International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:8, No:3, 2014

A Novel FIFO Design for Data Transfer in Mixed Timing Systems

Mansi Jhamb, R. K. Sharma, A. K. Gupta

***Abstract*—**In the current scenario, with the increasing integration densities, most system-on-chip designs are partitioned into multiple clock domains. In this paper, an asynchronous FIFO (First-in First-out pipeline) design is employed as a data transfer interface between two independent clock domains. Since the clocks on the either sides of the FIFO run at a different speed, the task to ensure the correct data transmission through this FIFO is manually performed. Firstly an existing asynchronous FIFO design is discussed and simulated. Gate-level simulation results depicted the flaw in existing design. In order to solve this problem, a novel modified asynchronous FIFO design is proposed. The results obtained from proposed design are in perfect accordance with theoretical expectations. The proposed asynchronous FIFO design outperforms the existing design in terms of accuracy and speed. In order to evaluate the performance of the FIFO designs presented in this paper, the circuits were implemented in 0.24µ TSMC CMOS technology and simulated at 2.5V using HSpice (© Avant! Corporation). The layout design of the proposed FIFO is also presented.

***Keywords*—**Asynchronous, Clock, CMOS, C-element, FIFO, Globally Asynchronous Locally Synchronous (GALS), HSpice.

I. Introduction

THE conventional VLSI systems being synchronous in their construction have a global clock signal which acts as a common timing reference for the operation of entire chip circuitry. Contrary to this, completely asynchronous designs do not have any global timing [10]. The benefits of asynchronous circuits design come from many aspects, such as getting rid of the power consumption from clock, no need to do global clock tree synthesis, and potentially higher system throughput [1]. The most critical problem with asynchronous circuit design is the poor support from EDA tools. Since there is no global synchronous clock signal, it is very hard for EDA tools to check if the function of the circuit is exactly the same as designed. In most cases, asynchronous circuits are designed by engineers manually. Current VLSI systems are systems-on-a-chip (SoC) comprising of mixed clock domains [19]. These SoCs are developed using numerous pre-designed modules [intellectual properties or IPs] which are integrated with a communication medium. Each and every IP has its specific clock and communication requirements. Numerous bus-based SoC design methodologies, employing either synchronous or asynchronous interfaces have been proposed [9]. Due to the limitations of buses, Networks-on-Chip were proposed as scalable interconnections by [8], [7]. Globally asynchronous locally synchronous (GALS), a new paradigm emerged as a solution for difficulty in synchronization of SoC components [6]. In this paradigm, blocks are built using traditional synchronous design techniques, but these synchronous blocks do not share global timing information and are asynchronous with respect to each other. Globally- Asynchronous Locally Synchronous (GALS) offers to solve the problem of clock skew and delay in System-on-chip (SoC) design. One of the three communication schemes are employed by GALS; pausible clocks, asynchronous and loosely synchronous [5]. Pausible clock systems stop the clock of a block during data transfers. The pausible clocking control is a scheme to avoid synchronization failure by adjusting the local clock. Reference [11] has shown that this technique is not suitable for interfacing large high-speed IP cores in SoCs. The highest degree of robustness and adaptability to a wide range of temperature, process and voltage variations along with the varying data rates are the key attributes of fully asynchronous interconnects. However the data transfer rates and latencies are limited due to handshaking [22]. In Loosely synchronous techniques some form of a FIFO between the sender and receiver is required to move data across clock domains [20]. Communication throughput and latency depends on the design of the FIFO, transmitter/receiver clock rates and communication patterns. An asynchronous FIFO would at most achieve a throughput of 1 datum/three clock cycles of the slower of the two clocks due to handshaking and synchronization between the two domains [5]. While it is often convenient to divide a system into multiple subcomponents, it is unlikely that these components will operate autonomously. Accordingly, data transfer is required between local synchronous blocks. Performing this task reliably and efficiently are key challenges in GALS designs. One structure that is particularly well-suited for this task is the dual-clock first-input first-output (FIFO) or mixed-clock FIFO [3], [21]. The basic FIFO architecture must be modified to accommodate two independent clock inputs. Data passing through the FIFO module will enter with reference to one clock and exit with reference to the other clock. In this way, data can be passed safely between independent clock domains [3]. References [12], [13] have presented high-level views of dual-clock FIFO structures, but details of dual-clock FIFO designs are lacking in the literature. Fully asynchronous FIFOs often appear in the literature , but these designs do not utilize clocks, and therefore, are difficult to apply in cases of synchronizing data between clock domains [14], [15]. Despite

