performance.

Figure 2 illustrates a circuit proposed by Nilsson, Torkelson, Vesterbacka, and Wanhammar

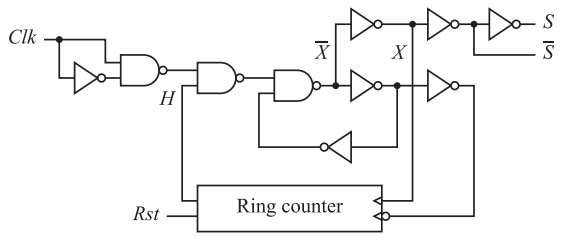

Figure 2. Clock generator proposed by Nilsson et al [7].

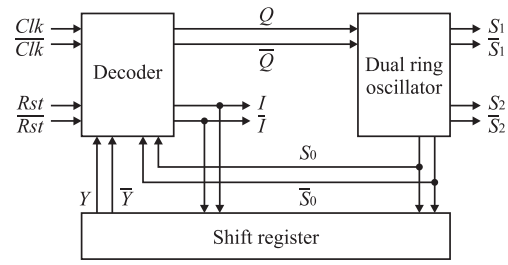

Figure 3. Block diagram of differential clock generator.

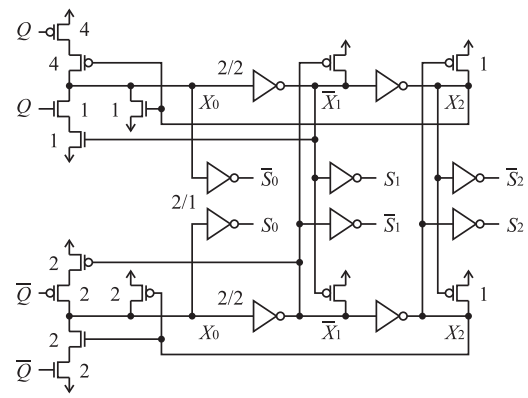

Figure 4. Dual ring oscillator.

(NTVW) to control a bit-serial module [7]. A ring oscillator generates the serial clock S. The output of the oscillator controls a ring counter, which is reset to $1000\ldots$ on power-on. After a given number of pulses, the "hold" output H of the ring counter activates and stops the oscillator. The rising edge of Clk creates a negative pulse that enables the ring oscillator again, repeating the cycle. This design is quite straightforward and functions properly at high frequencies. However, the serial clocks S and $\overline{S}$ generated by the oscillator are not aligned. Since the clocks may drive data latches and shift registers in the bit-serial module, having a symmetric output would allow the module to use latches with complementary enable inputs.

This paper presents a clock generator for bit-serial hardware that overcomes the limitations of the existing approach. In particular, the circuit produces symmetric clocks suitable for controlling a high-performance datapath. The circuit can also tolerate a wide range of output frequencies. Section 2 of this paper describes the design of the clock generator. Section 3 gives a series of layout simulations that illustrate the operation of the design. Section 4 analyzes the area, speed, and power consumption of the circuit, and compares the operating characteristics to the NTVW design. Finally, Section 5 summarizes the results and comments on potential applications.

2. Circuit design

Figure 3 depicts a functional block diagram of the differential clock generator. The design contains a dual ring oscillator to generate the serial clocks, a shift register to stop the oscillator after the required number of cycles, and a decoder to compute the initialization signals. Since most signals in the design are differential, assume from this point forward that a statement about a signal x implies an analogous statement about the complement $\overline{x}$, and vice versa.

The schematic of the dual ring oscillator appears in Figure 4. The circuit provides three differential outputs: S_0 for the shift register, S_1 for the main bit-serial module, and S_2 for any parts of the bit-serial module that require delayed clocks. Due to variations in parasitic capacitance, the two ring oscillators will likely have slightly different natural frequencies. However, the weak cross-coupled p-transistors ensure that the internal nodes remain in antiphase with each other. Although only one pair of p-transistors is necessary for this purpose, two pairs are used for extra assurance. The numbers in the figure denote the transistor sizes, in units of the minimum width.

