This article was downloaded by: [King Fahd University], [Muhammad Elrabaa]
On: 01 February 2012, At: 11:21
Publisher: Taylor & Francis
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK

International Journal of Electronics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tetn20

# A new FIFO for transferring data between two unrelated clock domains

Muhammad E.S. Elrabaa [a]
[a] Department of Computer Engineering, KFUPM, PO BOX 584, Dhahran 31261, Saudi Arabia

Available online: 01 Feb 2012

To cite this article: Muhammad E.S. Elrabaa (2012): A new FIFO for transferring data between two unrelated clock domains, International Journal of Electronics, DOI:10.1080/00207217.2011.651698

To link to this article: http://dx.doi.org/10.1080/00207217.2011.651698

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.tandfonline.com/page/terms-and-conditions

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

*International Journal of Electronics*
2012, 1–12, iFirst

Taylor & Francis
Taylor & Francis Group

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

# A new FIFO for transferring data between two unrelated clock domains

Muhammad E.S. Elrabaa*

*Department of Computer Engineering, KFUPM, PO BOX 584, Dhahran 31261, Saudi Arabia*

(*Received 17 April 2011; final version received 23 November 2011*)

A new simple-to-design FIFO that allows data transfer between two clock domains of unrelated frequencies has been developed. The fully synchronous interfaces significantly ease the system-on-chip integration process. With a relatively low gate count, the proposed FIFO allows the producer and consumer to put/get data at their respective frequencies (1 datum/clock cycle) till it gets filled, then the rates converge to the lower of the two frequencies. The maximum initial latency is three cycles of the consumer's clock. Several manifestations of the FIFO have been developed for different design cases including producer/consumer data width mismatch. Operation of the FIFO has been verified using both gate-level simulations and SPICE simulations with a 0.13 μm, 1.2 V technology. An 8-cell FIFO showed proper operation at producer and consumer clock frequencies of 2 and 3.125 GHz, respectively, with a data transfer rate of more than 2 giga datum/s and an average power of 721 μW.

**Keywords:** FIFO; on-chip communication; synchronous data transfer; system-on-chip; network-on-chip

## 1. Introduction

Currently, systems-on-chip (SoCs) are constructed using a wide range of pre-designed modules (intellectual properties or IPs) that are integrated together with a communication medium (typically a system bus). Each IP has its own clock and communication needs. This, coupled with the ever increasing demands on shorter time-to-market, necessitates developing efficient design flows that can achieve time closure of the whole SoC in short time while satisfying the communication needs of its various components. Many bus-based SoC design methodologies have been developed requiring either asynchronous or synchronous interfaces (Salminen, Lahtinen, Kuusilinna, and Hamalainen 2002). Due to the limitations of buses, Networks-on-Chip were proposed as scalable interconnections by Dally and Towles (2001) and later on by Henkel, Wolf, and Chakradhar (2004). Also, due to the difficulty of globally synchronising SoC components, another interconnection scheme has emerged; globally asynchronous locally synchronous (GALS) systems (Chapiro 1984) where asynchronous interconnects synchronous blocks.

GALS employ one of three communication schemes (Teehan, Greenstreet, and Lemieux 2007); pausible clocks, asynchronous and loosely synchronous. Pausible clock systems stop (or pause) the clock of a block during data transfers. With each additional input/output channel, the percentage of idle time would increase. This goes against the fundamental concept of decoupling 'computations' from 'communications' rendering this

*Email: elrabaa@kfupm.edu.sa

ISSN 0020–7217 print/ISSN 1362–3060 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/00207217.2011.651698
http://www.tandfonline.com

design style impractical. Dasgupta and Yakovlev (2007) have shown that this technique is not suited for interfacing large high-speed IP cores in SoCs. Fully asynchronous interconnects can adapt to a wide range of temperature, process and voltage variations, as well as varying data rates. As such, they stand to offer the highest degree of robustness and decoupling of different SoC design activities. The data transfer rates and latencies, however, are limited due to the required handshaking. Loosely synchronous techniques with dedicated point-to-point connections require some form of a FIFO between the sender and receiver to move data across their clock domains. Communication throughput and latency depends on the design of the FIFO, transmitter/receiver clock rates and communication patterns. An asynchronous FIFO, albeit being simple to design, would at most achieve a throughput of 1 datum/three clock cycles of the slower of the two clocks due to handshaking and synchronisation between the two domains (Teehan et al. 2007). In this study, a new data transfer interface (DTI) FIFO is proposed with the following key features:

(1) *Fully synchronous interfaces:* Both producer and consumer put and get data using their own clock with very simple synchronous interfaces, easing the SoC integration and timing closure without exposing the designers to asynchronous designs that most of them are not familiar with and is not compatible with most design flows.
(2) *Simplicity of design:* Can be synthesised in a standard cell-based flow.
(3) *Flexibility*: Can be adapted to different producer/consumer data widths or data rates.
(4) *Robustness*: Relatively independent of gates' sizing; the transfer latency adjusts automatically to gates' delays. Furthermore, full/empty conditions are detected and handled in a very simple manner.

