Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Adaptive-delay based reconfigurable asynchronous pipeline

Adnan Ghafoor [1, *], Arbab A. Khan [2]

[1]*Department of Electrical Engineering, International Islamic University, Islamabad, Pakistan*
[2]*Department of Electrical Engineering, Capital University of Science and Technology, Islamabad, Pakistan*

ARTICLE INFO

Article history:
Received 27 November 2017
Received in revised form
29 February 2018
Accepted 3 March 2018

Keywords:
Asynchronous
Adaptive delay
Pipeline
FPGA

ABSTRACT

This paper presents the asynchronous pipeline model implementable on FPGA platforms. The proposed event-controlled register acts as true adaptive delay element which adaptively prolongs the process of latching of data to store only the valid results, unlike other asynchronous approaches. Bundle data strategy with two-phase handshake protocol is used. In order to ensure the validity of the proposed pipeline, a fourth-order FIR filter was implemented on Xc7a100t-1csg324 FPGA. It was observed that the asynchronous pipeline implemented using auto place and route tools, adapts the delays of the data path and exhibits smooth functionality, with throughput supremacy over its synchronous counterpart.

1. Introduction

Escalation in clock skews due to down-scaling of technology compels the researchers to reconsider asynchronous design methodology (Choy et al., 2001). In asynchronous systems, subsystems work at their own pace using their local clock and synchronization among them is achieved by using either two-phase or four-phase handshake protocols (Spars and Furber, 2002). Handshake protocols are implemented using Request (Req) and Acknowledge (Ack) signals (Peeters and van Berkel, 1995). These signals form the control path which must match the speed of data being processed in data path (Beerel, 2002).

Data processing by multilayered logic blocks consume more time than control signals. Data may reach the subsystem earlier than Req. However, arrival of Req earlier than data may be catastrophic for the whole system. An earlier approach to solve the problem was to insert fixed delays in the Req line using delay pads which usually contain even number of inverters. Insertion of fixed delays is equivalent to slowing down the synchronizing clock of the synchronous systems; this excludes those from the domain of systems with adaptive delay.

The auto optimization tools during implementation remove the consecutive inversions; hence the delay pads in control path. Special parameters need to be passed to the CAD tools in order to preserve them in a design. Insertion of predefined delays does not guarantee the smooth functionality after the implementation of a design. This is because various place-and-route (PAR) tools during the routing process do not preserve the ratios between the control path and various fork legs of data path. The interconnect delays may become more significant than data delays and thus cannot be ignored (Hauck, 1995).

2. Background

Asynchronous systems exist in full custom domain like ASICs and are rarely implemented in FPGAs due to hardness of speed matching between control and data paths. This is because conventional FPGAs and their CAD tools are made for synchronous designs. This lack of support makes it impractical to precisely model logic and wire delays before implementation. Unbundled data strategy, where dual wire channel represents single logic level (Peeters and van Berkel, 1995), caters to the routing problem; however, it impacts speed, power and resource utilization.

High performance asynchronous pipelines including Micropipelines (Sutherland, 1989) and MOUSETRAP (Singh and Nowick, 2001) use predefined delays in their control path, whereas PSO pipelines (Williams, 1991) uses unbundled data strategy to cater to delays. Although a lot of quality work has already been presented in the literature for the implementation of asynchronous systems, the adaptive delay based implementation caught attention of the few. Completion detection, as an alternate approach towards adaptive delay, on bundle data through current sensing was originally

* Corresponding Author.
Email Address: adnan.phdee39@iiu.edu.pk (A. Ghafoor)
https://doi.org/10.21833/ijaas.2018.05.005
2313-626X/© 2018 The Authors. Published by IASE.

proposed by (Izosimov et al., 1990). The idea was implemented by (Dean et al., 1994) and further carried out by (Lampinen and Vainio, 2001; Bashirullah, 2006; Nigussie et al., 2011). Current sensing circuits, however, neither exists nor can be designed in FPGAs. The problem leads to the need of a logical Adaptive Delay Element (ADE) which dynamically adapts the delays of a design and the underlying technology.

Here, we propose an ADE that generates the control signals in correlation with the processing and flow of data, with adaptive delay needed at the instant that not only protects the system from such catastrophes but also improves the throughput of the system. The core idea behind the ADE is to wait for the stable result from the logic processing element after reception of the Req signal. The ADE controls the inter-stage latch which is the destination of partial results in a pipeline. The combination of ADE and inter-stage latch is named here as Event Controlled Register (ECR).

