Beyond Carrying Coal To Newcastle: Dual Citizen Circuits

Marly Roncken[1], Chris Cowan[1], Ben Massey[1], Swetha Mettala Gilla[1], Hoon Park[1], Robert Daasch[1], Anping He[2], Yong Hei[3], Warren Hunt Jr.[4], Xiaoyu Song[1], and Ivan Sutherland[1]

[1] Portland State University, Portland, Oregon, USA
[2] School of Information Science & Engineering, Lanzhou University, Lanzhou, China
[3] Institute of Microelectronics Chinese Academy of Sciences, Beijing, China
[4] The University of Texas at Austin, Austin, Texas, USA

Abstract. This paper makes self-timed circuits dual citizens by providing a clocked mode of operation in addition to their self-timed mode. The clocked or synchronous mode of operation re-uses the self-timed fabric and protocols, and thereby — beneficially — inherits the elasticity of the self-timed or asynchronous mode of operation. In exchange, clocked circuit operations can build confidence in self-timed circuit operations or replace aging or erratic self-timed circuit operations that need more time to finish. Once confidence is gained, the self-timed mode of operation can serve as a turbo mode to obtain better latency, throughput, energy, robustness to delay variations, or electro-magnetic compatibility. The dual citizen circuits in this paper have individual action control. As a result, the circuits can either run in a fixed mode — self-timed or clocked — and switch modes on the fly, or run in both modes concurrently.

Keywords: self-timed circuits, asynchronous, synchronous, dual mode

Figure 1 Carrying coal to Newcastle, ASYNC 2008.

Foreword

The title of this paper reflects an incident at the ASYNC 2008 conference chaired by Alex. Ivan, having a British mother, grew up knowing the futility of "Carrying Coal to Newcastle." Because ASYNC 2008 was in Newcastle, Ivan seized on the opportunity actually to carry coal to Newcastle. Being at pains to find coal in the San Francisco Bay area where it is a rare household fuel, he finally got a plastic sandwich bag of coal imported from Utah to California. He labeled it "mineral samples" to pass international inspection, and duly delivered it to Alex — see Figure 1.

1 Introduction

This paper expands [8] by not only naturalizing self-timed circuits but by turning them into dual citizens as well. Below, we explain what this means.

The "naturalized communication and testing" view [8] separates self-timed building blocks into links and joints. Links store and transport data. Joints serve as meeting points for links to coordinate state and exchange data. The actions of a self-timed system start in joints, and can be enabled or frozen selectively using separate *go* control signals in each joint. Joints act only when input links are full and output links are empty and *go* is enabled. Actions can be conditional or nondeterministic. For ease of explanation, this paper uses simple FIFO actions — see Figure 2.

Figure 2 shows a joint as a stick figure with data flowing in the direction of the arrow, and links as rectangles. Each link-joint-link triple represents a FIFO with an input link called *in*, and an output link called *out*. The most important property of a link is whether it is full or empty, just as the most important property of a parking place is whether or not it is occupied. We color the inside of a full link blue (grey in black and white print) and leave an empty link white.

Each link reports its full or empty state at both ends. It accepts a fill command at its input end and a drain command at its output end. Fill and drain commands change the state of a link. The impact of fill and drain commands is observed immediately at the near end of the link but may take time to traverse the length of the link before appearing at the link's far end.

The joint in Figure 2 acts only when its input link, *in*, is full and its output link, *out*, is empty, and its *go* signal is enabled. When it acts, it starts three concurrent operations that (1) copy the data from *in* to *out*, (2) drain link *in* — leaving it empty, and (3) fill link *out* — leaving it full. Note that when *go* is disabled, the joint is "frozen," and there is no action. Note also that the data value shown in Figure 2 as 60 stays in link *in* even after being copied — it may stay there until link *in* is filled with new data. By making link *in* empty and link *out* full, the action enables neighboring joints to act while it disables itself. This is what makes a self-timed circuit "tick."

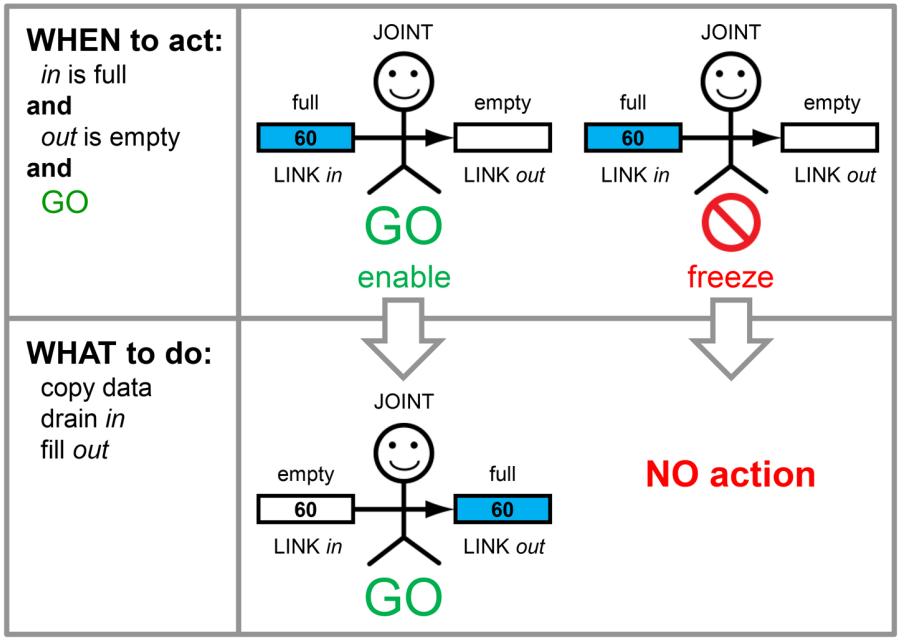

Figure 2 Self-timed FIFO action.

The full-empty protocol in Figure 2 works regardless of its handshake implementation. We advocate using it as standard interface to facilitate mixing and matching self-timed designs from different circuit families. The "naturalized" link and joint view offered in [8] captures the essence of self-timed systems.

This paper expands that view by adding a clocked or *synchronous* mode of operation to a naturalized self-timed circuit. Clock signals that retard the self-timed or *asynchronous* operation can be part of links or part of joints — we show circuits for each. A clocked link announces changes in its full or empty state only at times specified by its clocks. Likewise, a clocked joint acts only at times specified by its clocks. Not only do both self-timed and clocked modes of operation work, but simulations reported here also confirm mixed mode operation.

By providing a self-timed circuit with a clocked mode of operation, we aspire to increase the level of familiarity, comfort, and confidence of VLSI designers to integrate self-timed circuits into their systems. We regard the resulting circuit as both a self-timed and a clocked circuit — just as a "dual citizen" is regarded as a citizen of two countries. We therefore call this circuit a *dual citizen* circuit.

This paper is organized as follows. Section 2 shows a naturalized FIFO implementation for Figure 2 in Click, taken from [8], for use as reference design. Section 3 expands this reference design in two ways, by adding a clocked mode of operation to (1) its joint and (2) its links. Section 4 presents simulations for fixed mode operation, mode switching, and mixed mode operation. Section 5 discusses related work. Section 6 concludes the paper.

2 Naturalized Self-Timed Circuits

Most design methods for self-timed circuits use handshake protocols to encode the full or empty status of a link and validity of the link's data. Figure 3 shows a two-phase single-rail handshake [9] — or as we say a "two-phase non-return-to-zero handshake with bundled data" — and the way it encodes full, empty, and data validity. This protocol is used by the Click self-timed circuit family [6].

