Low-Power Dual-Edge-Triggered Synchronous Latency-Insensitive Systems

Dimitris Konstantinou, Anastasios Psarras, Giorgos Dimitrakopoulos
Electrical and Computer Engineering
Democritus University of Thrace, Xanthi, Greece

Chrysostomos Nicopoulos
Electrical and Computer Engineering
University of Cyprus, Nicosia, Cyprus

***Abstract*—Latency-insensitive data flow is a design paradigm that tolerates the latency variability of computations and communications and allows for correct-by-construction module integration. In this paper, we aim to reduce the dynamic power consumption of synchronous latency-insensitive systems by reducing the power of their clock network. In order to save on clocking power, we employ a Dual-Edge-Triggered (DET) clocking strategy and flow-control rules, whereby the clock operates at half the clock frequency, and data flow occurs on both rising and falling clock transitions. To support this operation, new low-cost DET elastic buffers are proposed that allow for full-throughput operation using only two latches per buffer, and without incurring any additional overhead relative to their baseline single-edge-triggered counterparts. Hence, the two design elements (flow control and elastic buffers) work synergistically to yield a highly efficient fundamental primitive building block that can seamlessly facilitate DET clocking in latency-insensitive systems.**

I. Introduction

Latency-insensitive (aka elastic) operation relaxes the strict global event scheduling requirements of conventional synchronous designs and enables the dynamic scheduling of operations based on the availability of the corresponding data [1], [2]. The latency-insensitive design paradigm preserves the functional correctness of the computations, or across-module communication, with respect to the trace of valid data observed at the inputs and outputs of the computation units and communication channels, even if the latency of the operations may vary dynamically at runtime. This is achieved through a distributed flow-control mechanism that governs the flow of valid data across the system [3], [4].

The flexibility offered by latency-insensitive system operation is exploited in System-on-Chip (SoC) IP integration using latency-insensitive wrappers, while its control handshake semantics are widely used in on-chip communication protocols such as the AMBA AXI and their Network-on-Chip (NoC) implementation [5], [6], [7]. Latency-insensitive operation is also preferred in certain forms of high-level synthesis, where hardware units are synthesized from high-level descriptions, or from dataflow programming models [8], [9], [10].

Over the last several years, low-power operation has become a key design criterion in digital systems. The effort to achieve low-power operation starts at the architectural level and continues through implementation and physical design. Clock power optimization is one of the most important objectives, as clock power can contribute a significant amount of the dynamic power consumption of a synchronous digital design [11]. The dynamic power consumption of the clock with frequency f is mostly due to the switching of capacitances and it is equal to $\frac{1}{2}\, a\, f\, C\, V_{DD}^2$, for a capacitance C (dis)charging between 0 V and supply voltage V_{DD} and switching frequency a.

Clock gating reduces the switching frequency, while Dual-Edge-Triggered (DET) synchronous operation enables the reduction of the clock frequency to half without changing the system throughput by sampling data on both the rising and the falling edges of the clock [12], [13]. DET clocking *halves* the power dissipation of the clock network, leading to significant overall system power savings.

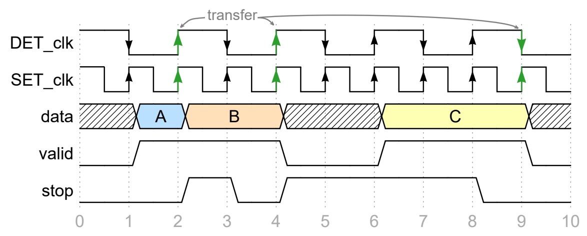
This work applies – *for the first time*, to the best of our knowledge – DET clocking to latency insensitive designs. Combining DET operation and latency-insensitive data flow requires modifications in both the protocol- and the circuit-level implementations of the dataflow blocks. We first introduce the semantics of the new flow control protocol that enables operation on both edges of the clock. Subsequently, we present novel and very low-cost DET elastic buffers comprising only two latches. These buffers can sample, store, and propagate their input on both clock edges, while they can be stopped by upstream elastic buffers on either clock edge.