As shown in Figure 5, the shift register contains a chain of dynamic latches. The number of latches determines the number of pulses generated by the oscillator. All control inputs are buffered so that the register can be of any length. The initialization signal I connects to the data input of the register. When I is high, all the latches reset asynchronously. When I is low, the internal data is shifted through the register toward the output Y.

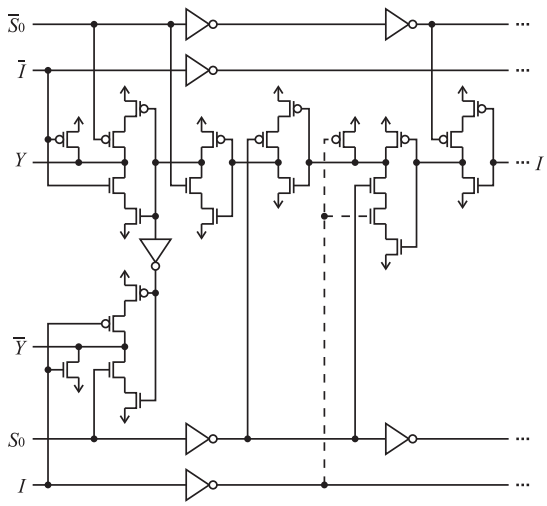

Figure 5. Shift register used to stop oscillator.

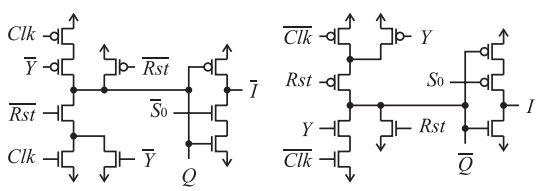

Figure 6. Decoder for initialization.

This signal initializes to high, transitions to low during the serial processing stage, and returns to high when the processing is complete.

The shift register uses several techniques to increase performance. The serial clock S_0 connects directly to the latches driving Y so that the oscillator can be stopped quickly. For the same reason, the register is designed to assert Y at the rising edge of the last pulse rather than the falling edge. Finally, notice that the shift register initializes by propagating the reset data through each latch from the input end to the output end. To expedite this process, some of the latches have an extra reset input.

The last component of the clock generator, the decoder, appears in Figure 6. This component reads the output Y of the shift register and generates the "quench" signal Q to stop the oscillator. The decoder is also responsible for producing the initialization signal I.

3. Operation

This section explains the operation of the design with the help of some simulated waveforms. The simulations were performed on a transistor

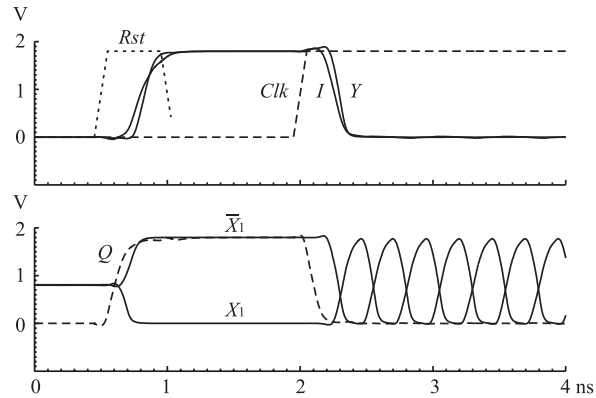

Figure 7. Reset and start of clock waveforms.

layout in 180-nm CMOS with all parasitic capacitances extracted. The chosen transistor sizes permitted the oscillators to run near the maximum frequency that maintained rail-to-rail swing. The shift register was designed to stop the serial clock after nine pulses, a common scenario for 8-bit computations.

Figure 7 depicts the initial state of the clock generator. To reset the circuit at power-on, the system asserts Rst and leaves Clk low. The compound gates on the left of Figure 4 assert Q to disable the oscillators. Hence, node X_1 is held low. The initialization signal I is also asserted, resetting the contents of the shift register. However, the output Y of the shift register is set to high. When the system releases Rst, Q remains high so the circuit remains in initialization state.

