Lightening Asynchronous Pipeline Controller with Reusable Delay Path Synthesis on FPGA and LT-Spice

Dr.A.Ranganayakulu [(1)] Mr.D.Satyanarayana [(2)] Arunala Sai Yasaswini [(3)] Anna Sai Sree [(4)] Kasinaboina Srikasi [(5)] M. Venkata Poorna Chandrika[(6)]

[1,2] *Krishna Chaithanya Institute Of Technology & Sciences, Ece Department, Markapur, Andhra Pradesh.*

[3,4,5,6,] *Krishna Chaithanya Institute Of Technology & Sciences, UG Student-ECE, Markapur, Andhra Pradesh.*

Abstract In this project, we tackle the synthesis problem of two-phase bundled-data synchronous pipeline controllers in this work, where buffer insertion is necessary to ensure proper handshaking operation at each pipeline stage, but at significant area increase cost. By introducing a new notion in logic synthesis, delay path sharing and reusing, we may drastically minimize the amount of expensive delay buffers, hence lightening the pipeline controllers. Specifically, we first provide a method for synthesizing an asynchronous pipeline controller that allows setup timing pathways on pipeline stages to share delay buffers, hence minimally allocating total delay buffers. Furthermore, by expanding the suggested delay path sharing concept, we create an area-efficient delay circuit structure known as a delay path unit (DPU) and provide a detailed synthesis sequence of an asynchronous pipeline controller employing DPUs. Our methods of synthesizing asynchronous pipeline controllers can reduce the controller area by up to 46.3%–59.4% and the leakage power by up to 33.0%–49.0% on average while maintaining the same level of performance, according to experiments conducted with benchmark circuits using a LT-spice Simulator.

Keywords: LAPC(Lightening Asynchronous Pipeline Controller),6G Communications, FPGA, Verilog HDL,Xilinx Tool.

I.INTRODUCTION

An asynchronous circuit is one of the attractive alternatives to the synchronous circuit design style. Contrary to the synchronization mechanism used in a global clock network in synchronous circuits, asynchronous circuits exploit handshaking protocol for the communication between circuit components, by which they consume less dynamic power and operate at a higher frequency than their synchronous counterparts in general [1], [2]. One of the barriers to the adoption of an asynchronous handshaking mechanism is a large area of a handshaking controller. One notable method to lower the barrier is employing handshaking controller templates [3]–[5], and those adapted to pipeline structure are widely used for high-performance applications [6] in particular.

For example, Sutherland and Fairbanks [7] customized their dynamic logic and inserted it for fast data transmission, and Singh and Nowick [8] proposed the pipeline controller template with full capacity storage. Later, Fant and Brandt [9], Martin and Nyström [10], and Xia et al. [11] suggested the methods of employing dynamic logic with dual outputs. However, their solutions require substantial care or experience to fit into industrial design flows. On the other hand, Sutherland [12] employed his custom latch (i.e., capture-pass) and a C-element in his pipeline

controller template, and Singh and Nowick [13] devised MOUSETRAP, a high-performance pipeline controller template using a normally transparent latch. Recently, on top of [13], Ho and Chang [14] achieved a significant reduction in dynamic power consumption in datapath by blocking glitches in data transmission using a normally opaque latch. Toan et al. [15] slightly ameliorated its energy efficiency and performance using a C-element, but the underlying operation is identical to that in [14].

Meanwhile, delay variation among manufactured chips has grown a lot over the past years due to operation at low Vdd and enlargement of process variations. Fig. 1(a) shows the drastic increase of delay variation caused by Vdd scaling for 28-nm process technology, and the curves in Fig. 1(b) indicate that the maximum operating frequency can be increased by more than 10× when the temperature changes from 0 ◦C to 75 ◦C for the subthreshold voltage (sub-Vth) regime [16]. Various kinds of post-silicon tuning techniques have been proposed and used to contract this wide performance gap through adjusting delay tunable components on individual chips based on their physical properties. One representative method is clock skew tuning, which resolves timing violations by inserting delay circuits such as adjustable delay buffers (ADBs) to control local clock skews. One of the examples is a capacitor bank based ADB implementation [17], and ARM recently used a long delay chain of ring oscillators called tunable delay stages (TDSs) for tracking temperature variation in the system [16].

For a bundled-data protocol-based asynchronous circuit, a receiver should not accept a request signal before arriving data signals from its sender. To ensure this, most of the previous studies focused on the optimization and logical operation of pipeline controller templates with the assumption of using these post-silicon tunable delay circuits [4], [14]. As a result, they required many delays buffers to intentionally provide a long delay path on each pipeline stage, causing a considerable area overhead. Fig. 2 shows the change of the whole controller and delay buffer area as the required delay for satisfying handshaking communication increases.

Lightening a pipeline controller directly impacts two domains: 1) mitigating the increase of controller area and 2) reducing leakage power consumption. In this work, we target employing a new delay circuit structure as well as resynthesizing the conventional state-of-the-art pipeline controller in [14] to achieve the two factors in Domains 1 and 2 while retaining all the benefits (e.g., a considerable saving in glitch power) reaped from the controller in [14]. Our work can be summarized as follows.