The basic concept of the newly developed DTI FIFO is described in Section 2. The design and verification of the basic FIFO cell is introduced in Section 3 followed by the FIFO construction methodology for different use cases in Section 4. Performance evaluation of an 8-cell FIFO using SPICE simulations along with gate count calculations are presented in Section 5 followed by conclusions in Section 6.

## 2. Basic concept of the proposed DTI FIFO

Conventionally, synchronous FIFOs are usually designed as circular (or cyclic) buffers with two pointers pointing to the head and tail of the data within the buffer. The pointers' values are exchanged between the PUT and GET controllers to determine the FIFO status. This becomes complicated when the PUT and GET clocks are different and unrelated and may lead to a reduced throughput and increased latency. Several FIFO designs have been proposed to overcome this problem and allow data transfer between two different clock domains at maximum possible data rate. Chakraborty and Greenstreet (2003) proposed using complex training circuitry to estimate the frequency difference so that synchronisations are only limited to high-risk transfers. Seizovic (1994) proposed pipelining both data and synchronisation alongside one another reducing the synchronisation probability of failure and eliminating the need for detecting full/empty conditions. This not only increased latency with capacity, but the sender and receiver had to operate at the same data rate. Chelcea and Nowick (2004) proposed a FIFO that enables maximum data

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

transfer rates but requires elaborate circuitry to detect when the FIFO is empty, full, near empty and near full. Ono and Greenstreet (2009) proposed a very similar FIFO with standard cell implementation that increased the number of gates. Strano, Ludovici, and Bertozzi (2010) proposed a similar family of dual-clock FIFOs but used ring counters for the read/write pointers with asynchronous comparisons. The comparison result is then synchronised to each clock domain. A point-to-point bidirectional link based on an asynchronous FIFO was proposed by Chattopadhyay and Zilic (2005) with a minimum of three clock cycles (of the slower of the two clocks) transfer latency and limited capacity. A FIFO based on dual-port SRAM was proposed by Apperson, Yu, Meeuwsen, Mohsenin, and Baas (2007). Suited for large buffers, it uses Grey-coded pointers to limit the synchronisation to one-bit when they are exchanged between the write/read ports. Configurable logic is used to reserve space in the FIFO to compensate for the synchronisation latency and to control the skew of data and control signals on both sides, increasing both complexity and latency.

The proposed DTI FIFO is a latch-based circular buffer, but instead of having cells made of regular latches, it uses simple asynchronous pipelines as the basic cell. These pipelines (with a minimum of two latch stages) independently operate from one another (no signals are passed between different pipelines) with separate control, as shown in Figure 1. The total FIFO capacity in bits is:

$$\text{FIFO capacity} = n \times k \times m \tag{1}$$

where $n$ is the number of cells, $k$ the number of stages per pipeline and $m$ the data width. Data synchronously enter a pipeline from one clock domain (the producer's), asynchronously propagate through the pipeline and synchronously leaves from the other end to the second clock domain (the consumer's). So both the producer and the consumer see a fully synchronous interface. Synchronisation of the control signals is required at both ends of the pipeline, but by simultaneously overlapping putting/getting data to/from several cells, these delays are hidden and the put/get rates are maximised. Two modulo binary counters are used as PUT and TAKE pointers. Each pointer operates at its side's clock, and its

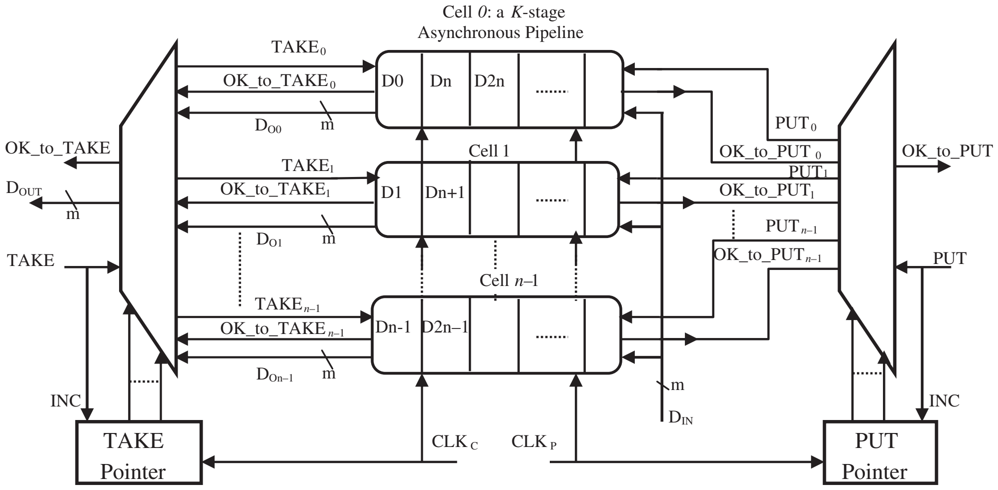


Figure 1. Block diagram of the proposed DTI FIFO structure.