Click is the *most synchronous* asynchronous circuit family that we know. Its implementation style was chosen to resemble clocked or synchronous circuits as much as possible. It uses flipflops in every loop and it uses flipflops to store data. The flipflops facilitate the use of conventional optimization, timing, and test tools used for clocked circuits.

Figure 4 shows a self-timed circuit implementation for Figure 2, based on Click and two-phase non-return-to-zero handshake signaling with bundled data, but adapted for "naturalized communication and testing" [8]. The circuit has been adapted by moving the link-joint interface. The original interface separated links and joints at the handshake request, acknowledge, and data signals — here named *R*, *A*, and *Dstored*. The new interface separates the links and joints at their natural communication signals: *full*, *drain*, and *Dstored* for links carrying data into a joint, and *empty*, *fill*, and *D* for links carrying data away from the joint. The new interface takes full advantage of the handshake protocol without exposing it. It thereby diverts any "Tower of Babel" effect that a multitude of handshake protocols in use [9] might create if their handshake signals were exposed to each other.

By "naturalizing" the communication we gain translation-free communication. Moreover, we gain it whilst keeping the peculiarities of each handshake protocol and the specific skills that it supports to create circuits with better latency, throughput, energy, robustness, or electro-magnetic compatibility [1, 4, 5].

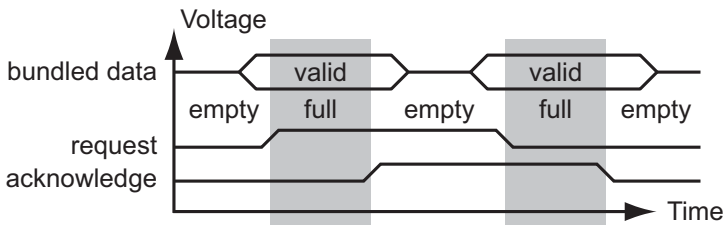

Figure 3 Example of a two-phase non-return-to-zero handshake with bundled data. This protocol has two control signals, *request* and *acknowledge*, and zero or more data signals, also known as *bundled data*. A link using this protocol is full when the voltage levels of its request and acknowledge differ, and empty otherwise. Its data signals are valid when the channel is full. A full channel may be drained, i.e. made empty, and an empty channel may be filled, i.e. made full.

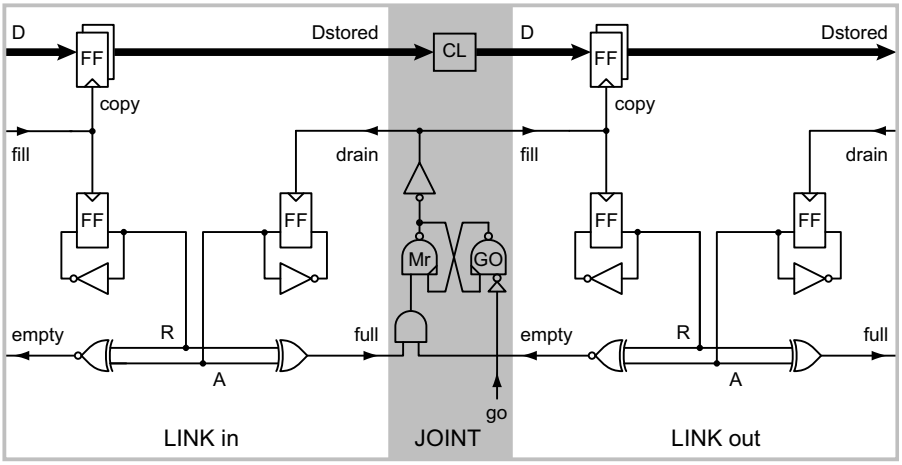

Figure 4 Naturalized Click FIFO circuit presented at ASYNC 2015 [8].

The links in Figure 4 use edge-triggered flipflops to store their full or empty state and the data they transfer. Combinational logic (CL) for datapath operations is kept in the joint. A FIFO that merely fills its output link with data copied from its input link uses simple wire connections for combinational logic.

Each link stores its full or empty state on two signals, a request signal, R, and an acknowledge signal, A. Each fill operation, performed as soon as signal *fill* goes high, changes R, making it differ from A. Each drain operation, performed as soon as signal *drain* goes high, changes A, making it match R. XOR and XNOR gates generate the full or empty state of the link by comparing the signal values of R and A. They report this state to signals *full* and *empty*. The link changes each R and A signal by complementing its value. Because R and A are separate signals, the link has separate flipflops to store the old and new values.

The joint in Figure 4 contains an AND function and the combinational logic for the datapath. The AND function combines the *full* and *empty* signals of links *in* and *out* with a *go* signal. When all three signals are high, the AND function "acts" by making signals *drain* and *fill* both high. Thus the action starts concurrently (1) a drain operation in link *in*, and (2) a fill operation in link *out* that copies the data from link *in*. In turn, the fill and drain operations make both *full* and *empty* signals low, thus disabling the AND function and causing both *drain* and *fill* to go low, which ends the action.

The *go* signal comes with its own arbiter to decide what to do when *go* is low. When *go* is low, the arbiter decides cleanly whether to stop at once or to complete a pending or ongoing action in the joint. The arbitrated circuit is called MrGO, pronounced "Mister GO" — see Figure 5(a). To control actions selectively, each MrGO has its own *go* signal. *We use MrGO for single-step, multi-step, and at-speed test and debug as explained in [8], and for switching between self-timed and clocked modes of operation as explained later in this paper.*

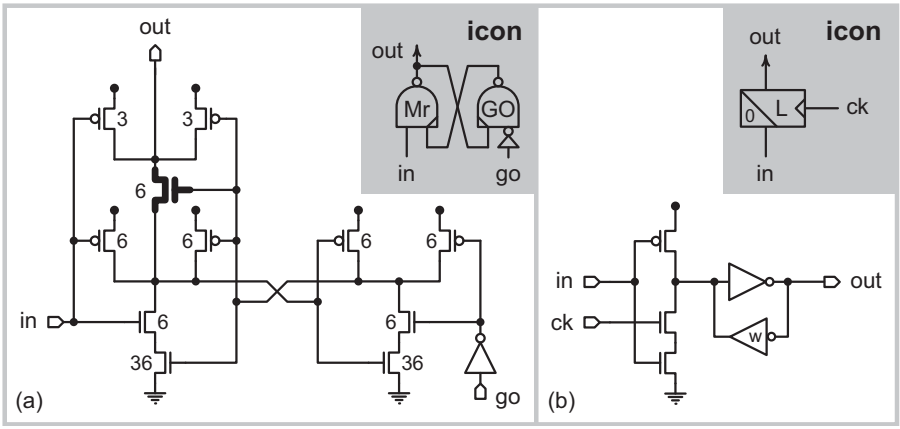

Figure 5 Transistor-level details for (a) MrGO and (b) a zero-passing latch.
(a) *MrGO*
The schematic for MrGO with its icon inset in the grey area is copied from [8], except that here we use correctly matching *in* and *go* parts. When *go* is high, MrGO acts as an inverter from *in* to *out*. When *go* is low, MrGO uses arbitration to decide cleanly whether or not to make *out* high. The bold central transistor delays active-low signal *out* by conducting only after metastability ends in favor of a low *out* signal. Metastability can occur during arbitration decisions, when a high-to-low (falling) transition on *go* to make and keep *out* high concurs with a low-to-high (rising) transition on *in* to make *out* low. Transistors are sized to reduce the logical effort from *in* to *out*. Split pull-up transistors avoid a floating *out* signal. **We use MrGO for single-step, multi-step, and at-speed test and debug as explained in [8], and for switching between self-timed and clocked modes of operation as explained in this paper.**
(b) *Zero-passing latch*
The latch stops a high input signal *in* from propagating until clock signal *ck* is high. When both *in* and *ck* are high, or when *in* is low, output signal *out* copies the value of *in*. Back to back inverters on *out* keep its value. To reduce drive fights, the reverse inverter of the keeper is weak.