The rising edge of Clk causes the decoder to deactivate Q and I, turning on the two oscillators. In the shift register, Y falls low as the first internal values are shifted to the output. The cross-coupled p-transistors in the oscillators align X_1 and $\overline{X_1}$ very well, even at this high frequency.

Figure 8 illustrates how the clock generator returns to initialization state after producing nine pulses. Assume that Clk has transitioned back to low. When S_0 rises for the last time, the shift register sets Y to high. This event pulls Q high as well, but the circuit does not enter into initialization state yet. Observe in Figure 4 that the p-transistor path to node X_0 is cut off, preventing any more positive transitions, but the n-transistor path cannot truncate the output pulse prematurely since $\overline{X_1}$ is low. After a short time, X_2 rises and pulls down X_0. Then $\overline{X_1}$ rises and

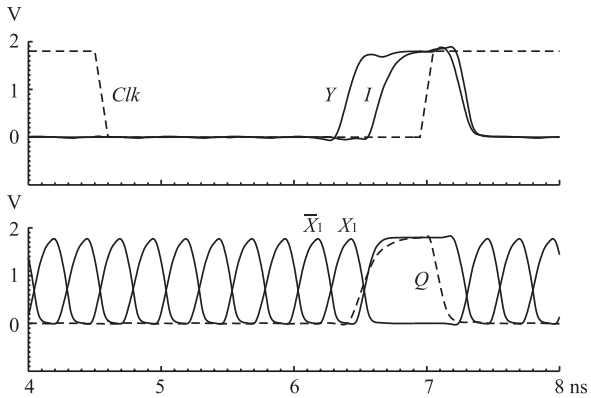

Figure 8. End of clock waveforms and initialization.

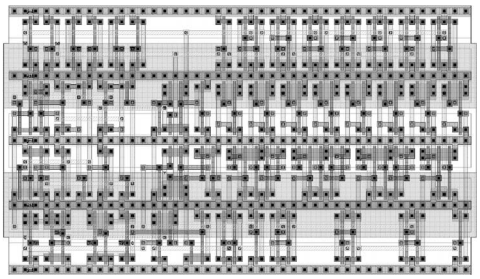

Figure 9. Layout of clock generator.

holds N_0 low. The circuit asserts I only after S_0 becomes low to prevent conflicts in the shift registers. The initialization state lasts until Clk rises again, repeating the process.

4. Analysis

Figure 9 shows the transistor layout of the clock generator. The layout occupies an area of 34.5 μm by 20.0 μm in 180-nm CMOS technology. The size is sufficiently small for the system to include a separate clock generator for each bit-serial module, if desired. Localizing the clocking circuitry in this manner avoids the problems inherent to distributing a high-frequency clock to multiple modules in the system.

Figure 10 depicts the overall results of the layout simulations. The serial clock contains nine pulses at a frequency of 2 GHz. The system clock runs at 200 MHz, providing some leeway for parameter variations. To increase the oscillation frequency, the n-transistors within the ring oscillators could be enlarged. This change would augment the driving capability of the pull-down stages compared to the cross-coupled p-transistors. Other simulations have suggested that

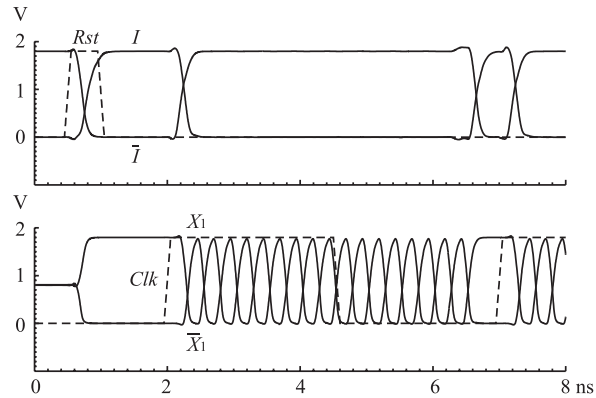

Figure 10. Layout simulation showing 2-GHz operation.