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

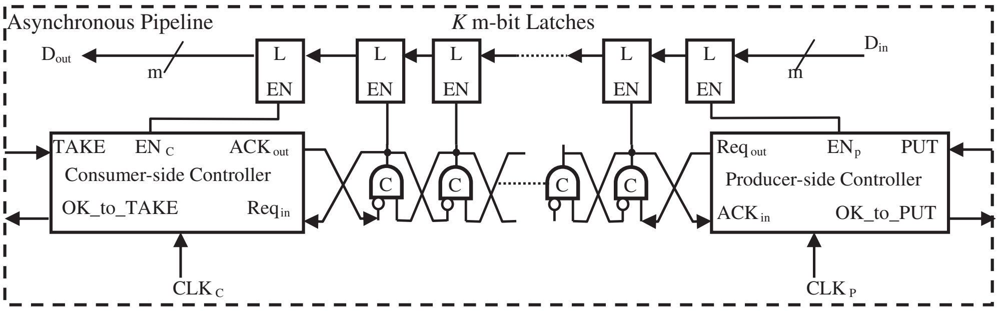


Figure 2. Structure of the asynchronous pipeline used as the basic DTI FIFO cell.

value is not passed to the other domain, since data are automatically taken in the same order that they were put. The producer can put a new data item whenever the OK_to_PUT signal is high and the consumer can take a data item out whenever the OK_to_TAKE signal is high. The PUT pointer is incremented after each put operation and so is the TAKE pointer after each take operation. No special circuits for detecting empty/full FIFO conditions are required. If data cannot be put to the cell currently selected by the PUT pointer, the external OK_to_PUT signal is not activated and the producer cannot put any new data. Similarly, if the cell currently selected by the TAKE pointer is not ready, the external OK_to_TAKE signal is not activated and the consumer cannot get any new data. The transfer latency through a cell depends on the producer/consumer clock frequencies and number of stages in the pipeline. If this latency is less than the clock period of the faster of the two clocks, then the FIFO's latency could be as low as two cycles of the slower clock.

The use of an asynchronous pipeline as a FIFO cell instead of a single latch, or dual-ported SRAM as in other FIFOs greatly simplifies the design by decoupling the PUT (to the producer's side latch) and GET (from the consumer's side latch) operations. It also gives another alternative for increasing the FIFO's capacity by increasing the cell capacity instead of increasing the number of cells.

## 3. Design of the asynchronous pipeline

### 3.1. *Basic structure*

The basic structure of the asynchronous pipeline is illustrated in Figure 2. Simple four-phase micropipeline latch control with Muller-C elements as in Furber and Day (1996) is adopted to simplify the timing constraints and thus the circuit design. This reduces the pipeline capacity since when the pipeline is full only *half* of the stages will contain data. Implementing full buffer stages would require pulsed latch control with stricter timing constraints, increasing the design complexity and reducing reliability.

### 3.2. *Signalling protocol*

The signalling protocol for transferring data through a pipeline is shown in Figure 3 (using signal names from Figure 2) for equal producer and consumer clock frequencies

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

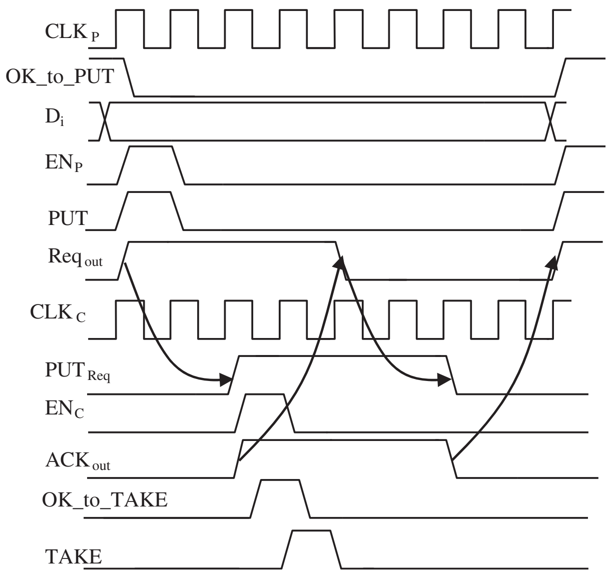


Figure 3. Signalling protocol for data transfer across an asynchronous pipeline. $PUT_{Req}$ is the $Req_{in}$ signal after synchronisation with the consumer's clock.

(worst case condition). The producer initiates a transfer (when OK_to_PUT is high) by setting up the data and asserting the PUT signal. The pipeline controllers transfer the data to the other side of the pipeline and set the OK_to_TAKE signal high to indicate a data item is ready to be taken. This signal is reset when the consumer removes the data, indicated by setting the TAKE signal high. The larger the difference in frequency between the producer and the consumer, the lesser is the number of cycles it takes to transfer data through the pipeline. This number could have been reduced by overlapping data transfers within the same pipeline at the expense of more complexity (more timing constraints to meet). Overlapping transfers in different pipelines, however, increases the throughput to one datum per cycle with latency as low as one cycle of the lower clock without any added complexity.