Overall, this work achieves a double-faceted goal: to increase the applicability of the DET clocking approach, and to reduce the clock-tree power of latency-insensitive systems. The DET clocking strategy alleviates frequency scaling problems by retaining the data throughput of SET clocking at *half* the clock frequency. At the same time, latency-insensitive operation offers a practical solution to variable interconnect latencies (due to short and longer wires), and simplifies physical design and integration even at high clock frequencies. The seamless integration of DET clocking in latency-insensitive systems – as proposed in this work – significantly increases the versatility and scalability of the latency-insensitive concept in the nanometer regime.

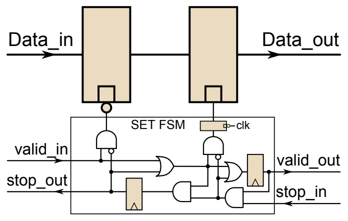
II. SET flow control and Elastic buffers

The implementation of elastic flow control requires special clock-edge-triggered registers, called *Elastic Buffers* (EB). Besides their data inputs and outputs, EBs also provide two additional control signals, *valid* and *stop*, which enable the generation of local forward and back pressure, i.e., elasticity in the data's flow. The valid signal indicates when an upstream transmitter is attempting to send valid data. The stop signal is used to signal the downstream receiver's inability to capture data sent in the current clock cycle. In this case, the transmitter should hold the data on the link until the receiver resumes from the stall. On the contrary, when valid=1 and stop=0, a successful data transfer occurs [8].

In the Single-Edge-Triggered (SET) implementation of this protocol, the assertion and the de-assertion of the valid and stop bits are aligned to one edge of the SET clock, as shown



(a)



(b)

Fig. 1. (a) An example of the operation of a flow-controlled (elastic) channel, where a SET or DET design uses the SET/DET_clk accordingly, and (b) the latch-based implementation of a SET EB.

in Fig. 1(a). Thus, any data transfer between the downstream and upstream modules occurs once per cycle.

The most efficient SET EBs are built using a master and a slave latch connected in series – similar to a positive (negative) edge-triggered flip-flop – that are controlled by a simple SET FSM, as shown in Fig. 1(b)[1]. By connecting two latches in series, driven by opposite levels of the clock (i.e., one latch will be in the opaque state, while the other in the transparent state), there is no direct path that connects the input of the EB to its output, since the output terminal is always connected to a latch that is opaque. This feature isolates timing paths across any two connected EBs [2], [3]. Thus, when the output of an EB chain stalls, the stall can only propagate back one stage per cycle, leaving any upstream EB blind to the state of the downstream EB for one clock cycle.To handle this, all EBs hold two words, one for the stalled output, and one "captured" when necessary from the previous stage. Effectively, each EB acts as a 2-slot FIFO informing the upstream EB to stop, only when it is full.

III. Dual-Edge-Triggered Flow Control and Elastic Buffers

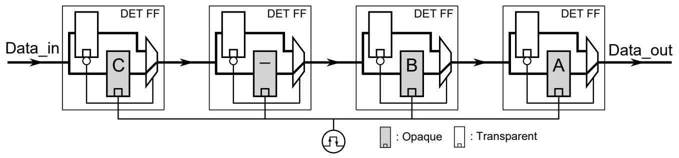
The extension of the valid/stop handshake protocol to the DET clocking scheme maintains the same fundamental flow control rules of the SET approach, but includes one fundamental differentiation: the valid and stop bits can change on *both* edges of the clock.

A data transfer occurs between the transmitter and the receiver when valid=1 and stop=0, irrespective of which clock edge (rising/falling) this condition is satisfied. Therefore, data can effectively be transferred on both clock edges, yielding the same data transfer throughput as the SET flow-control rules at *one half the clock frequency*. This property is shown in Fig. 1(a), where the functionality remains unchanged, even though the DET clock is used as reference. Data produced by the transmitter on various clock edges are consumed on the following (and opposite) clock edge, if the receiver is ready to accept it.