3 Dual Citizen Circuits

The link and joint model of a self-timed circuit extends naturally to two solutions with a clocked mode of operation: clock the joint, as in Figures 6, or clock the link, as in Figure 7. The two *dual citizen* circuit solutions in Figures 6–7 both extend the Click circuit of Figure 4. Both circuits re-use the self-timed fabric and protocols. As a result, even in clocked mode, each circuit inherits the elasticity of the self-timed mode of operation to act only when and where needed. This is beneficial not only because it reduces power and saves energy, but also because it simplifies scheduling of clocked operations. With protocols rather than clock cycles in charge of the flow of control, the clocked operations of a dual citizen circuit can function correctly even when operating out of lockstep.

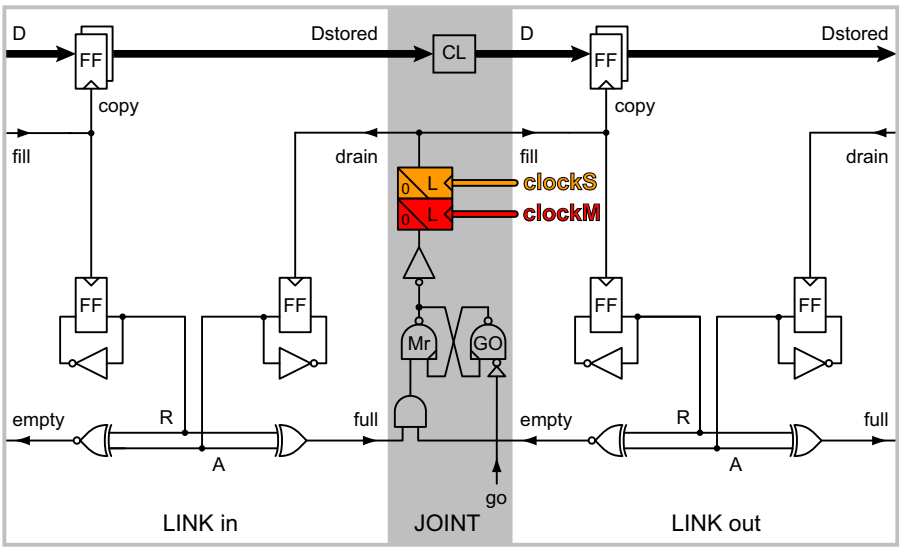

Figure 6 Dual citizen version of Figure 4 with clocks in the Joint.

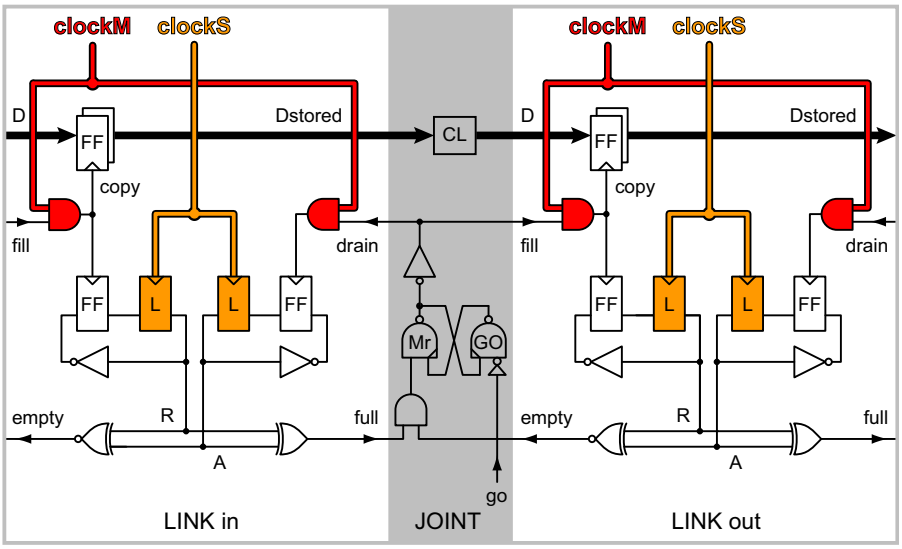

Figure 7 Dual citizen version of Figure 4 with clocks in each Link.

Because they re-use the self-timed fabric, the clocked circuit operations can build confidence in the self-timed circuit operations or replace aging or erratic self-timed circuit operations that need more time to finish.

Once confidence in the correctness of the self-timed operation is established, the self-timed mode can serve as "turbo mode" to obtain better latency, throughput, energy, robustness to delay variations, or electro-magnetic compatibility.

Both circuits use master and slave clocks, *clockM* and *clockS*. For self-timed operation, both clocks remain high. For clocked operation, a high pulse on *clockM* is followed by a high pulse on *clockS*.

The circuit in Figure 6 adds *clockM* and *clockS* at the end of the AND function in the joint where the clocks control a serial pair of zero-passing latches. For a transistor level schematic of a zero-passing latch, see Figure 5(b). In self-timed mode, *clockM* and *clockS* both remain high and the latches remain transparent to amplify the output signal of the MrGO controlled AND function. In clocked mode, each latch passes a low incoming signal by making its output low, but keeps its output as is when the incoming signal is high and its clock is low. A high incoming signal propagates only during a high pulse of the latch clock. Zero-passing latches allow the reset part of the joint action to run unhindered to completion — even in the clocked mode of operation. Because neighboring joints act in mutual exclusion, allowing each action to complete by making its *copy*, *fill*, and *drain* signals low before starting another action in the next clock cycle, facilitates the use of latches instead of edge-triggered flipflops in the datapath.

The key advantages of the dual citizen circuit in Figure 6 are (1) its simplicity and (2) its generality: many self-timed circuit families use the same joints [8]. Its key disadvantage is that the clocks leave some self-timed loops free running:

- The inverting loops of the link flipflops run freely, even in clocked mode. As a result, hold violations on these flipflops, due to an exceedingly fast data inversion loop, will affect both self-timed and clocked modes of operation.
- The action, once started, runs freely to completion. As a result, active-high pulse width violations on copy, fill, and drain signals are beyond clock control, and will affect both self-timed and clocked modes of operation.

The dual citizen circuit in Figure 7 adds *clockM* and *clockS* in each link. This circuit also allows the reset part of a joint's action to run to completion. But it does so while keeping a firm grip on all self-timed loops. The circuit in Figure 7 can repair all aging or erratic self-timed circuit operations that need more time to finish by switching to a clocked mode of operation, and by setting the high and low pulse widths for *clockM* and *clockS* as wide as needed.

In Section 4, we show simulation waveforms of dual citizen circuits interacting in self-timed and in clocked mode. Some interactions use MrGO to control joints selectively. In clocked operations, we change *go* signals during the low phase of *clockM*. None of the simulation scenarios in this paper require the arbitration function of MrGO. But if needed, arbitration in MrGO can be avoided in clocked operations by changing the *go* signal only when both *clockM* and *clockS* are low.

4 Simulation Experiments

We use four simulation scenarios to illustrate what one can do with dual citizen circuits. We simulate a ripple FIFO with ten joints, Joint 1 to Joint 10, and eleven links, Link 0 to Link 10, where Link 0 and Link 10 are connected to the external environment — see Figure 11. We assume that initially all links are empty (low) and all input signals and signal values stored in latches or flipflops are zero (low). For details on initialization, see [8].