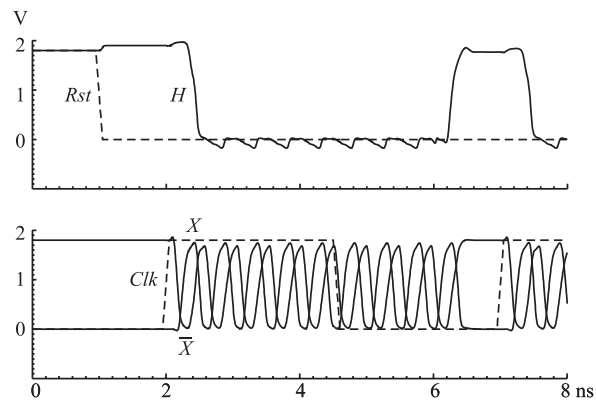

Figure 11. Simulation of NTVW clock generator [7].

the design performs correctly up to 2.5 GHz, but at this point the internal signals no longer swing from rail to rail. To decrease the oscillation frequency, the transistor widths could be decreased, or more stages added to the oscillators.

When scaled to the same 180-nm technology, the NTVW clock generator [7] can run above 2 GHz but does not produce symmetric outputs. Figure 11 contains simulated waveforms for this design at 2 GHz. The serial clocks S and $\overline{S}$ are misaligned by approximately one-sixth of a cycle. Using these outputs to drive data latches might lead to race conditions unless the bit-serial module used a circuit style such as NORA-CMOS [8]. Having a symmetric differential clock allows the module to use other elements, such as transmission gates and dynamic circuitry, for maximum performance.

The NTVW design also has a problem generating the serial clock at lower frequencies. In Figure 12, the frequency of the ring oscillator has been reduced to about 1 GHz by adding

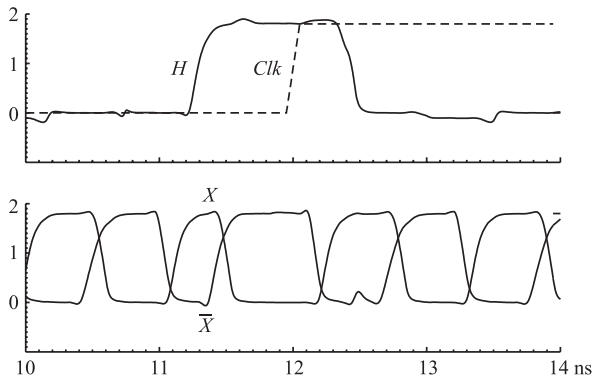

Figure 12. NTVW design cuts off pulse at lower frequencies.

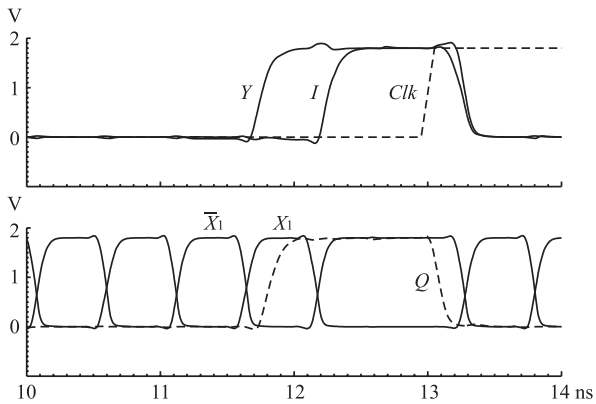

Figure 13. Proposed design can operate at lower frequencies.

more inverter stages. The output H of the ring counter rises at the beginning of the last pulse. This event reinitializes the circuit, cutting off the pulse prematurely. One solution to this problem would be to modify the ring counter so that it asserted H at the falling edge of the last pulse. However, this change would prevent the design from operating at high frequencies, due to the propagation delay between the ring counter and the oscillator.

Figure 13 demonstrates that the proposed design supports lower frequencies with no changes. As described earlier, stopping the oscillator requires a two-step process whereby the serial clock must complete the last pulse before the circuit initializes. This property also increases the resilience of the clock generator to parasitic capacitances and parameter variations.