Mansi Jhamb is with the University School Of Information and Communication Technology, GGSIPU, Sector-16C, Dwarka, New Delhi, India (e-mail: mansi.jhamb@gmail.com)

Prof. R.K. Sharma and Prof. A.K. Gupta are with the Department of Electronics and Communication Engineering, NIT Kurukshetra, India (e-mail: mail2drrks@gmail.com, anilg699@rediffmail.com)

all these issues, in the GALS architecture, blocks in different clock domains communicate with each other using asynchronous connections [16]. Asynchronous FIFO is an important component for the efficient data transfer in asynchronous communication [3]. Hence the asynchronous FIFO design is necessary for implementing GALS structure in a SoC design [17], [4], [18]. Asynchronous FIFO using micropipeline is presented in [10]. The main characteristic of micropipeline FIFO is that the data will flow through all data cells in the FIFO before reaching the output port. Hence the delay due to data movement from input to output is unavoidable [4]. Reference [17] presented an asynchronous FIFO using counter as control logic, avoiding data movement at the cost of increased complexity. Reference [4] presented an asynchronous FIFO structures using token passing (sender/receiver can transmit/receive data to/from FIFO only when it has a token) and a common data bus for data in and out. This enables the data to be pushed or popped from asynchronous FIFO without data movement inside FIFO. Hence the latency caused in a micropipeline is eliminated resulting in power reduction [10]. Reference [2] has demonstrated globally asynchronous locally synchronous (GALS) clocking applied to a System-on-chip resulted in a design where each core is a synchronous block and communication between the cores is controlled by wrapper logic around the cores. Data is transferred between synchronous blocks through asynchronous communication channels which may be pipelined with self-timed FIFOs. Each channel has its own request and acknowledges handshake signals which accompany bundled data words. The synchronization strategies used for inter-core communication are a strong source of non-determinism, which causes a chip to randomly choose one of a set of possible correct responses to a given input stimulus. Non –determinism greatly complicates chip-level debug and test because these activities rely on the existence of a unique correct response with which an observed response is compared for error detection. Reference [2] has described a novel deterministic GALS methodology called "Synchro-tokens" whose parameterized wrappers are flexible enough to be used in wide range of applications. In this paper an efficient architecture of an asynchronous FIFO employed as a data transfer interface in GALS design, is proposed. This paper is organized as follows. In Section II, the structure of existing asynchronous FIFO is presented in detail. In Section III, a three stage four bits FIFO is simulated. The gate level simulation results depict a problem in the existing FIFO design. Thereafter the corrections are suggested and a new asynchronous FIFO design is proposed. The behavior of proposed FIFO is then simulated and the results for the maximum clock frequencies at which sender and receiver blocks can be clocked are determined. The conclusion is drawn in Section IV.

II. FIFO Architecture

This paper evaluates the schematic-level design of the data channels shown in Fig. 1. Letter "T" signifies output data port (FIFO Tail) and "H" corresponds to input data port (FIFO Head). This asynchronous FIFO supports asynchronous communication between the either sides. The details of architecture are presented as follows.

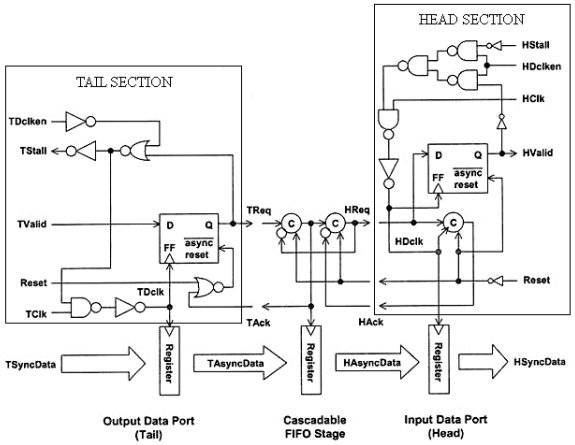

Fig. 1 The design of Asynchronous FIFO for GALS

A. C-Element as Stage-Controller

C-element is the key component of the design and is responsible for the handshake signals between different FIFO stages as shown in Fig. 2. In this FIFO design, C-element ensures correct asynchronous transmission of data between the two ends.