All simulations were done in Verilog and use discrete delay models. For gate delays we model the number of signal inversions: each inversion counts as one time step. More precisely: signal changes through INVERTER and NAND gates take one time step. It takes two time steps to go through AND, X(N)OR, FLIPFLOPS, and LATCHES. The environment takes five time steps to respond. Environment actions that fill and drain a link are synchronized with a link's slave clock whenever the link operates in clocked mode, and are self-timed otherwise.[5]

All simulation waveforms shown in this paper are generated using the dual citizen Click FIFO with clocks added to the links, as shown in Figure 7, and with 6-bit wide data signals and simple wire connections for combinational logic. If instead of clocking the links we clock the joints, as shown in Figure 6, the Verilog test benches produce similar waveforms with the same test stimuli and responses.

Sections 4.1–4.3 below discuss the following simulation scenarios:

- Run in fixed mode mode, either self-timed or clocked.
- Switch modes after starting a self-timed operation to finish it clocked.
- Mix modes by running self-timed and clocked operations concurrently.

4.1 Run in Fixed Mode — Self-Timed or Clocked

The simulation waveforms in Figure 8 show the FIFO operating in self-timed mode. Those in Figure 9 show the FIFO operating in clocked mode.

The horizontal axis at the top of both Figures shows the progression of time throughout the course of the operation. The signal waveforms are displayed vertically, row by row. The vertical axis on the left shows the signal names. The signal called *start* indicates the end of initialization — we use it to start the operation cleanly. Any grey-colored, i.e. undefined, waveform values and any waveform changes prior to *start* going high can be ignored.

The master and slave clocks, *clockM* and *clockS*, are both high in Figure 8, as required for self-timed operation, while in Figure 9 they start ticking as soon as *start* goes high. Signals *in_empty*, *in_fill*, and *in_D* go between Link 0 and the environment. Likewise, signals *out_D*, *out_full*, and *out_drain* go between Link 10 and the environment. The remaining signals, *Dstored0* to *Dstored9*, are the data signals stored in Link 0 to Link 9.

[5] The reset part of fill and drain actions remains self-timed at all times.

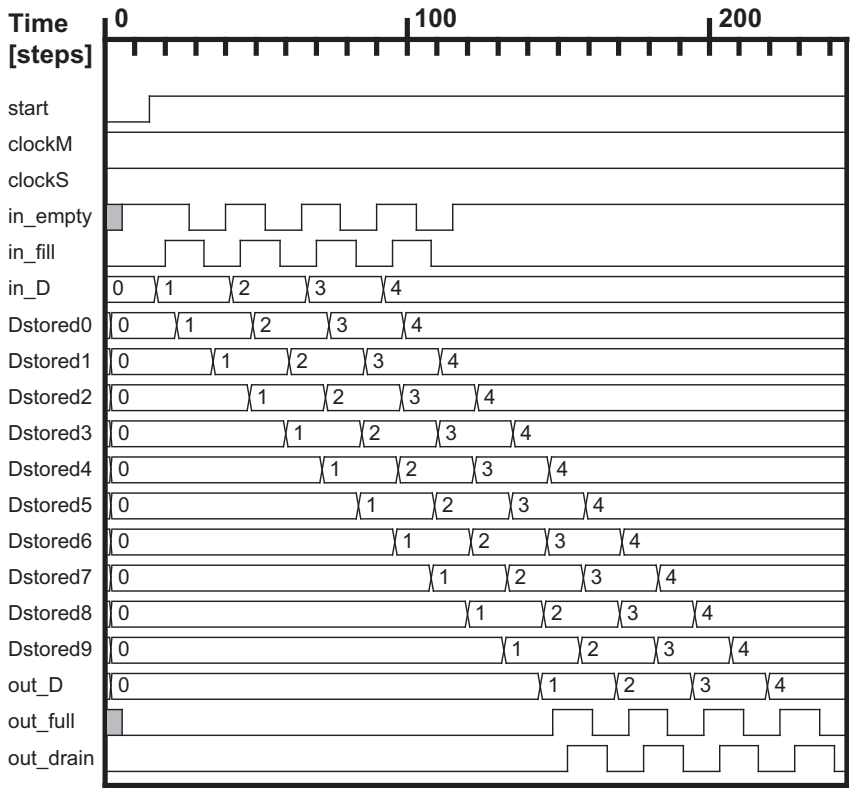

Figure 8 Self-timed FIFO operation transferring four data items.

As soon as *start* goes high, the environment starts communicating with the FIFO, using the protocols on Link 0 and Link 10. The entire operation consists of: (1) the environment sending four data items, with successive values 1 to 4, through Link 0 into the FIFO, (2) the FIFO forwarding these data items from Link 0 to Link 10, and (3) the environment collecting the data items at Link 10.

Note that data values stay in the links until they are overwritten by new values. This is particularly visible for the initial data values of 0 and for the final data values of 4. The FIFO's output data, *out_D*, for instance, keeps its initial value 0 for approximately 140 time steps in Figure 8, which is how long it takes a data item to ripple through the FIFO in self-timed mode. In the clocked mode of operation shown in Figure 9, this takes approximately 700 time steps. Likewise, the value 4 of the last data item stays on all the links after all four data items have rippled through the FIFO.

Figures 8–9 show that both the self-timed and the clocked operation are immune to old data values lingering on links. This is as expected, because both modes of operation obey the same dataflow protocols. Both use the full or empty status of a link to decide when to pay attention to and when to ignore the link's data.

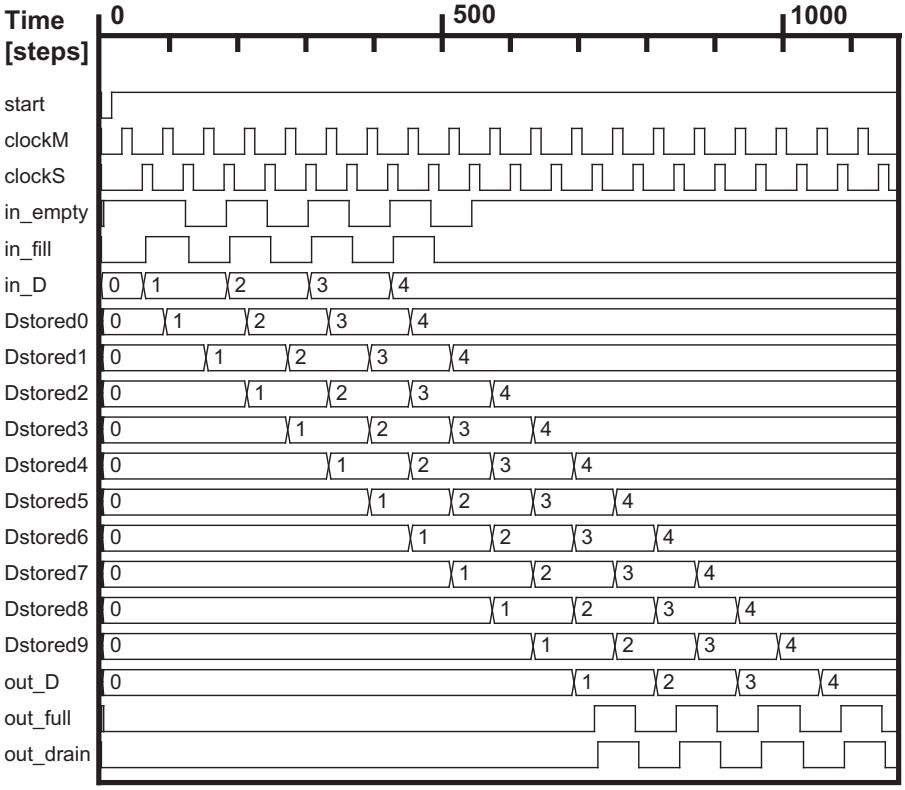

Figure 9 Clocked FIFO operation transferring four data items.