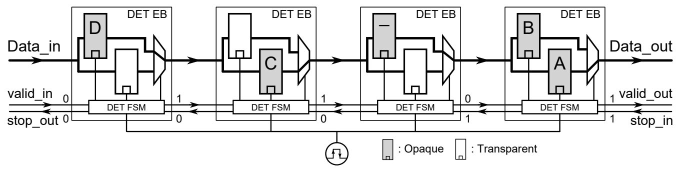
Either side can assert or de-assert their corresponding flow-control signals on both clock edges. For example, the receiver that was ready (stop=0) can become unavailable (stop=1) on the following clock edge, and vice versa. Once the receiver is not ready, the transmitter is obligated to hold its valid data for all the following clock edges, until the receiver becomes ready, irrespective the clock edge that the valid data first appeared. In Fig. 1(a) such case is present, where valid data C appears at the output of the transmitter positive at clock edge #6(of the DET clock). This data word is consumed by the receiver on negative clock edge #9, when the receiver is actually ready to accept it (the stop signal is de-asserted on positive clock edge #8).

A. Transforming a DET flip-flop into a DET elastic buffer

A DET flip-flop typically utilizes two latches in parallel [16], [17], [18], as illustrated in Fig. 2(a). Scan DET FFs needed for testing the circuit after fabrication follow also the same structure [19]. On every clock transition, one latch samples upstream data, i.e., the transparent one, while the other is opaque and selected by the multiplexer to drive the output. It is imperative that the output is driven only by an opaque latch to isolate (by construction) any input-output paths and avoid any data races [17].



(a): DET flip-flops and pipeline without flow control.



(b): DET flow-controlled (elastic) pipeline with progressive valid/stop signal propagation.

Fig. 2. The operation of (a) simple, and (b) flow-controlled (i.e., elastic) pipelines comprising DET flip-flops and DET EBs, respectively.

When transforming a DET pipeline to a latency-insensitive one, we assume that (a) each flip-flop is replaced by an EB, and (b) the valid and stop signals propagate one stage per half-cycle for maximum timing flexibility [12]. Thus, similar to SET latency-insensitive pipelines, to handle progressive stall propagation, each EB needs to store two data items. In this paper, we will show how to use the existing latches of a DET flip-flop and "transform" them into 2-slot buffers during stalls. The other latency-insensitive pipeline primitives, such as forks, joins, branches, and selects [3], [14] remain the same as in the case of SET latency-insensitive pipelines.

Unlike a simple DET flip-flop, a DET EB facilitates *elasticity* in the data flow, i.e., it must be flow-controlled. The desired behavior is as follows: when there is no stall in the data flow, the EB should keep one latch transparent and the other opaque,

[1]Besides latch-based implementations, various flip-flop-based implementations of EBs are also possible [14], [15].

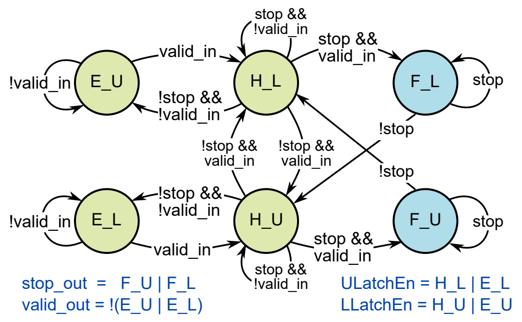


Fig. 3. The DET FSM that controls the operation of the DET EB.

while the output multiplexer should always point to the output of the opaque latch. On a stall, the multiplexer of the DET EB freezes to preserve the correct order of data arrival.

To transform a simple DET flip-flop into a DET EB, the latches and the output multiplexer must be controlled *independently* using a control FSM, as illustrated in Fig. 2(b). Each DET EB consists of an *upper* and a *lower* latch (just like the simple DET flip-flops of Fig. 2(a)), and the control DET FSM orchestrates the data flow through the EB, based on handshaking (valid/stop) signals.

The fact that the transitions happen on every clock edge imposes the need for a DET FSM that is able to react on every clock edge. A DET FSM is built as a traditional SET FSM, with the only difference being that the state flip-flops are implemented using DET flip-flops, thereby causing state transitions and output logic changes on both clock edges.