The $Req_{in}$ and $ACK_{in}$ signals are synchronised to the consumer and producer clocks, respectively, using two D-FF synchronisers. A two-FF synchroniser is a fairly safe method for signal synchronisation (Ginosar 2003) but may not provide a large enough mean time between failures (MTBF) at high clock rates. The MTBF could be increased by adding more synchronisation stages. Throughput can still be maintained by increasing the number of cells to hide the increased cell's latency.

## 3.3. *Controller design*

Figure 4 shows the design of the producer/consumer-side controllers. Each controller is a simple two-state FSM. SR latches are used to generate the OK_to_PUT and OK_to_TAKE signals that indicate the status of the pipeline (empty/full) to ensure that these signals remain constant when a pipeline is not selected and do not *evaporate*.

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

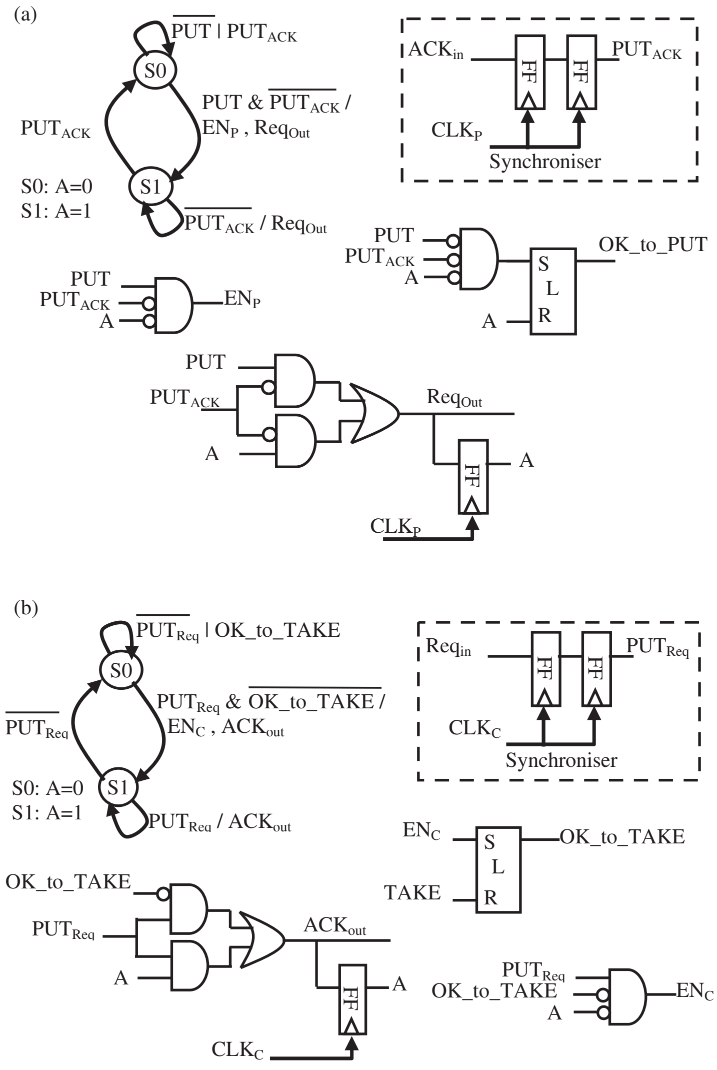


Figure 4. Design of the asynchronous pipeline controllers: (a) design of the producer-side controller and (b) design of the consumer-side controller.