The average power consumption of the clock generator is 1.66 mW when running at 2 GHz. Although the NTVW design is not fully delineated in [7], circuit simulations suggest that its power consumption is around 1.3 mW for the same situation. This result does not include parasitic interconnect capacitances, so the actual power consumption would be closer to the proposed design. Notice that both designs save power by turning off the ring oscillator when not in use.

5. Conclusion

This paper has presented a novel clock generator for bit-serial hardware. The circuit contains two ring oscillators interlocked with p-transistors to create a differential serial clock at high frequency. A shift register of arbitrary length counts the number of pulses on the outputs and asserts a signal when the last pulse has arrived. The circuit then turns off the ring oscillators in a two-step process and outputs a completion signal. This signal reinitializes the shift register as well as the bit-serial module. Layout simulations demonstrate that the design can produce a serial clock of 2 GHz with full-rail swing.

Future work could consider several improvements to the design. Currently, the system clock must transition from high to low while the serial clock is still running; otherwise, the oscillator does not stop correctly. Modifying the logic to correct this problem would be straightforward. Another potential limitation of the design is that it has no mechanism to change the oscillation frequency at runtime. For bit-serial applications, this capability usually is not essential. However, a tuning element such as a current-starved inverter could be added into the oscillation loop. The special initialization circuitry already permits the design to operate over a wide frequency range.

Finally, one application that could potentially benefit from this high-performance clock generator is reconfigurable hardware. The authors are currently developing a bit-serial reconfigurable architecture that uses this clock generator in each cell. Each cell performs arithmetic or memory functions on 4-bit or 8-bit data words in a bit-serial fashion. The calculated results are collected into a register and then transferred to the next cell as a bit-serial string.

Acknowledgment

M. Myjak is supported by a graduate fellowship from the U.S. Department of Homeland Security

(DHS). The DHS Scholarship and Fellowship Program is administered by the Oak Ridge Institute for Science and Education (ORISE) for DHS through an interagency agreement with the U.S. Department of Energy (DOE). ORISE is managed by Oak Ridge Associated Universities under DOE contract number DE-AC05-00OR22750. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORISE.

References

[1] P.T. Balsara and D.T. Harper III, "Understanding VLSI bit serial multipliers," *IEEE Trans. Education*, vol. 39, iss. 1, pp. 19–28, Feb. 1996.

[2] A.K. Daneshbeh, "Area efficient high speed elliptic curve cryptoprocessor for random curves," in *Proc. Int. Conf. on Information Technology: Coding and Computing*, vol. 2, pp. 588–592, Apr. 2004.

[3] W.P. Marnane, "Optimised bit serial modular multiplier for implementation on field programmable gate arrays," *Electronics Letters*, vol. 34, iss. 8, pp. 738–739, Apr. 1998.

[4] A. Ohta, T. Isshiki, and H. Kunieda, "New FPGA architecture for bit-serial pipeline datapath," in *Proc. IEEE Symp. on FPGAs for Custom Computing Machines*, Napa Valley, CA, pp. 58–67, Apr. 1998.

[5] T. Isshiki et al, "High density bit-serial FPGA with LUT embedding shift register function," in *Proc. 2002 Asia-Pacific Conf. on Circuits and Systems*, vol. 1, pp. 475–480, Oct. 2002.

[6] N. Ohsawa, M. Hariyama, and M. Kameyama, "Architecture of a field-programmable VLSI processor using memory-based cells," in *Proc. 41st SICE Annual Conf.*, vol. 3, pp. 1849–1852, Aug. 2002.

[7] P. Nilsson, M. Torkelson, M. Vesterbacka, and L. Wanhammar, "CMOS on-chip clock for digital signal processors," *Electronics Letters*, vol. 29, iss. 8, pp. 669–670, Apr. 1993.

[8] J. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 2nd ed., Upper Saddle River, NJ: Pearson Education, 2003, pp. 361–363.