3. Event controlled register

Fig. 1 shows the schematic of ECR. The active-high Enable signal to the N-bit latch makes it transparent to data with a delay time of one latch, while new data keeps latching in it, the comparator keeps comparing it. The comparator's output goes high only when there are no more transitions in the data. The high output of the comparator guarantees that the result becomes stable and valid thus can be stored in the N-bit latch. When the valid result is present in ECR, the logical AND of the Enable signal and the output of the comparator goes to the 1-bit D-Latch. This matches the state of its output as Ack to the new state of the Req. The new value of Ack reaches as Ack to the Req initiating stage, that its result has been saved. The Ack also reaches as Req signal to the next stage of the pipeline asking it to store new result from its logic processing block.

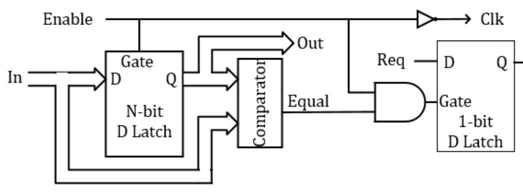

Fig. 1: Event controlled register

Clock (Clk) signal is also derived by inverting the Enable line, it meets the setup and hold time of sequential elements. Although the ECR does not contain any synchronous element, it generates clock signal to drive any sequential element in the design if needed.

4. ECR based asynchronous pipeline

Fig. 2 shows the model of a three-stage asynchronous pipeline using ADE, wherein the ECR serves as inter-stage latch. Delay pads in the control path are not required in this pipeline design because of the adaptive delay nature of the ECR that also enables the PAR tools to implement design in their way.

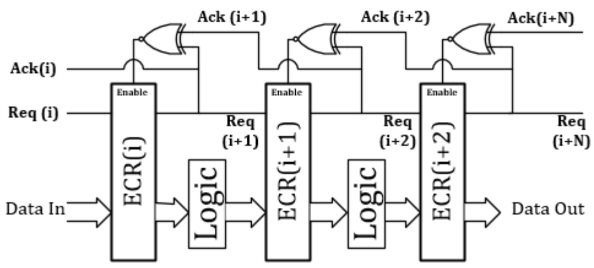

Fig. 2: ECR based asynchronous pipeline

Starting with the initial state when both Req and Ack are at same logic state, let it be logical 'low', in turn the output of all the XNOR gates will be 'high' and thus all the ECRs of the pipeline will be enabled. After giving new data at the input of the first ECR and flipping the state of its Req to 'high', the output of the comparator will remain 'low' till transitions in the data at the input of its ECR vanish and valid result is present in it. Then the 'high' output of the comparator will save the new value of the Req in its 1-bit latch, which is opposite to the value of Ack of the next stage, making the output of the first XNOR 'low'. This latches the data in its ECR so that the second stage may process it. After the processing when input and the output busses of the second stage ECR match, its 1-bit latch will flip to 'high' that will disable the ECR of the second stage and will also enable the ECRs of the first stage allowing it to take new data at the input of the pipeline if there is a Req. This way even and odd number stages work alternately. The process continues till the processed data emerges from the last stage. This way a single data presented to the pipeline will flip the phase of its all 1-bit latches, while the processing of second data will restore it. This signaling protocol seems matching to that of MOUSETRAP (Singh and Nowick, 2001) pipeline, but the major distinction is that MOUSETRAP uses fixed delay pads, thus it is not in the domain of adaptive delay architecture, whereas the ECR based pipeline has adaptive delay. Further it is implementable easily on FPGA platforms without disabling its optimizing tools.

The pipeline can communicate with the other asynchronous modules through its handshake signals. In order to make it a standalone system, Req signal of the first stage and Ack to the last stage are generated from inside logic. Inverted Ack from second stage goes as Req to the first stage and logical XNOR of Req and Ack of the last stage forms the Enable signal of that stage, however, the rest of the logic remains the same.

5. FIR filter

In order to check the validity of the presented concepts, the fourth order Gaussian low pass filter with 8-bit unsigned coefficient is designed as shown in Fig. 3. An 8-bit counter as sample generator feeds samples to the filter. Fig. 4 shows the post layout simulation of FIR filter that confirms the smooth functionality of the design and an average processing of data in less than 3 ns.