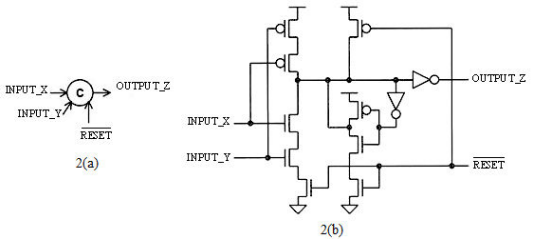

Fig. 2 The C-element with an active low reset signal

In the FIFO stage, two C-elements can be connected together as a stage controller. The handshake flow of a stage controller is shown as Fig. 3.

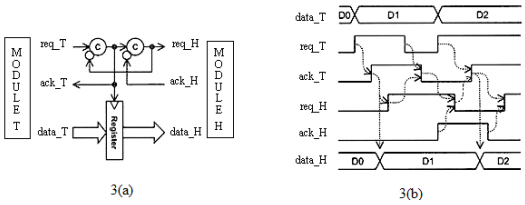

Fig. 3 The Phenomenon of Handshaking in stage controller

The process of Handshaking is explained as following steps:

Step 1. Initially all four handshake signals are low (Reset).

Step 2. The module T sends out data, in the meantime pulls req_T high.

Step 3. Since req_H is low,ack_T will be high. The leading

edge of ack_T is also the clock signal for data register, thus data_T is loaded to the stage register.

Step 4. The ack_T signal is also the input of the next C-element, ack_T when asserted high pulls req_H high.

Step 5. When req_T goes low, ack_T will also become low.

Step 6. The second C-element waits for the ack_H signal from next stage, when ack_H is asserted high, it conveys that next stage is ready to get the data in register. Then with low ack_T ,req_H will be asserted high.

B. The Data Transmission in Asynchronous FIFO

Data is transferred between synchronous blocks through asynchronous communication channels which are pipelined by self-timed FIFO as shown in Fig. 1. Each channel has its own request and acknowledges handshake signals accompanying bundled data words. The entire process of asynchronous data transmission is as follows:

When TDclken is asserted high, data in the tail begins to input to the buffer stage by sending Treq to the buffer. In case the buffer is empty, it will receive data from the head register and pulls up TAck. TReq will not be low until TAck is high. If the buffer is full, it can only receive data when the data in its only register has been sent to the next stage buffer. Before that, TAck will always be low. With more stages of buffer, more data could be sent to buffer before the signal TStall is pull up. This signal tells the data sending block to stop because buffer is FULL.

In the head side, data can be received when HDclken is asserted high and HStall is low. As shown in Fig. 1, there is another C-element in the head section. This C-element controls reception of data in the head section. One of the inputs in this C-element is HDclk, which is controlled by Hclk, HDclken and HStall. The leadingedge of HDclk leads to reception of data from the buffer register, and HValid becomes high when HReq is asserted high. Data in the head register can be read out when HValid is high.

The C-elements permit movement of data from one clock domain to another domain through this FIFO, only when the registers in the FIFO are not full. And in case the FIFO is full, it signals the block connected to the FIFO to stop sending new data. On the other hand reception of the data happens only when the block connected to the FIFO is ready to receive it.

III. Simulations and the Proposed Design

In this design of asynchronous FIFO, the clock in the tail and head parts of the FIFO can be totally different, which is not possible in synchronous CMOS design. The clock signals in this FIFO are not directly sent to flip-flops and registers. There are several control signals to gate clocks from outside.Due to the asynchronous clocks and gated clocks, it is not possible for EDA tools to do timing check and clock tree synthesis in asynchronous system design. Thus manually custom design for asynchronous circuit is performed. In order to evaluate the performance of the FIFO designs presented in this paper, the circuits were implemented in 0.24µ TSMC CMOS technology and simulated at 2.5V using HSpice (© Avant! Corporation). In this paper the behavior of three stage four bits FIFO is simulated. The width of this FIFO is 4 bits, thus four registers are needed for every stage of data registers. Special care has been taken to make the simulations realistic .This section first presents the gate-level simulation results of the existing design shown in Fig. 1 and depicts the problem incurred in the simulations due to the fault prevailing in the existing design. Thereafter the corrections are suggested in the existing FIFO design and a new design for asynchronous FIFO is proposed. Simulation results are then obtained for proposed FIFO structure. The results thus obtained are in perfect accordance with theoretical expectations. In order to determine the maximum frequency for this proposed FIFO, its behavior is simulated at different clock periods of the Tail and Head sections. The sizes of all transistors in this design are carefully selected so that the circuit can have correct functionality and high performance. For the performance of this circuit to be as high as possible, shortest critical path is achieved. Finally the layout is obtained for the new FIFO architecture.