As a result — omitted from Figure 9 — the master and slave clocks in the clocked operation can keep ticking after the operation completes, without jeopardizing any of the final values of status or data signals.

As noted earlier, the clocked mode of operation shown in Figure 9 is slower than the self-timed operation in Figure 8. The clocked operation is slower because the clocks retard the self-timed fabric and protocols. The clock periods must be longer than the worst-case cycle times of the self-timed fabric and protocols.

4.2 Switch Modes — from Self-Timed to Clocked

The simulation waveforms in Figure 10 show the FIFO first operating in self-timed mode and then switching mode to operate in clocked mode. The overall operation is similar to each of the operations shown in Figures 8–9: four data items pass through the FIFO. Approximately the first 250 time steps of the simulation run self-timed. In self-timed mode, we execute the first half of the operation, which consists of (1) the environment sending four data items, with

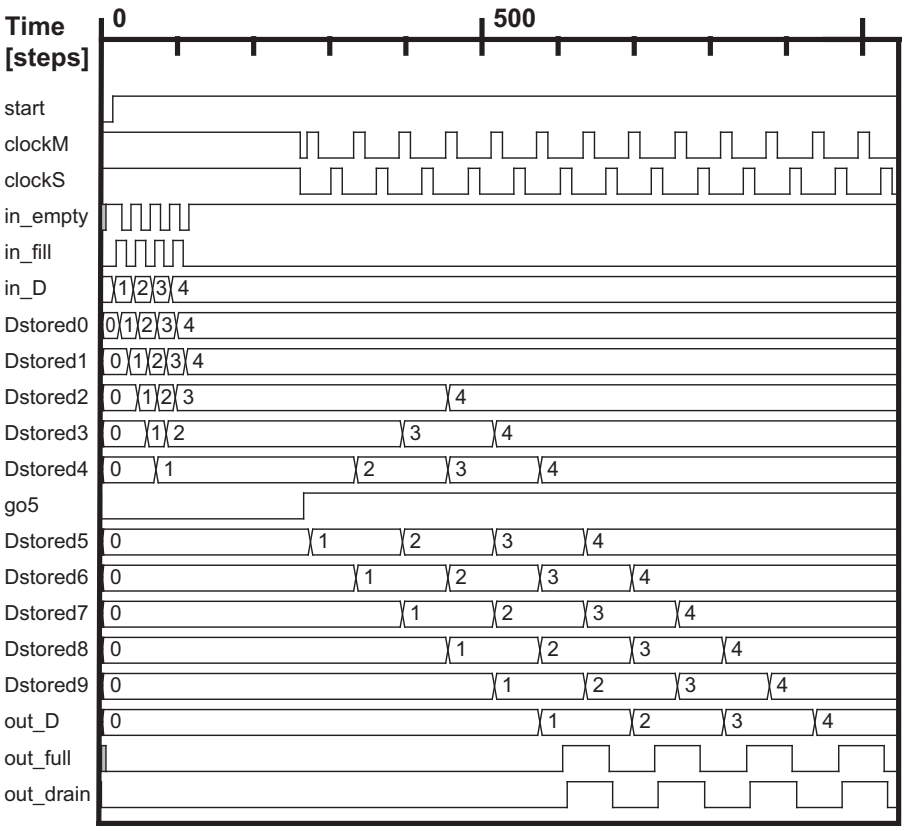

Figure 10 From self-timed to clocked FIFO operation passing four data items.

successive values 1 to 4, through Link 0 into the FIFO, and (2) the FIFO storing these data items in Link 0 to Link 4. The remaining time steps of the simulation run clocked. In clocked mode, we execute the second half of the operation, which consists of (3) the FIFO forwarding the data items stored in Link 0 to Link 4, and (4) the environment collecting the data items at Link 10. In between, the mode of operation switches from self-timed to clocked.

To split the operation in two and switch the mode of execution between the two halves, we deploy MrGO [8]. Specifically, we use go control signal *go5* of Joint 5. Signal *go5* is the only new signal that we inserted into the waveform display of Figure 10 — just below the center. All other signals match those of Figures 8–9.

A pictorial view of the role of *go5* in splitting the operation follows in Figure 11. First, we freeze Joint 5, by making *go5* low — see Figure 11(a). This prevents Joint 5 from acting. Then we run the first half of the operation in self-timed mode. The operation ends in a stable state in which Link 1 to Link 4 are full and the other links are empty — see Figure 11(b). The stable state allows us to switch

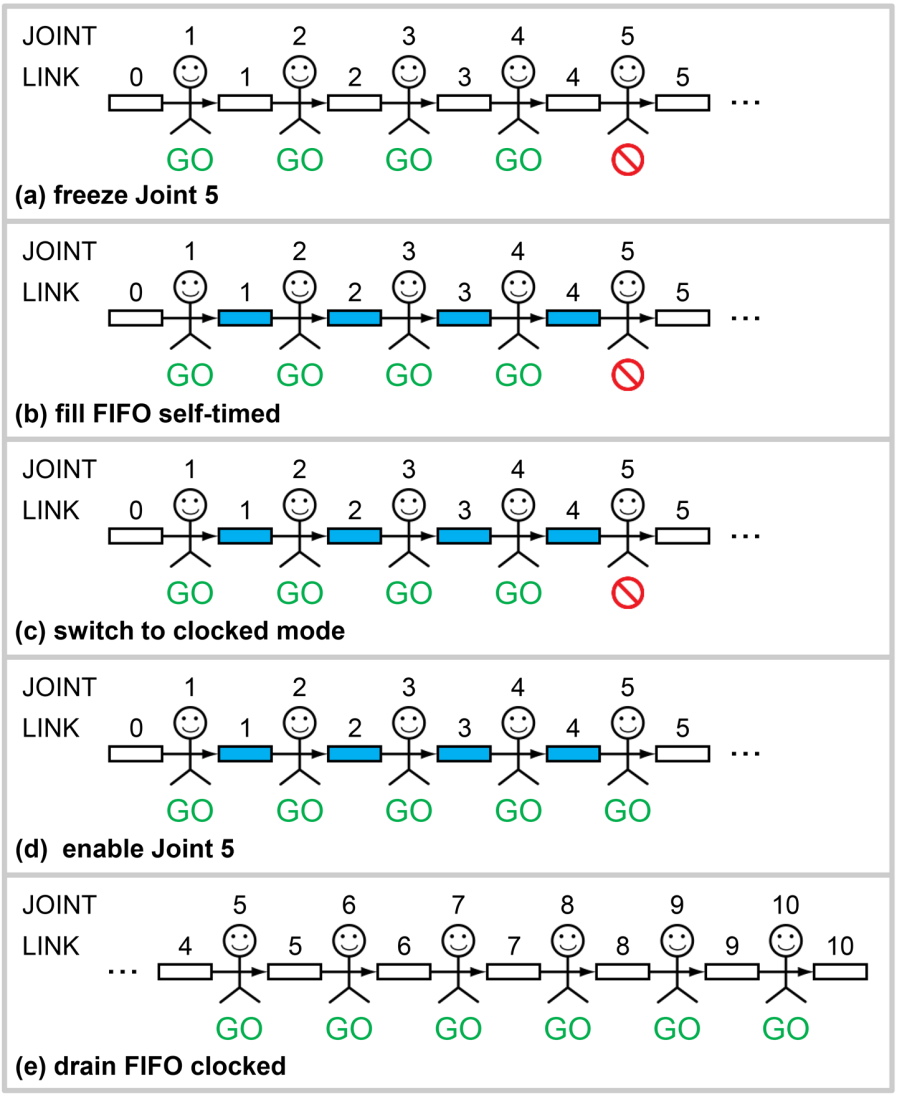

Figure 11 Pictorial view of how to switch modes from self-timed to clocked. We color full links blue (or grey) and empty links white — see also Figure 2.

the mode of operation reliably from self-timed to clocked — see Figure 11(c). Next, we enable Join 5 by making *go5* high — see Figure 11(d). To do this safely, *go5* must change from low to high during the low phase of the master clock, *clockM*. The waveforms in Figure 10 show a safe low to high transition for *go5* sufficiently in advance of the first high pulse on *clockM*, around 275 time steps into the simulation. The second and clocked half of the operation can now forward and drain the four data items from the FIFO — see Figure 11(e).