B. The DET FSM Controller: States and Transitions

The DET FSM that controls the operation of the DET EB is shown in Fig. 3. It consists of six states that are described by a two-letter symbol. The first letter indicates the number of data items that are stored in the DET EB, i.e., 'E' (Empty=0 items), 'H' (Half-full=1 item), or 'F' (Full=2 items). The second letter, 'U' (Upper), or 'L' (Lower), denotes the latch of the DET EB from which we can safely read data through the output multiplexer. For example, in the state H_U, the DET EB holds one data item that will be read out through the upper latch. In this case, the upper latch is necessarily in the opaque state. On the contrary, F_L indicates a full DET EB where both latches being opaque.The older data item lies in the lower latch (L symbol) implying the data order.

Initially, in an empty state, the buffer waits for valid upstream data. On such an event, the buffer writes the data in its currently transparent latch and transitions into a half-full state. In that half-full state, the multiplexer points to this newly captured data item providing it to the downstream channel. This scenario is highlighted by the transitions E_L→H_U and E_U→H_L. In the E_L state, invalid data are read out from the opaque lower latch while the upper latch is transparent and ready to capture valid data. If valid data are actually written, it should be read on the next clock edge from the upper latch, as depicted by the transition to H_U and not to H_L.

In a half-full state (H_U or H_L), without any downstream stall, the FSM will transition either between the H_U and H_L states, emulating the baseline DET pipelined operation shown in Fig. 2(a), or the FSM will move to an empty state.

The first step towards latency-insensitive operation is to implement a mechanism that can handle a stall (stop) signal coming from the downstream receiver. A stall in an empty state is actually ignored, since there are no valid data to be sent downstream and the buffer has space to store any possibly new incoming data.

In a half-full state, the DET EB the stored data item cannot be read out, since the downstream channel is stalled. The stall is assumed to propagate progressively in the previous DET EBs. Therefore, the upstream channel is not aware yet of the blocking, and may still have in-flight data. If this is the case, the DET FSM transitions from a half-full state to a full state, where both latches become opaque.

To preserve the correct data order (a strict requirement in latency-incensitive systems), the word that first encountered the stall must be served first, once the stall is removed. This is depicted by the transitions H_U→F_U and H_L→F_L. Even if a new data item has been written, the read direction has not changed, i.e., the second letter of the start and ending states of the FSM are the same in either case. On the contrary, if no valid incoming data are present, we need to stay in the same state to avoid storing a bubble in the DET EB that would appear as a valid data item.

In a full state, a stall retains the state irrespective of the other signals, due to lack of free space. When the stall de-asserts, the FSM switches to a half-full state, since the older data has been read out, and the respective latch is freed (i.e., it becomes transparent) to accept new incoming data. In parallel, the multiplexer inverts its selection and points to the newer data. This toggle of the multiplexer's selection is highlighted by the transitions F_U→H_L and F_L→H_U.

C. The DET FSM Controller: Output Logic

The DET FSM is responsible to drive the enable signals of the upper and lower latches (denoted, respectively, as 'ULatchEn' and 'LLatchEn'), the selection signal of the output multiplexer, and the two output handshake signals: (a) the signal *valid_out* of the downstream channel and (b) the signal *stop_out* of the upstream channel.

The values of all those signals are determined by the corresponding states of the DET FSM, as depicted in Fig. 3. For example, the output is valid, i.e., valid_out=1, when the DET EB is not empty. Similarly, the DET EB asserts the stop_out signal when its FSM is in a full state.

The upper latch is enabled (i.e., becomes transparent) when the DET EB is either half-full, or empty, and the output multiplexer is currently reading from the lower latch. The opposite condition is needed for the lower latch to be enabled. Similarly, the output multiplexer selects the upper latch in all 'U' states and the lower latch in all 'L' states.

A complete logic-level diagram of the proposed DET EB, including the datapath latches and the DET FSM, is shown in Fig. 4. The 6-state FSM is mapped to three DET state flip-flops, and the associated logic follows a delay-optimized organization: the stop_out and valid_out interface signals are driven directly from output registers, and the corresponding stop_in and valid_in signals are consumed as late as possible in the DET FSM. A similar optimization exists for the internal datapath select signals ULatchEn, LLatchEn, and Mux_sel.