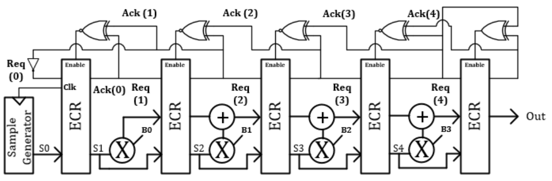

Fig. 3: FIR filter using ECR based pipeline

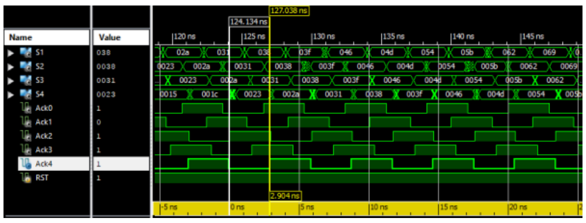

Fig. 4: Post layout simulation of FIR filter

6. Results and discussion

The design was implemented on Xc7a100t-1csg324 FPGA device using ISE 14.7 with auto PAR tools. No predefined delays were inserted in the Req line. Timing constraints were specified to optimize the routing process. Optimization goal was set to speed and effort level to high. XPower tool was used to estimate power consumption of the design based on the switching rate activity file obtained from the ISim tool. Both the synchronous and asynchronous designs were implemented using same parameters. The power and speed comparisons of both the designs are shown in Table 1. Speed improvement of asynchronous design is due to data dependent delays by completion detection circuitry (Beerel, 2002). Asynchronous implementation consumed more power than its synchronous counterpart. This difference is due to extra logic gates that were packed in logic slices and more switching activity due to the high processing speed.

Table 1: Speed and power comparison of synchronous and asynchronous implementations

FPGA Family	Design Methodology	Cycle Time (ns)	Power (mW)
Xc7a100t-1csg324	Synchronous	3.1	84
	Asynchronous	2.9	93

7. Conclusion

This paper demonstrates the concept of delay in the implementation of asynchronous pipeline over reconfigurable devices using auto PAR tools. Worst case delay in Req line is replaced with ADE in combination with inter-stage latch named as ECR. The ECR prolongs the latching process till the input data becomes stable. ECR based designs work on data dependent delays in contrast to predefined worst delays, therefore, they exhibit supremacy in speed over the synchronous and asynchronous designs with predefined fixed delays in addition to being technology independence.

References

Bashirullah R (2006). Reduced delay sensitivity to process induced variability in current sensing interconnects. Electronics Letters, 42(9): 531-532.

Beerel PA (2002). Asynchronous circuits: An increasingly practical design solution. In the International Symposium on Quality Electronic Design, IEEE, San Jose, CA, USA: 367-372. https://doi.org/10.1109/ISQED.2002.996774

Choy CS, Butas J, Povazanic J, and Chan CF (2001). A new control circuit for asynchronous micropipelines. IEEE Transactions on Computers, 50(9): 992-997.

Dean ME, Dill DL, and Horowitz M (1994). Self-timed logic using current-sensing completion detection (CSCD). In: Meng TH and Malik S (Eds.), Asynchronous circuit design for VLSI signal processing: 7-16. Springer, Boston, USA.

Hauck S (1995). Asynchronous design methodologies: An overview. Proceedings of the IEEE, 83(1): 69-93.

Izosimov OA, Shagurin II, and Tsylyov VV (1990). Physical approach to CMOS module self-timing. Electronics Letters, 26(22): 835-836.

Lampinen H and Vainio O (2001). Dynamically biased current sensor for current-sensing completion detection. In the IEEE International Symposium on Circuits and Systems, IEEE, Sydney, NSW, Australia, 4: 394-397. https://doi.org/10.1109/ISCAS.2001.922256

Nigussie E, Tuuna S, Plosila J, Liljeberg P, Isoaho J, and Tenhunen H (2011). Boosting performance of self-timed delay-insensitive bit parallel on-chip interconnects. IET Circuits, Devices and Systems, 5(6): 505-517.

Peeters A and van Berkel K (1995). Single-rail handshake circuits. In the 2nd Working Conference on Asynchronous Design Methodologies, IEEE, London, UK: 53-62. https://doi.org/10.1109/WCADM.1995.514642

Singh M and Nowick SM (2001). MOUSETRAP: Ultra-high-speed transition-signaling asynchronous pipelines. In the International Conference on Computer Design, IEEE, Austin, USA: 9-17. https://doi.org/10.1109/ICCD.2001.954997

Spars J and Furber S (2002). Principles asynchronous circuit design. Kluwer Academic Publishers, Norwell, USA.

Sutherland IE (1989). Micropipelines. Communications of the ACM, 32(6): 720-738.

Williams TE (1991). Self-timed rings and their application to division. Ph.D. Dissertation, Stanford University, Stanford, USA.