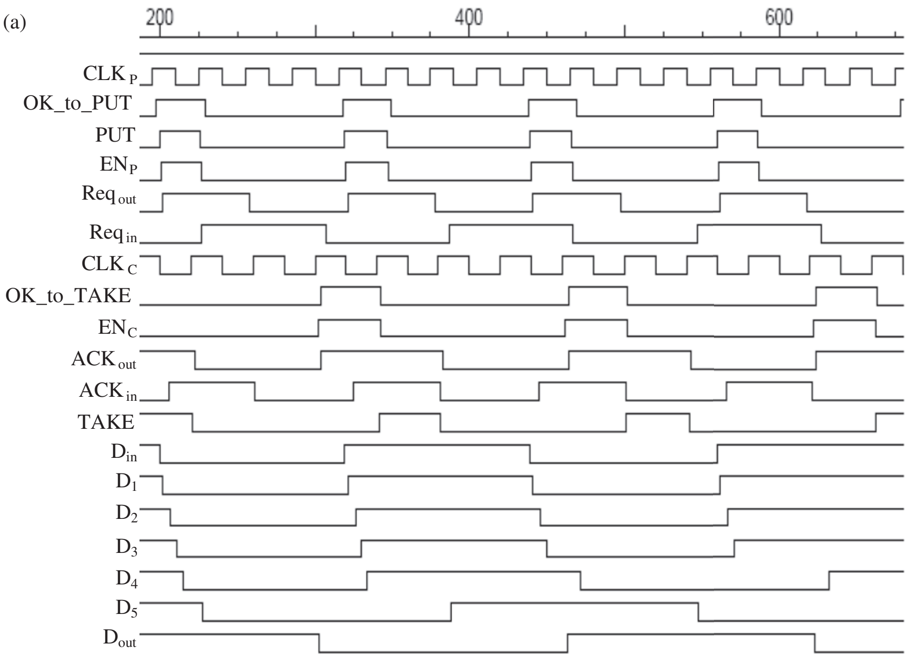
## 3.4. *Pipeline design verification*

To verify the functionality of the asynchronous pipeline, gate-level simulations of a six-stage, 1-bit wide pipeline was conducted with unit gate delays under different producer/consumer clock frequency ratios (Figure 5). When the rate mismatch is relatively small (25%), both producer and consumer are initially able to PUT/TAKE data every four cycles of their clocks with an initial latency of two cycles of the slower clock (Figure 5a). The PUT/TAKE rates eventually converge to 1 datum every four cycles of the slower clock. Figure 5(b) shows the waveforms for 5× frequency ratio. The pipeline gets filled after three puts, stalls till the consumer starts taking data and then the put operations

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

(a)



(b)

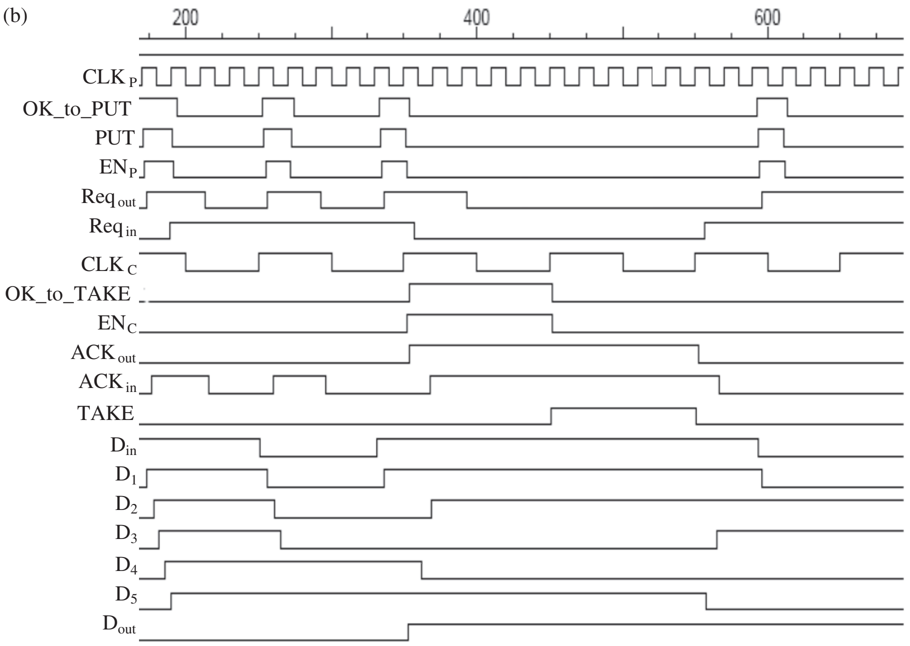


Figure 5. Gate-level simulation results for a six-stage asynchronous pipeline: (a) producer/consumer clock ratio = 1.25× and (b) producer/consumer clock ratio = 5×.

resume at a rate of 1 datum every four clock cycles of the consumers clock. For these results, the delay of a pipeline stage was less than the smallest clock period. If it was more, then the maximum PUT/TAKE rate would have dropped to 1 datum every five clock cycles or even less.

## 4. FIFO construction

There are many possible manifestations of the DTI FIFO architecture depending on the use case. The two design parameters number of cells and number of latch stages per cell determine the FIFO's design and performance. The objective would be to construct an FIFO with the required capacity that will allow the producer and consumer to put and take data at the maximum rate of 1 datum/cycle of their respective clocks. Assuming that the delay per asynchronous pipeline stage $T_{AS}$ is smaller than both the producer's and consumer's clock periods ($T_{CLKP}$ and $T_{CLKC}$, respectively) the initial latency (IL) of the FIFO is:

$$IL \leq T_{DMP} + T_{PCU} + k * T_{AS} + 3 * T_{CLKC} + T_{CCU} + T_{MC} \tag{2}$$

where $T_{DMP}$ and $T_{MC}$ are the delays through the PUT/Data demultiplexer/multiplexer on the producer/consumer sides, respectively, and $T_{PCU}$ and $T_{CCU}$ are the delays through the producer/consumer control units. It is assumed that the consumer will consume the data one clock cycle after the assertion of the OK_to_TAKE signal; hence, the $3*T_{CLKC}$ term above (a maximum of 2 $T_{CLKC}$ are needed for the $\text{Req}_{\text{in}}$ synchronisation). FIFO construction for several design cases is discussed below:

(1) ***Consumer limited case ($T_{CLKP} \leq T_{CLKC}$):*** The minimum number of cells in this case should be four to allow the maximum put rate. For equal frequencies, three latches would suffice for maximum PUT/TAKE rates. For greater frequency ratio, the FIFO would get filled after a number of puts that depends on the FIFO's total capacity *C*. The number of latch stages per cell could be increased to the point where a whole packet or burst of data is put before the FIFO gets filled.