IV. Case Study: A DET Network-on-Chip

The proposed simple DET flow-control rules and low-cost DET EBs that utilize merely two latches per buffer can be used

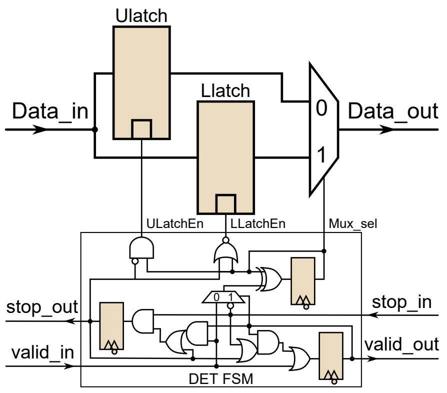


Fig. 4. The logic-level implementation of the proposed DET EB, using two latches in parallel and an FSM controller with DET flip-flops.

in the place of a SET EBs, which also need two data latches per buffer, to reduce the clock-tree power of the system. The amount of reaped savings solely depends on the percentage of registers used in the design, and the complexity of the resulting clock tree. In any case, operating the clock tree at *half the clock frequency of the original clock* under the same throughput is always advantageous, irrespective of the specific design scenario.

To test our approach, we implemented a wormhole Network-on-chip (NoC), where the input and output buffers of the routers of the NoC were replaced by the proposed DET EBs [5], [15]. Due to the simplicity of the flow-control rules, the proposed DET EBs can be placed in the physical layout either close to the router's switching logic, or they can be distributed across the NoC links to facilitate timing closure.

To enable DET clocking in all router components, all the remaining registers of the design – either in the control logic, or in the arbitration logic of the routers – were transformed into DET registers. Unfortunately, our implementation was impeded by a deficiency that afflicts the vast majority (to the best of our knowledge) of commercial standard-cell libraries: they do not contain any dual-edge-triggered flip-flops. Consequently, we were forced to *emulate* the DET flip-flop functionality by using one positive- and one negative-edge triggered flip-flop for each SET flip-flop of the original design. This transformation created an unnecessary overhead, since the DET routers introduced a larger clock-pin capacitance and led to more complex clock trees. Additionally, the output multiplexers of the emulated DET flip-flops led to an increase in the router's critical path. However, it should be stressed that these complications are *artifacts of the emulation* of DET flip-flops using SET ones, which was inevitable due to the lack of DET flip-flops in most standard-cell libraries.

Nevertheless, despite the above-mentioned limitations in the implementation flow [13], which limited our ability to perform a true "apples-to-apples" comparison, we were still able to observe *clock-tree power savings that reached 30%*, after testing different scenarios with respect to data width and the number of router input/output ports. In all examined cases, the SET routers were assumed to operate at 1 GHz, although they were able to reach a higher clock frequency at 45 nm/0.8 V. The DET NoC operated at 500 MHz to also cover any pessimism due to the additional clock jitter that needs to be taken into account under dual-edge clocking. Most importantly, both NoC implementations (SET and DET) exhibit equivalent cycle-by-cycle behavior during logic simulations.

V. Conclusions

Latency-insensitive design combines the modularity of asynchronous design with the efficiency of synchronous implementations, offering a correct-by-construction paradigm that can be applied at the early or the latest stages of the design, without any impact on the functionality of the system. On the other hand, DET clocking provides a concrete solution to the issues of frequency scaling and timing closure, by retaining the data throughput of SET clocking at half the clock frequency. In this work, we efficiently merge – *for the first time* – the benefits of both design principles, by proposing DET flow-control rules and low-cost DET EBs that yield scalable latency-sensitive systems with markedly reduced clock-tree power. The proposed DET EB design constitutes a promising *new primitive building block* that can facilitate DET clocking in latency-insensitive setups.