4.3 Mix Modes — Self-Timed and Clocked

The waveforms in Figure 12 show the FIFO operating simultaneously in both self-timed and clocked modes. In addition to showing waveforms for the signal names introduced earlier for Figures 8–10, Figure 12 also includes the waveforms for go control signal *go7* of Joint 7, and for status signals *empty5* and *empty6* of Link 5 and Link 6. The FIFO is partitioned into three regions that share the same source clocks but can run in different modes of operation:

- **Region 1**, the FIFO's input region, covers Link 0 to Link 4, and operates continuously in self-timed mode. Signals clockM1, clockS1 refer to its clocks.
- **Region 2**, the FIFO's *airlock*, covers Joint 5 to Joint 7. It switches mode repeatedly. Signals clockM2 and clockS2 refer to its clocks.
- **Region 3**, the FIFO's output region, covers Link 7 to Link 10, and operates continuously in clocked mode. Signals clockM3 and clockS3 refer to its clocks.

We call the middle region, Region 2, "the FIFO's airlock" because it permits status, control, and data to pass reliably between the input and output regions of the FIFO, Region 1 and Region 3, just as an airlock permits safe passage of people and objects between environments of different air pressures.

A pictorial view of how the airlock provides safe passage of status, control, and data between Region 1 and Region 3 follows in Figure 13. To operate the airlock, we deploy the two MrGO circuits in Joint 5 and Joint 7 — the two joints that separate the airlock from its neighbors. By freezing or enabling Joint 5, using go control signal *go5*, we disconnect the airlock from or engage it with its predecessor region in the FIFO, Region 1. Likewise, freezing or enabling Joint 7, via *go7*, disconnects the airlock from or engages it with its successor region in the FIFO, Region 3. The use of go control signals to accommodate the airlock operation is an extension of their use in the mode-switching operation depicted in Figure 11.

The waveform and pictorial views in Figures 12–13 relate to each other as follows:

- **Figure 13(a) — fill airlock self-timed**
  Initially, Joint 5 is enabled and Joint 7 frozen, because *go5* is high and *go7* low. This engages the airlock with Region 1 for self-timed filling. The fill operation ends with Links 0 to 6 full, around 200 time steps into Figure 12.
- **Figure 13(b) — engage airlock with clocked successor**
  The stable state of the airlock allows us to disconnect the airlock safely from its self-timed predecessor, Region 1, which we do by freezing Joint 5, i.e. by making *go5* low. Now, we can switch the airlock's mode of operation reliably from self-timed to clocked. Next, we engage the airlock with its clocked successor, Region 3, by enabling Joint 7, i.e. by making *go7* high. To do this safely, *go7* must change during the low phase of the master clock. The waveforms in Figure 12 show a safe low to high transition for *go7* sufficiently in advance of the engaging high pulse on *clockM2* or *clockM3*, now the same, around 275 time steps into the simulation.

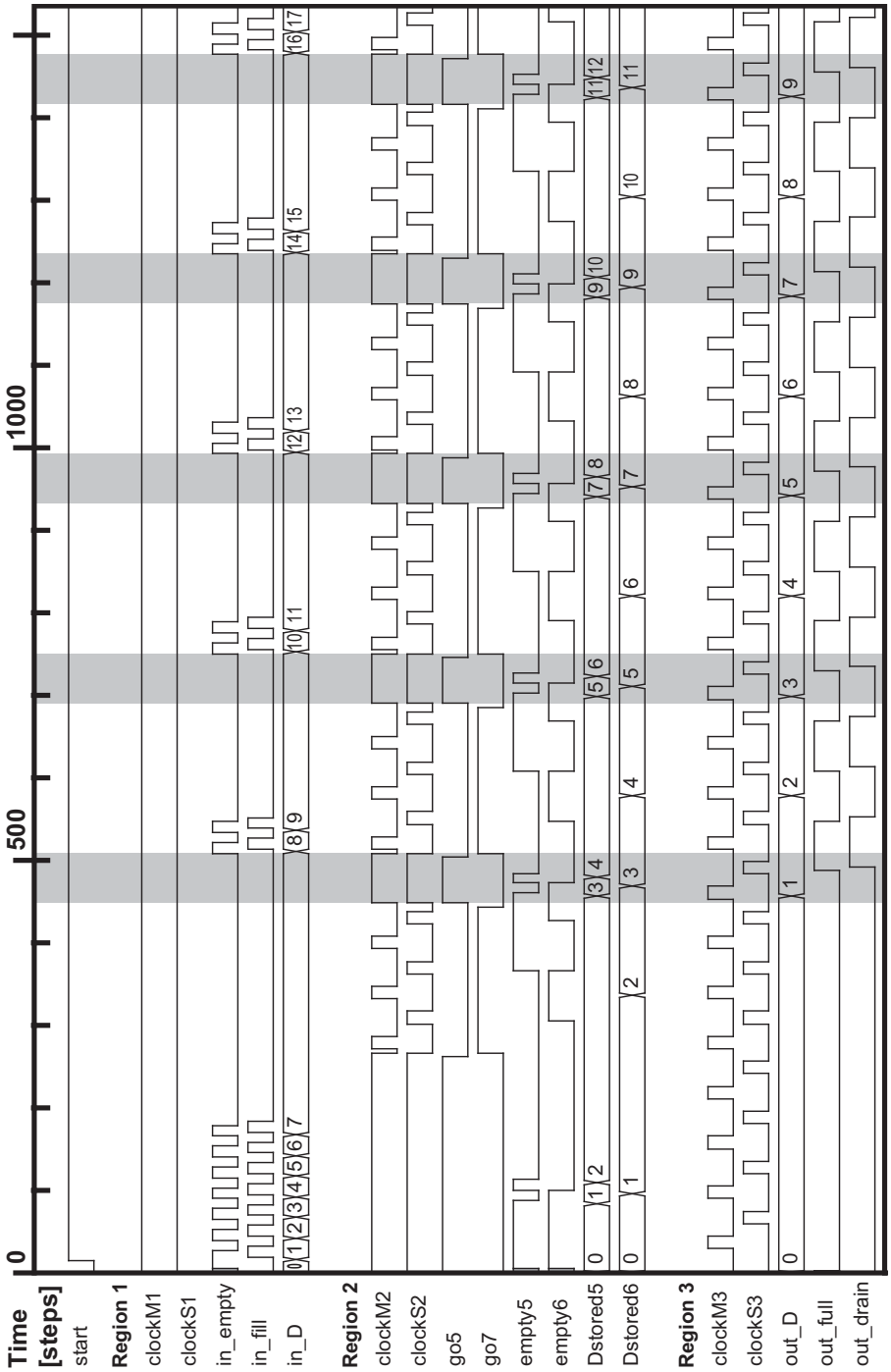

Figure 12 Mixed-mode self-timed and clocked FIFO operation with airlock.