(2) ***Producer limited case ($T_{CLKC} < T_{CLKP}$)*:** For this case, increasing the cell size beyond three latches is not beneficial. So using 8 cells with two latches per cell would ensure maximum put rate (1 datum/producer's cycle) for any frequency difference. The consumer, however, would eventually take data at the same put rate.

(3) ***Different producer/consumer data widths:*** This situation may arise when the different IPs being integrated are not compatible in terms of data widths. The construction of the FIFO for all possible data width mismatches is illustrated in Figure 6. Some of the details (e.g. muxing/demuxing) were omitted for clarity.

The case of integer data width ratio (*l*) is illustrated in Figure 6(a) and (b). When the producer has a larger data width, it simultaneously puts data to a group of *l* cells while the consumer would take the data from one cell at a time. When the producer has a smaller data width, it puts data to one cell at a time while the consumer takes data from the *l* group of cells. The general case of non-integer data width ratio is illustrated in Figure 6(c). Now, *l* represents the common divider for producer/consumer data widths (could be 1-bit). The producer's data width is $m \times l$, whereas the consumer's is $q \times l$. The producer now puts

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

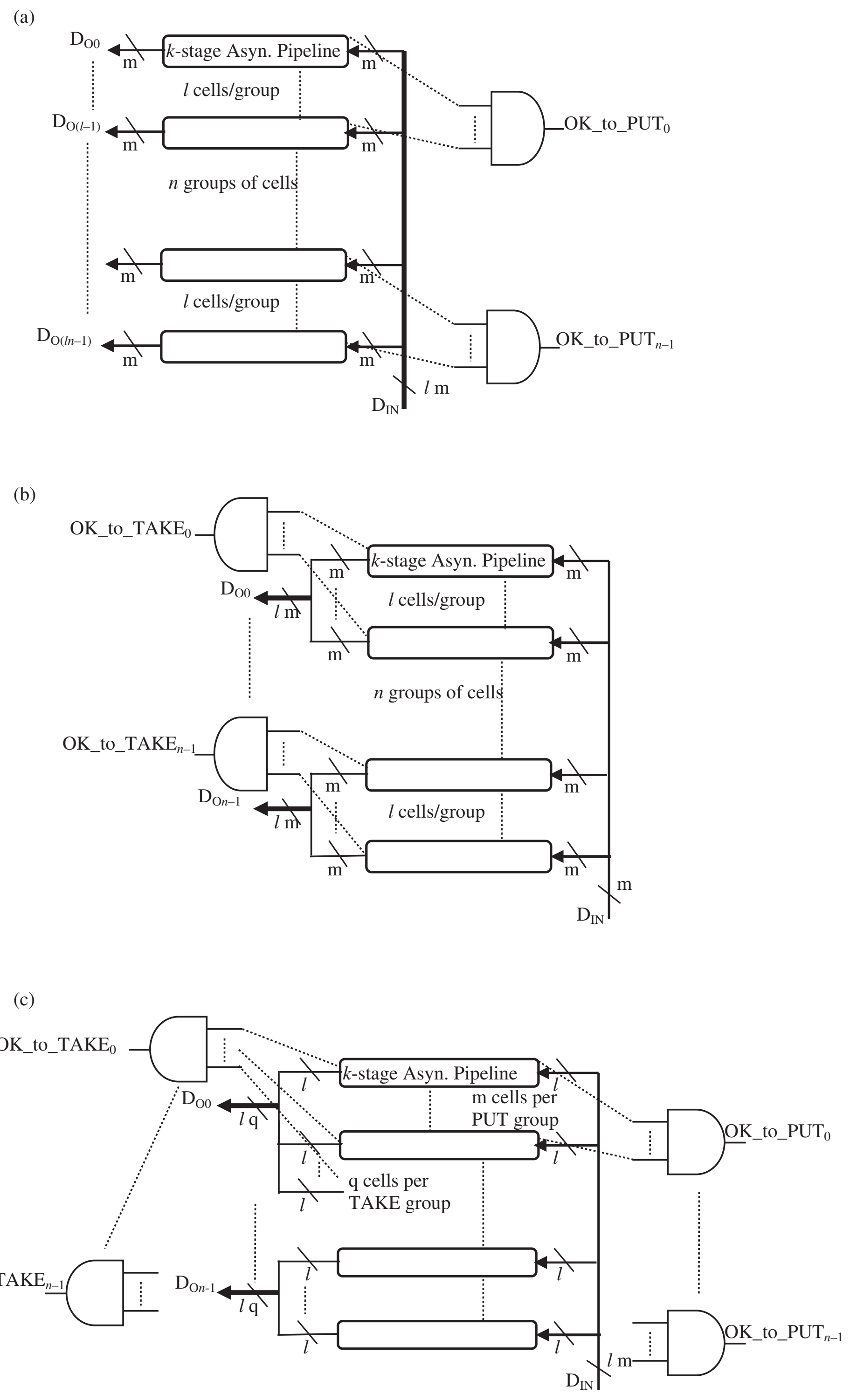


Figure 6. FIFO construction for non-equal producer and consumer data widths: (a) producer/consumer data width ratio = $l$; (b) consumer/producer data width ratio = $l$; and (c) general case; $l$ is the common divider for producer/consumer data widths.