References

[1] L. Carloni and A. Sangiovanni-Vincentelli, "Coping with latency in soc design," *IEEE Micro*, vol. 22, no. 5, pp. 24–35, Oct. 2002.

[2] J. Cortadella, M. Kishinevsky, and B. Grundmann, "Synthesis of synchronous elastic architectures," in *Proceedings of the 43rd Annual Design Automation Conference*, 2006, pp. 657–662.

[3] H. M. Jacobson, P. N. Kudva, P. Bose, P. W. Cook, S. E. Schuster, E. G. Mercer, and C. J. Myers, "Synchronous interlocked pipelines," in *Proc. Int. Symp. on Asynchronous Circuits and Systems*, April 2002, pp. 3–12.

[4] M. Vijayaraghavan and Arvind, "Bounded dataflow networks and latency-insensitive circuits," in *IEEE/ACM Int. Conf. on Formal Methods and Models for Co-Design*, July 2009, pp. 171–180.

[5] G. Michelogiannakis, D. U. Becker, and W. J. Dally, "Evaluating elastic buffer and wormhole flow control," *IEEE Transactions on Computers*, vol. 60, no. 6, pp. 896–903, Jun. 2011.

[6] G. Dimitrakopoulos, I. Seitanidis, A. Psarras, K. Tsiouris, P. M. Mattheakis, and J. Cortadella, "Hardware primitives for the synthesis of multithreaded elastic systems," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014.

[7] L. P. Carloni, "From latency-insensitive design to communication-based system-level design," *Proc. of the IEEE*, vol. 103, no. 11, pp. 2133–2151, Nov 2015.

[8] M. Butts, A. M. Jones, and P. Wasson, "A Structural Object Programming Model, Architecture, Chip and Tools for Reconfigurable Computing," in *IEEE FCCM*, Apr. 2007, pp. 55–64.

[9] R. Panda and S. Hauck, "Dynamic Communication in a Coarse Grained Reconfigurable Array," in *IEEE Int. Symp. on Field-Programmable Custom Computing Machines*, 2011, pp. 25–28.

[10] Y. Huang and et al., "Elastic CGRAs," in *Proc. of the ACM/SIGDA Intern. Symp. on Field programmable gate arrays*, 2013, pp. 171–180.

[11] D. Papa, C. Alpert, C. Sze, Z. Li, N. Viswanathan, G.-J. Nam, and I. Markov, "Physical synthesis with clock-network optimization for large systems on chips," *IEEE Micro*, vol. 31, no. 4, pp. 51–62, Jul. 2011.

[12] N. Nedovic and V. G. Oklobdzija, "Dual-edge triggered storage elements and clocking strategy for low-power systems," *IEEE Trans. on VLSI Systems*, vol. 13, no. 5, pp. 577–590, May 2005.

[13] A. Bonetti, N. Preyss, A. Teman, and A. Burg, "Automated integration of dual-edge clocking for low-power operation in nanometer nodes," *ACM TODAES*, vol. 22, no. 4, pp. 62:1–62:20, May 2017.

[14] B. Cao, K. A. Ross, M. A. Kim, and S. A. Edwards, "Implementing latency-insensitive dataflow blocks," in *Proc. of Int. Conf. on Formal Methods and Models for Codesign*, 2015, pp. 179–187.

[15] G. Dimitrakopoulos, A. Psarras, and I. Seitanidis, *Microarchitecture of Network-on-Chip Routers: A designer's perspective*. Spinger, 2014.

[16] S. H. Unger, "Double-edge-triggered flip-flops," *IEEE Transactions on Computers*, vol. C-30, no. 6, pp. 447–451, June 1981.

[17] M. Pedram, Q. Wu, and X. Wu, "A new design of double edge triggered flip-flops," in *Proc. ASP-DAC*, Feb 1998, pp. 417–421.

[18] A. Bonetti, A. Teman, and A. Burg, "An overlap-contention free true-single-phase clock dual-edge-triggered flip-flop," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 2015, pp. 1850–1853.

[19] X. Lin, "On applying scan based structural test for designs with dual-edge triggered flip-flops," in *IEEE Int. Test Conference*, Oct 2017.