LOCAL GENERATION OF THE FALLING CLOCK EDGE FOR SILICON RESOURCE SHARING

Glen Morris and Gerard M Blair

A simple mechanism for sharing combinatorial logic without increasing the global clock frequency is described, together with a transitor-level circuit for controlling the falling edge of a locally generated clock.

1. Introduction

With the increased speeds offered by sub-micron technologies, there is often time to spare when performing even complex functions with purely combinatorial (i.e. non-pipelined) logic. This is especially true when the input data rate is fixed, perhaps by a communication standard. In some cases, silicon resources can be shared although this normally implies that the system must be run at a higher clock rate than the data to accommodate the multiple use of hardware within a single sample period.

A simple approach to doubling throughput is to use both edges of the clock. However, this imposes the restriction that both phases must independently be longer than the critical path of the combinatorial logic, which adds to the clock generation/distribution problem. In this letter we introduce a simple circuit to produce a local clock with the same frequency as the global clock but with a locally controlled falling edge.

2. General architecture

Figure 1 shows the general scheme. The locally-clocked resource-sharing region is preceded and followed by edge-triggered registers controlled by the global clock. The local clock controls both a multiplexor which selects between two sets of inputs, and transparent latches which retain the output of the first set of inputs after the clock changes phase. Thus the results calculated from data on both phases of the local clock are available when the global clock latches them into the following registers.

The methodology may be seen as a variant of pausible/stoppable clocks[1, 2] and the asynchronous controller design for *burst-mode* specification for asynchronous controllers reported by Nowick *et al*[3]. In the former work, a local clock is generated in response to an event or control signal, performs a task requiring several (local) clock cycles and then stops/pauses until its output is consumed and/or a new control signal arrives. In the latter, changes on a specified set of data inputs cause a single clock pulse which updates the state registers.

3. Local clock generation circuit

The new circuit is shown in figure 2 and may be seen as a cross between an asymmetric Muller-C gate[4] and an edge-triggered Dtype. The Δ-delay is matched to the critical path of the combinatorial logic (specifically that of the positive clock phase) - this could also be achieved using completion detection. The smaller-drawn inverters are weak feed-back inverters: both render their outputs static, and the right feed-back inverter also compensates for the threshold voltage drop across the n-type pass-transistors.

The operation relies upon the assumption that the Δ-delay is less that half the clock period (and, more precisely, that the sum of the low-phase and high-phase delays is less that the clock period). This implies that the reset signal is LOW at the end of the low-phase of the global clock; the operation thence is as follows:

- with *clk* and *reset* both LOW, *x* is HIGH and the node at *y* retains its value (which we will show is LOW)

- when *clk* rises, the value at *x* is latched HIGH and its inverse (LOW) is established on *y* through the pass-transistor controlled by *clk* - this raises the local clock signal

One of two events can occur next: either (A) the *clk* signal falls, or (B) the *reset* rises. In case (A):

- when *clk* falls, node *x* charges to HIGH and node *y* becomes latched (remaining LOW); by the delay assumptions, *clk* will not change again until after *reset* has both risen and fallen
- when *reset* rises, node *x* is pulled LOW and *y* becomes HIGH through the pass-transistor controlled by *reset* - the local clock becomes LOW
- when *reset* falls, node *x* becomes HIGH and node *y* becomes latched (remaining HIGH)

In case (B):

- when *reset* rises, node *x* is pulled LOW and *y* becomes HIGH through both pass-transistors - the local clock becomes LOW

One of two events can occur next: either (B1) the *clk* signal falls or (B2) the *reset* falls. In case (B1):

- when *clk* falls, no change occurs since the HIGH *reset* signal maintains the same values on *x* and *y*
- when *reset* falls, node *x* becomes HIGH and node *y* becomes latched (remaining HIGH)

In case (B2):

- when *reset* falls, node *x* becomes latched (remaining LOW) and node *y* is unchanged (HIGH)
- when *clk* falls, node *x* is charged HIGH and node *y* becomes latched (remaining HIGH)

In summary, the local clock rises when *clk* rises, and falls when *reset* rises. Figure 3 illustrates this for various values of the *clk* duty cycle.

4. Conclusions

We have demonstrated a technique for sharing large combinatorial networks within an otherwise purely synchronous environment, without increasing the system clock frequency. A general technique for merging a locally-clocked region within an otherwise purely synchronous environment has been demonstrated together with a novel transistor level circuit for generating the falling clock edge.

References

1. C Traver, “A testable model for stoppable clock ASICs”, *Proc IEEE Int. ASIC Conference and Exhibit*, pp. 6-3.1-6-6.5, 1991.
2. Kenneth Y. Yun and Ryan P. Donohue, “Pausible clocking: A first step toward heterogeneous systems”, *Proc. Int. Conf. on Computer Design*, pp. 118-123, 1996.
3. Steven M. Nowick, Kenneth Y. Yun, and David L. Dill, “Practical asynchronous controller design”, *Proc. Int. Conf on Computer Design*, pp. 341-345, IEEE, 1992.
4. S B Furber and P Day, “Four-phase micropipeline latch control circuits”, *IEEE Trans VLSI Systems*, vol. 4, no. 2, pp. 247-253, June 1996.

Glen Morris is a Teaching Company Associate with Wolfson Microelectronics and Gerard M Blair is a Senior Lecturer at the Department of Electrical Engineering in the University of Edinburgh

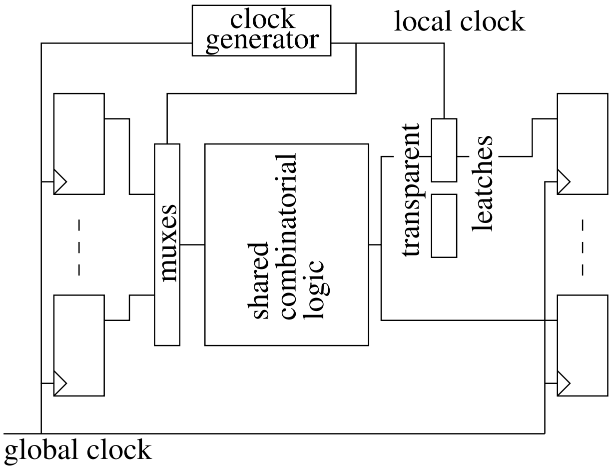

Fig. 1: general interface between local-clock and global-clock regions

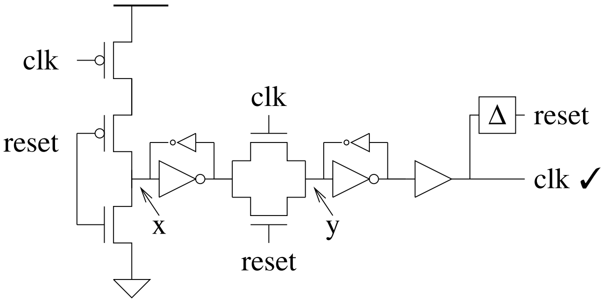

Fig. 2: local clock generator

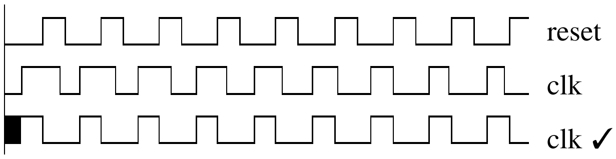

Fig. 3: local clock waveform