A. Fault Detection in Existing Design

The overall schematic of the existing asynchronous FIFO is shown in Fig. 4.

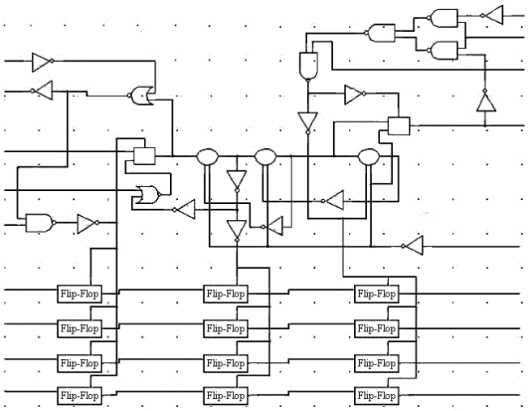

Fig. 4 Asynchronous FIFO schematic

In the schematic simulation, the rise and fall time of all inputs is considered to be 50ps, clock period for tail and head is selected to be 5ns and 3ns respectively. The HSpice simulation results are shown as Fig. 5.

While using the FIFO architecture shown in Fig. 4 the simulation results exhibit some problems. Fig. 5 shows there is a problem with the Tstall signal. This signal should be high only when either the next stage register is full or TDclken is high. In the simulation trace when the next stage register is not full and TDclken is low, Tstall signal has several glitches (these places are highlighted with circles). This problem leads to the incomplete clock pulse of TDclk, as shown encircled.

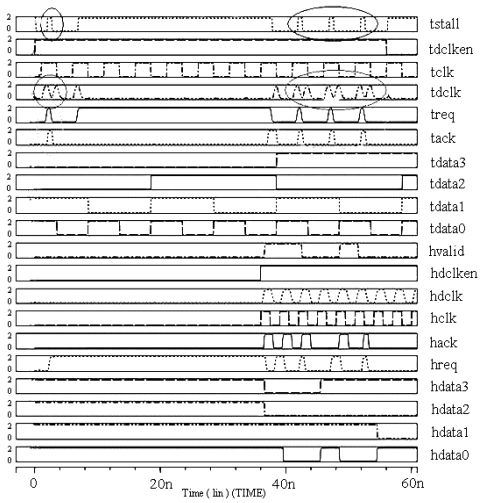

Fig. 5 The gate-level simulation results for Original FIFO

The problem arises because the pulses of Tack signal lead to pulses of TReq signal as shown encircled in Fig. 5. And the pulses of TReq signal are passed through NAND and INV to TStall signal which causes pulses in TStall signal .In an attempt to fix this problem, a modified tail design is proposed in Fig. 6. The proposed design involves addition of a latch after the NAND gate which is connected to TReq signal in the existing design. This latch passes TReq signal only when Tclk signal is low. This latch prevents passage of TReq signal to TStallsignal when clock is high, thus eliminating the glitches in Tstall signal.

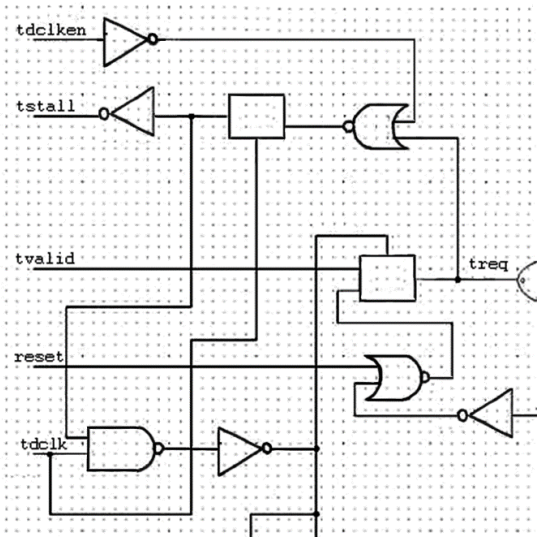

Fig. 6 The Proposed Tail Design

The Hspice simulation results for the proposed FIFO are shown in Fig. 7. Fig. 7 demonstrates that an addition of latch in the existing FIFO architecture, eliminates the glitches in Tstall signal.