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

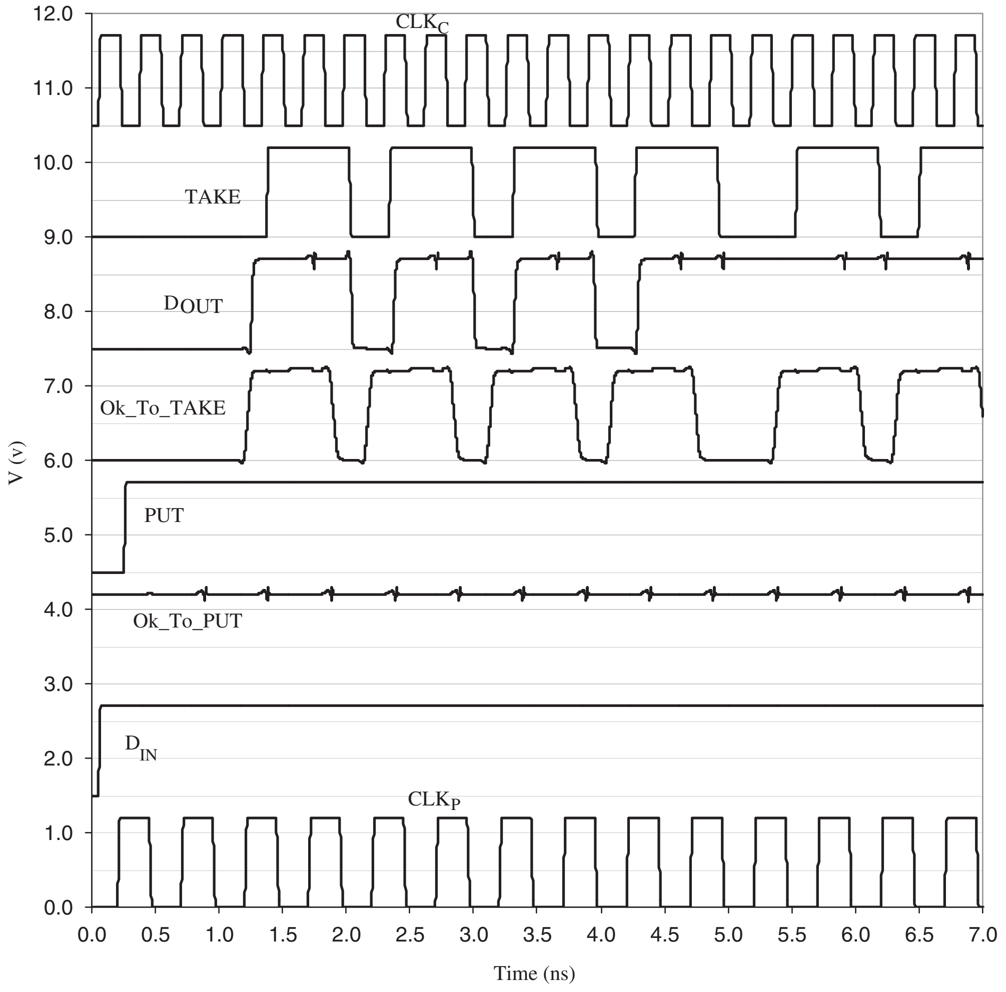


Figure 7. SPICE simulation waveforms of the new FIFO.

data to *m*-cells at a time and the consumer takes data from *q* cells at a time. The total FIFO capacity in this case is $l \times n \times \max(m, q)$ bits.

## 5. FIFO performance evaluation

An 8-cell, two-latches/cell FIFO with a datawidth of 8-bit was designed at the transistor level using a 0.13 μm, 1.2 V CMOS technology. Transistors were sized to achieve a producer and consumer clock rates of 2 and 3.125 GHz, respectively. SPICE simulation results (bit 0 of input/output data, OK_to_PUT and OK_to_TAKE signals) are shown in Figure 7 with waveforms being shifted along the vertical axis by various amounts for clarity. The input data were kept high and a datum was taken every time the OK_to_TAKE signal was high. The producer is able to PUT data every cycle

Table 1. Comparison between different FIFOs based on 8-cells/FIFO and 8-bit data width.

| FIFO | Number of gates[a] | Initial latency | Maximum throughput |
|---|---|---|---|
| This work | 1250 | One to two cycles | One data per slower clock cycle |
| Seizovic (1994) | ~300 | Eight cycles | One data per slower clock cycle[b] |
| Chelcea and Nowick (2004) | ~1500 | One to two cycles | One data per slower clock cycle |

Notes: [a]Estimated by the author based on logic diagrams in published work. [b]The circuit imposes the restriction that the producer and consumer must operate at the same data rate.