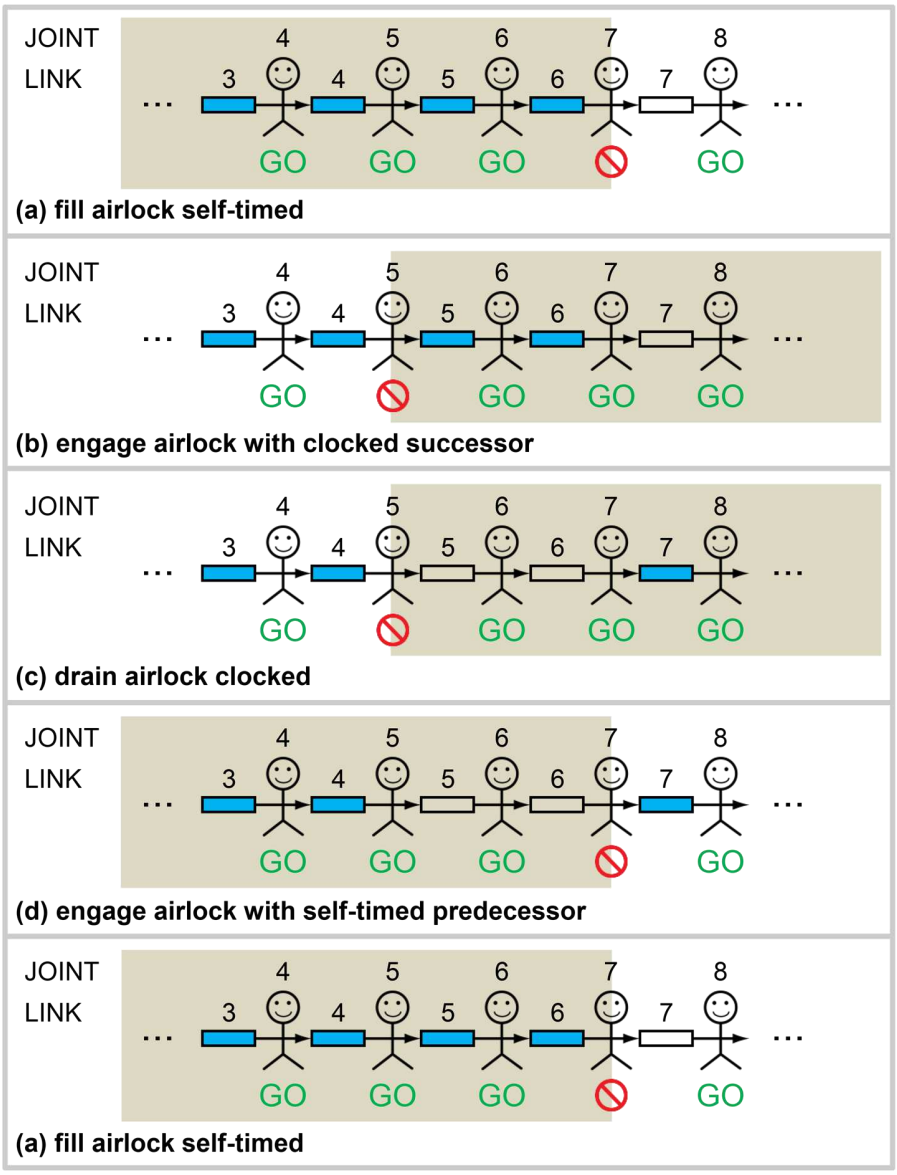

Figure 13 Pictorial view of the airlock from Joint 5 to Joint 7 passing data. We color full links blue (or grey) and empty links white — see also Figure 2.

- **Figure 13(c) — drain airlock clocked**
  We can now drain the airlock using a clocked mode of operation. The drain operation forwards data value 1 on *Dstored6*, stored in Link 6, followed by data value 2 on *Dstored5*, stored in Link 5, draining both links in the process. The drain operation ends in a stable airlock state with Link 5 and Link 6 both empty, around 450 time steps into Figure 12.
- **Figure 13(d) — engage airlock with self-timed predecessor**
  The stable airlock state allows us to disconnect the airlock safely from its successor, Region 3, which we do by freezing Joint 7, by making *go7* low. Next, we engage the airlock with its self-timed predecessor, Region 1, by making *clockM2* and *clockS2* both high to enable self-timed operation, and by making *go5* high to enable Joint 5. These engagement steps happen shortly after the airlock becomes empty, about 450 time steps into Figure 12.
- **Figure 13(a) — fill airlock self-timed**
  We can now fill the airlock using a self-timed mode of operation, just like we did initially. The fill operation ends with Link 0 to Link 6 full, and with a data value of 3 on *Dstored6* in Link 6 and a data value of 4 on *Dstored5* in Link 5 — about 500 time steps into Figure 12. This specific fill operation is marked by the first grey-colored vertical band in Figure 12. Similar grey-colored bands mark similar fill operations further along in the simulation.

Note that each grey-colored band in Figure 12 starts with *empty5* and *empty6* both high, and ends with *empty5* and *empty6* both low. This indicates that each grey-colored band starts with an empty airlock and ends with a full airlock. The airlock is filled using a self-timed mode of operation within a grey band, and is drained using a clocked mode of operation between grey bands.[6]

Note also that the data exchange rate at the output of the FIFO is constant. The output environment receives a new data value for *out_D* for every two pairs of non-overlapping *clockM3*–*clockS3* pulses, i.e. every two clock cycles. This is the fastest clock cycle time that the simulated dual citizen Click circuit can support. *The output environment works at full speed, and can be completely agnostic of the existence of the self-timed input environment and the FIFO's airlock!*

5 Comparison to Related Work

The idea of clocking a self-timed circuits is not new by itself. Prior work published in [7, 10, 3, 2] explains how to combine clocks with handshake protocols or with other forms of elastic protocols. Only the circuits published in [10] and [3] use both a clocked and a self-timed mode of operation, as do we, but neither publication discusses running a system in both modes concurrently as we do in Figures 12–13. Below follows a more specific comparison.

The work reported in [7] adds clocks to initially self-timed handshake circuits but then optimizes the circuits for synchronous operation by simplifying those

[6] Note that *clockM2* and *clockS2* remain high within a grey band, for self-timed filling, and match *clockM3* and *clockS3* between grey bands, for clocked draining.

parts of the circuits that provide flow control for self-timed operation but that are redundant under clocked operation. The resulting circuits, though generated with the same design flow, are no longer self-timed and may have lost some of their elasticity, but can be used for FPGA mappings or for integration into a completely synchronous system.

The circuits presented in [2] remain elastic when clocked, and will thus tolerate variations in computation and communication delays when clocked. The supporting design, analysis, and optimization techniques described in the paper can be used for clocked as well as for self-timed circuit designs. As such, the choice "to clock or not to clock" can be deferred until late in the design process. The paper gives no examples nor any indication of keeping both choices, clocked and self-timed, available to the final circuit implementation.

The dual-mode synchronous/asynchronous CORDIC processor for wireless broadband communication presented in [3] can select its mode of operation to fit system demands and application needs. For instance, when the received signal is weak, the processor can be switched into self-timed mode to reduce electro-magnetic interference. The clocks in [3] bypass the handshake control circuits. Consequently, clocked operation of the CORDIC forfeits the elasticity provided by the handshake protocols. Also, in bypassing the handshake control, the CORDIC's clocked mode of operation will be of marginal use for building confidence in the CORDIC's self-timed operations.