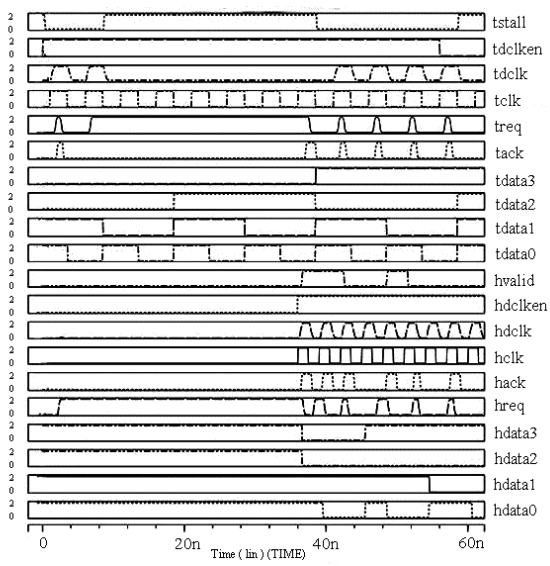

Fig. 7 The gate-level simulation results of proposed FIFO

B. Performance Evaluation of Proposed FIFO Design

In order to determine the maximum frequency this FIFO can operate at, the simulations are performed at different clock periods for head and tail sections of proposed FIFO architecture. The maximum clock speeds of the tail and head section are tested separately. Firstly the clock period of tail is reduced till the results become erroneous.

Case I: Tail Clock = 1900 ps and Head Clock = 800 ps

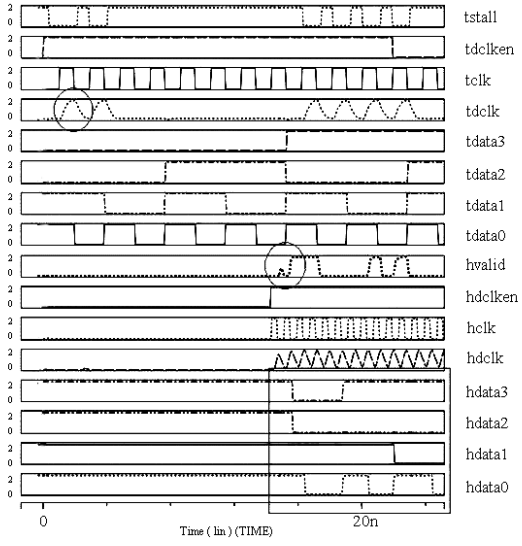

Fig. 8 The Simulation results for Tail Clock = 1900 ps and Head Clock = 800 ps

Fig. 8 shows that the rise and fall slopes of TDclk signal are getting mild and the Hvalid signal begins to show glitch at this clock speed. However, the results are still correct as shown in the box.

Case II: Tail Clock = 1800 ps and Head Clock = 800 ps

The simulation result for this case is shown in Fig. 9.

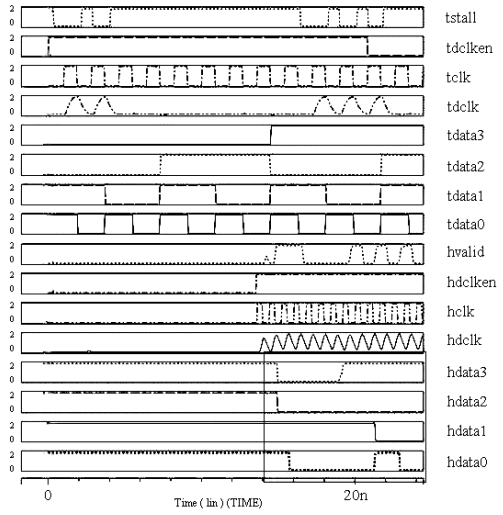

Fig. 9 The Simulation results for Tail Clock = 1800 ps and Head Clock = 800 ps

As shown in Fig. 9, the outputs in the box are deviated from the expected results shown in Fig. 8. One 4-bit data is missing. This is due to the reason that the speed of the tail clock is so high that the tail part can't finish passing data of the previous clock before the arrival of the next clock edge. Thus for the tail part, the minimum clock period for correct data transmission is 1900ps.

Case III: Tail clock = 1900 ps and Head clock = 700 ps

If the tail clock is kept at 1900ps and head clock is reduced to 700ps, the simulation results are shown in Fig. 10.