(the OK_to_PUT signal remained high during the whole simulation time), while the consumer was taking data at a rate of 2 datum every three cycles of its own clock, or an effective transfer rate of 2 GB/s. This is a remarkable performance at this technology node. The total gate count for this FIFO is ~1250 gates and the measured average power was 721 μW (~90 μW/cell), a very reasonable consumption.

Table 1 gives a comparison between the new FIFO and published FIFOs with similar capabilities based on 8-cell FIFOs with 8-bit data width. Though the FIFO reported by Seizovic (1994) has the lowest gate count, its latency is directly proportional to the FIFO size. It also restricts the producer and consumer to operate at the same data rate. This is not the case for the other FIFOs. The new FIFO achieves a performance comparable to the best reported FIFO with significantly fewer gate count and simpler design.

## 6. Conclusions

A new interface FIFO that can efficiently transfer data between two unrelated clock domains has been developed. With a relatively low gate count, it allows fully synchronous data communication between the two domains at the maximum rate of 1 datum per cycle of the lower frequency of the two no matter what is the frequency ratio between the two domains. Different manifestations of the new FIFO have been developed for various use cases including data width mismatch between communicating IPs. The correct operation of the developed circuits was verified with both gate- and transistor-level simulations. An 8-cell FIFO with a total gate count of 1250 gates operating at producer/consumer clocks of 2 and 3.125 GHz consumed an average power of 721 μW. Compared to other FIFOs with similar capabilities, the new FIFO is simpler to design due to the absence of timing constraints.

**Acknowledgement**

The author acknowledges the support provided by King Abdulaziz City for Science and Technology through the Science and Technology Unit at King Fahd University of Petroleum and Minerals (KFUPM) for funding this study through project no. 08-ELE-43-4 as part of the National Science, Technology and Innovation Plan.

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012

## References

Apperson, R.W., Yu, Z., Meeuwsen, M.J., Mohsenin, T., and Baas, B.M. (2007), 'A Scalable Dual-clock FIFO for Data Transfers Between Arbitrary and Haltable Clock Domains', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15, 1125–1134.

Chakraborty, A., and Greenstreet, M.R. (2003), 'Efficient Self-timed Interfaces for Crossing Clock Domains', in *Proceedings of 9th International Symposium on Asynchronous Circuits and Systems*, pp. 78–88.

Chapiro, D.M. (1984), 'Globally-asynchronous Locally-synchronous Systems', PhD thesis, Stanford University.

Chattopadhyay, A., and Zilic, Z. (2005), 'GALDS: A Complete Framework for Designing Multiclock ASICs and SoCs', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13, 641–654.

Chelcea, T., and Nowick, S. (2004), 'Robust Interfaces for Mixed Timing Systems', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12, 857–873.

Dally, W.J., and Towles, B. (2001), 'Route Packets, not Wires: On Chip Interconnection Networks', in *Proceedings of 38th Design Automation Conference*, pp. 684–689.

Dasgupta, S., and Yakovlev, A. (2007), 'Comparative Analysis of GALS Clocking Schemes', *IET Journal of Computers and Digital Techniques*, 1, 59–69.

Furber, S.B., and Day, P. (1996), 'Four-phase Micropipeline Latch Control Circuits', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4, 247–253.

Ginosar, R. (2003), 'Fourteen Ways to Fool Your Synchronizer', in *Proceedings of 9th International Symposium on Asynchronous Circuits and Systems*, pp. 1–8.

Henkel, J., Wolf, W., and Chakradhar, S. (2004), 'On-chip Networks: A Scalable, Communication-centric Embedded System Design Paradigm', in *Proceedings of 17th International Conference VLSI Design*, pp. 845–851.

Ono, T., and Greenstreet, M. (2009), 'A Modular Synchronizing FIFO for NoCs', in *Proceedings of 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pp. 224–233.

Salminen, E., Lahtinen, V., Kuusilinna, K., and Hamalainen, T. (2002), 'Overview of Bus-based System-on-chip Interconnections', in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 372–375.

Seizovic, J. (1994), 'Pipeline Synchronization', in *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 87–96.

Strano, A., Ludovici, D., and Bertozzi, D. (2010), 'A Library of Dual-clock FIFOs for Cost-effective and Flexible MPSoC Design', in *Proceedings of International Conference on Embedded Computer Systems (SAMOS)*, pp. 20–27.

Teehan, P., Greenstreet, M., and Lemieux, G. (2007), 'A Survey and Taxonomy of GALS Design Styles', *IEEE Design and Test of Computers*, 24, 418–428.

Downloaded by [King Fahd University], [Muhammad Elrabaa] at 11:21 01 February 2012