In contrast to [3], the clocked or synchronous mode of operation implemented in [10] re-uses the self-timed fabric and protocols — as do we. The resulting level-sensitive synchronous mode of operation thus inherits the elasticity of the self-timed mode of operation. As a result, the synchronous mode of operation can be used to build confidence in the self-timed circuit operations, which is one of the key reasons for us to add it, though this is not addressed in [10] which mentions only its potential use for system-level diagnosis and debug. The paper provides a systematic solution for adding clocks and test inputs to a self-timed circuit. The use of clocks to run the circuit in synchronous mode acts as a stepping stone in that solution. The key feature of [10] is the systematic addition of a clocked scan test mode of operation.

The dual citizen circuits that we present in this paper offer a new approach to clocking self-timed circuits, because the circuits are built around the ideas of naturalized communication and testing [8]. By differentiating *links* from *joints* the design solutions for adding a clocked mode of operation fall naturally into solutions that clock the links versus solutions that clock the joints. By differentiating *actions* from *states* we can avoid adding energy-costly slave latches into the datapath that an action-agnostic approach like [10] would add, because we know that the joints at opposite ends of a link act in mutual exclusion. Last but not least, we can control actions individually, using MrGO. By enabling or freezing selective actions at run time, different parts of the circuit can be made to (1) run in different modes, (2) switch modes reliably, and (3) exchange data without the need for synchronizers.

6 Conclusion

The "naturalized communication and testing" view [8] unifies thinking about a wide variety of self-timed circuit families and facilitates mixing and matching these families within a single system. In this paper, we extend this unity to embrace clocked circuits; we add a clocked or synchronous mode of operation to self-timed circuits. We call the resulting circuits *dual citizen* circuits.

As reference circuit, we chose a ripple FIFO implemented in Click [6] but adapted for naturalized communication and testing [8]. Its dual citizen solutions and simulation results apply broadly to other self-timed designs and circuit families.

Clock signals that retard self-timed operation can be part of *links* or of *joints*. Links store and transport data. Joints act on the data. We have shown how to add clocking to each. Each of our clocking additions re-uses the self-timed protocols, and thereby inherits their elasticity to act only when and where needed. Need-driven action is beneficial not only because it saves energy, but also because it simplifies scheduling of operations. With protocols rather than clock cycles in charge of flow control, the clocked operations of a dual citizen circuit can function correctly even when operating out of lockstep.

By recognizing joint *actions*, we can avoid adding latches and clocked gates into the datapath As a result, dual citizen circuits operating in self-timed mode can maintain the energy-efficiency of the original self-timed circuit.

By enabling or freezing selective actions at run time, using MrGO, different parts of the circuit can (1) run in different modes, (2) switch modes reliably, and (3) exchange data without the need for synchronizers. We have shown simulations of fixed mode operation, mode switching, and mixed mode operation.

The clocked mode of dual citizen circuits can bolster confidence in the correct functionality of the self-timed mode — or vice versa — in various ways.

- Engineers most comfortable with clocked systems can easily see how dual citizen circuits work when clocked. The datapath is identical in both modes and so may be understood in either mode. The self-timed control fabric and protocols are shared in both modes. Seeing the control work when clocked can therefore build confidence in its self-timed behavior.
- Simulations reported here exhibit mixed mode behavior. Data received in self-timed mode may be delivered to a clocked destination and vice versa. The ability to change between clocked and self-timed modes of operation can bolster confidence in the correctness of either mode.

Because their self-timed mode of operation is faster than their clocked mode, the clocked mode of operation can be used as a "crutch" to support aging or erratic self-timed circuit operations that need more time to finish. The link-based clocking addition makes a good crutch because it keeps a firm grip on self-timed loops and can retard any of these as much as needed by using wider clock pulses. On the other hand, the self-timed mode of operation can provide a "turbo" performance boost when needed, to obtain better latency, throughput, energy, robustness to delay variations, or electro-magnetic compatibility.

Moving safely from clocked circuits through self-timed circuits and then back again provides a path for synchronous designers to embrace self-timed design incrementally. We clear this path by providing self-timed circuits with a clocked mode of operation. This approach deserves thorough study, so the costs in terms of design, analysis, verification, and engineering can be quantified, and — we hope — proven competitive with the state of the art in distributed VLSI design.

Acknowledgement

We thank corporate and private sponsors of the Asynchronous Research Center at Portland State University. We also thank the National Natural Science Foundation of China for sponsoring part of this work under Grant No. 61402121. Last but not least, we thank our friend and colleague Alex Yakovlev, for his unforgettable "Terminator" accent — now either lost or too familiar to us — and for the many years of camaraderie, open-mindedness, and fresh ideas and students he carried with him to and from Newcastle. To strengthen our ties, this paper ends with a family recipe for Pavlova from the New Zealand branch of the Sutherland clan. Face shots are in order of authors. Happy birthday, Alex!

References

1. Peter Beerel and Marly Roncken. Low Power and Energy Efficient Asynchronous Design. *Journal of Low Power Electronics (JOLPE)*, 3(3):234–253, 2007.
2. Joseph Carmona, Jordi Cortadella, Mike Kishinevsky, and Alexander Taubin. Elastic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1437–1455, 2009.
3. Eckhard Grass, Bodhisatya Sarker, and Koushik Maharatna. A Dual-Mode Synchronous/Asynchronous CORDIC Processor. In *Asynchronous Circuits and Systems (ASYNC)*, pages 76–83, 2002.
4. Steven Nowick and Montek Singh. Asynchronous Design — Part 1: Overview and Recent Advances. *IEEE Design & Test*, 32(3):5–18, 2015.
5. Steven Nowick and Montek Singh. Asynchronous Design — Part 2: Systems and Methodologies. *IEEE Design & Test*, 32(3):19–28, 2015.
6. Ad Peeters, Frank te Beest, Mark de Wit, and Willem Mallon. Click Elements: An Implementation Style for Data-Driven Compilation. In *Asynchronous Circuits and Systems (ASYNC)*, pages 3–14, 2010.
7. Ad Peeters and Kees van Berkel. Synchronous Handshake Circuits. In *Asynchronous Circuits and Systems (ASYNC)*, pages 86–95, 2001.
8. Marly Roncken, Swetha Mettala Gilla, Hoon Park, Navaneeth Jamadagni, Chris Cowan, and Ivan Sutherland. Naturalized Communication and Testing. In *Asynchronous Circuits and Systems (ASYNC)*, pages 77–84, 2015.
9. Jens Sparsø and Steve Furber (Eds.). *Principles of Asynchronous Circuit Design — A Systems Perspective*. Kluwer Academic Publishers, 2001.
10. Kees van Berkel, Ad Peeters, and Frank te Beest. Adding Synchronous and LSSD Modes to Asynchronous Circuits. In *Asynchronous Circuits and Systems (ASYNC)*, pages 161–170, 2002.

Pavlova photo by Hazel Fowler, Wikimedia Commons

Pavlova

3 (old) egg whites
9 oz castor sugar
1 tsp vanilla
1 tsp vinegar
1 pinch salt
cream
fruit

Beat egg whites with a pinch of salt
until stiff enough to peak.
Fold in sugar, vanilla, and vinegar.
Place on baking paper on greased tray.
Bake slowly about 1-1.5 hours at 250F.
Dress with fresh whipped cream,
kiwi fruit, strawberries, or similar.
BON APPÉTIT !