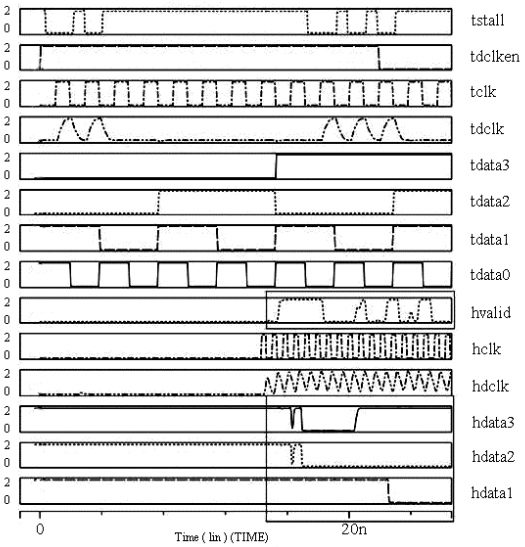

Fig. 10 The Simulation results for Tail clock = 1900 ps and Head clock = 700 ps

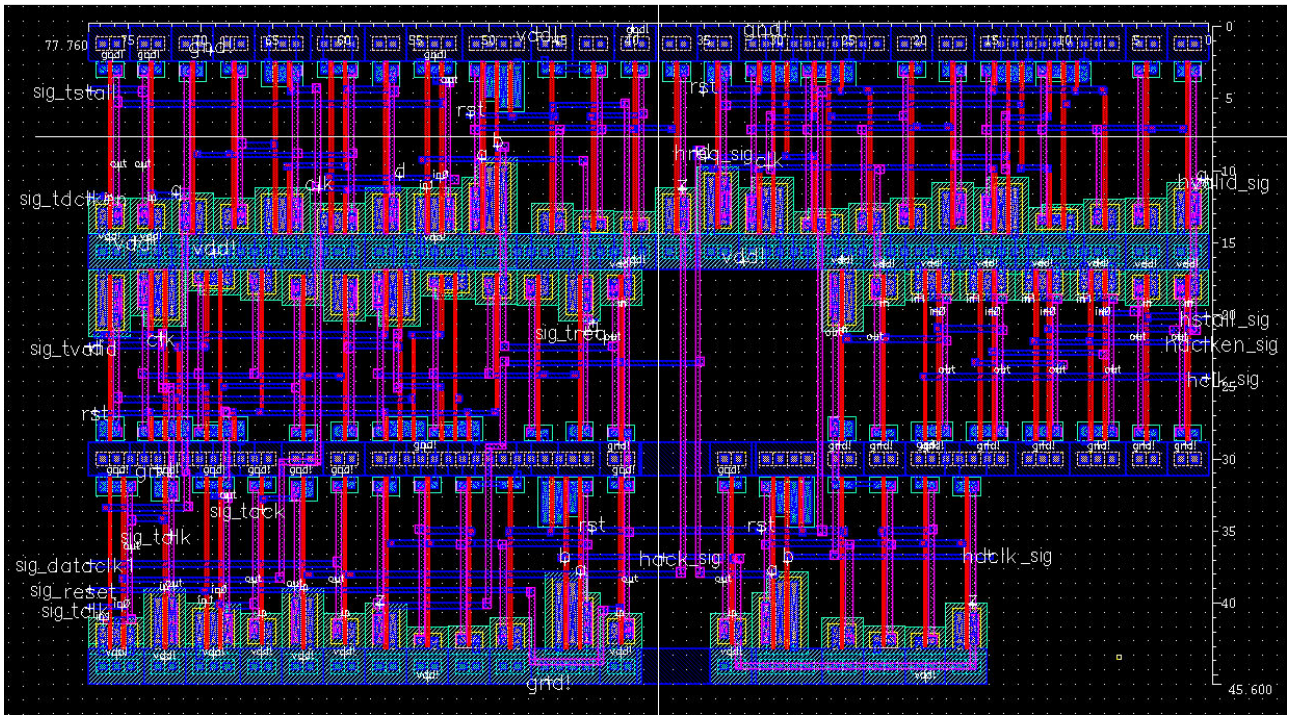

Fig. 11 The Layout Design for Asynchronous FIFO

As shown in Fig. 10, the results become erroneous when the head clock period is decreased to 700ps. And the glitches in Hvalid signal are becoming severe. Thus for the head part, the minimum clock period for which data can be correctly received is about 800ps.

The maximum clock frequencies at which Head and Tail section can be clocked are presented in Table I.

TABLE I
MAXIMUM CLOCKING FREQUENCIES FOR NEW FIFO

Version	Head Section	Tail Section
Mixed Clocks	1.25 GHz	0.52 GHz

C. Layout Design

The layout design of the proposed asynchronous FIFO is shown in Fig. 11. The height of the layout is 46um and the width is 74um. The ratio is 74/46=1.6.

IV. CONCLUSION

This work presents a novel and an efficient self-timed FIFO design which acts as a data transfer interface between the blocks with unrelated clock speeds. The proposed design is based on the idea of token passing. Our solution is developed from the flaw depicted in existing FIFO design [2]. The proposed design does not need any data synchronization and is able to correctly interface the blocks operating at different clock speeds. The critical part in this work is the design of C-elements, which handle pairs of handshake signals between different clock domains. The C-elements ensure the correct passage of data from one clock domain to another domain through this self-timed FIFO only when the registers in the FIFO are not full. In case the FIFO is full, the block connected to the FIFO is directed to stop the sending of new data. On the other hand data reception happens only when the block connected to the FIFO is ready to receive it. The results obtained from proposed design are in perfect accordance with theoretical expectations. The proposed asynchronous FIFO design outperforms the existing design [2] in terms of accuracy and speed. The result is quite promising and leaves the possibilities for further improvements open, leading to more area and power efficient implementation.

This mixed – clock design can also be adapted for interfacing asynchronous and synchronous subsystems.

REFERENCES

[1] Jens Sparso, "Principles of Asynchronous Circuit Design: A Systems Perspective," *Kluwer Academic Publishers*, 2002
[2] Matthew W. Heath, Wayne P. Burleson, Ian G. Harris, "Synchro-Tokens: A Deterministic GALS Methodology for Chip-Level Debug and Test," *IEEE transactions on Computers*, December 2005 (vol. 54 no. 12)
[3] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Bass, "A Scalable Dual-Clock FIFO for Data Transfers Between Arbitrary and Haltable Clock Domains," *IEEE Transactions on Very Large Scale Integration*, vol. 15, no. 10, pp. 1125–1134, Oct 2007.
[4] Chelcea, T., and Nowick,S.(2004), 'Robust Interfaces for Mixed Timing Systems',*IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12, 857–873.
[5] 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.
[6] Chapiro, D.M. (1984), 'Globally-asynchronous Locally-synchronous Systems', PhD thesis, *Stanford University*.
[7] Henkel, J., Wolf, W., and Chakradhar, S. (2004), 'On-chip Networks: A Scalable, Communication-centric Embedded System Design 'Proceedings of *17th International Conference VLSI Design*, pp. 845–851.
[8] 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
[9] 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.
[10] I.E Sutherland, "Micropipelines," *Communications of the ACM*, Volume 32 Issue 6, June 1989.
[11] Dasgupta, S., and Yakovlev, A. (2007), 'Comparative Analysis of GALS Clocking Schemes', *IET Journal of Computers and Digital Techniques*, 1, 59–69
[12] W. J. Dally and J. W. Poulton, Digital Systems Engineering.Cambridge, U.K.: *Cambridge Univ. Press*, 1998.
[13] M. Balch, Complete Digital Design, 1st ed.New York: *McGraw-Hill*2003.
[14] J. Ebergen, "Squaring the FIFO in GasP," in Proc. *Int. Symp. Asynch.Circuits Syst*., 2001, pp. 194–205.
[15] C. E. Molnar, I. W. Jones, W. S. Coates, and J. K. Lexau, "A FIFO ring performance experiment," in Proc. *Int. Symp. Asynch. Circuits Syst*.,1997, pp. 279–289.
[16] Xin Wang, TapaniAhonen, JariNurmi, "A Synthesizable RTL Design of Asynchronous FIFO," Proc. *International Symposium on System-on-Chip*, 2004.
[17] A.V. Yakovlev, A.M. Koelmans, L.Lavagno, "High-Level Modeling and Design of Asynchronous Interface Logic," *IEEE Design and Test of Computers*, Spring 1995.
[18] E. Brunvand, "Low Latency Self-Timed Flow through FIFOs," in 16th *Conference on Advanced Research inVLSI*, UC Santa Cruz, March 1995, pp. 76–90.
[19] 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.
[20] Para 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.
[21] 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.
[22] 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.