1) Delay Path Sharing: We propose a technique of synthesizing an asynchronous pipeline controller in a way to share delay buffers among setup timing paths on pipeline stages so that the total delay buffers should be minimally allocated, formulating the allocation problem into linear programming (LP).

2) Delay Path Reusing: By extending the delay path sharing concept, we devise a new area-efficient delay circuit structure called delay path unit (DPU) and propose an in-depth synthesis flow of an asynchronous pipeline controller using DPUs.

2. BACKGROUND AND RELATED WORK: LITERATURE SURVEY

A. Bundled-Data Versus Delay-Insensitive Asynchronous Circuits

Bundled-data encoding is a coding style that transmits each data bit using exactly one signal wire, for which a handshaking mechanism through request and acknowledgment signal wires between two components should be installed,. A bundled-data asynchronous circuit consists of a datapath, which is the same structure as a synchronous circuit, and a controller. It is particularly attractive since it has relatively less area overhead over its delay-insensitive counterpart. However, since the delays of handshaking signals (req, ack) control all the timings of datapath operation, sufficient timing margins should be allotted to endure variation. Contrary to the bundled-data encoding, the delay-insensitive encoding is a technique that entails a data value and its validity simultaneously by using multiple wires for every data bit. Thus, it is robust to timing variation with no timing margins. However, it incurs a substantial area overhead because of multiple signal wires and detection logic for checking the completion of logic operations.

B. Two-Phase Versus Four-Phase Bundled-Data Protocol

A transaction of two-phase bundled-data protocol starts with issuing data and making a transition on a request signal by a sender. When the receiver accepts this signal, it starts to read the transaction data and finishes the transaction by making a transition on its acknowledgment signal. On the other hand, a four-phase bundled-data protocol operates as follows. First, a sender issues data and sets the request signal to 1. Then, the receiver detects this signal and begins to read data while setting the acknowledgment signal to 1. The sender accepts this acknowledgment signal and then initializes the request signal to 0, and finally, the receiver follows it by setting the acknowledgment signal to 0, completing the transaction.

II.EXISTING SYSTEM AND PROPOSED SYSTEM

1.EXISTING SYSTEM:

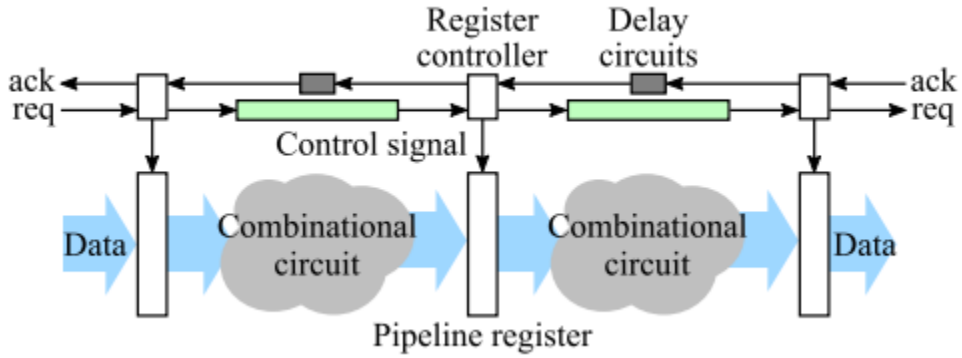

FIG.1. Structure of a bundled-data asynchronous pipeline circuit. A delay circuit, e.g., a delay buffer chain, should be inserted on each pipeline stage (the long green and short gray bars) to build up the setup and hold timing paths.

An asynchronous pipeline controller, also known as a handshake or asynchronous control system, is a type of control mechanism used in digital systems where data flows through a pipeline in a stage-wise manner, and the progression of stages is controlled without a centralized

clock signal. Unlike synchronous systems that use a global clock to synchronize operations, asynchronous systems rely on handshaking signals to manage data flow between pipeline stages.
Components and Characteristics of an Asynchronous Pipeline Controller:
Handshaking Signals:
Asynchronous control relies on handshaking signals exchanged between adjacent pipeline stages. Common signals include "request" and "acknowledge" signals.
Request and Acknowledge Protocol:
The request-acknowledge protocol is used to signal when a stage is ready to accept new data or when it has completed processing the current data. It ensures that data is transferred only when both the source and destination stages are ready.
Data Validity:
Asynchronous pipelines often use a "valid" signal to indicate the validity of the data at each stage. The data is transferred only when the valid signal is asserted.
Null Cycles:
Null cycles can be introduced in asynchronous pipelines, where a stage remains idle until it receives a valid input. This dynamic nature allows stages to operate independently and progress as soon as the necessary conditions are met.
Flexibility and Adaptability:
Asynchronous pipelines can adapt to variations in processing times of different stages, making them more flexible than synchronous counterparts in certain scenarios.
Data-Driven Operation:
The operation of an asynchronous pipeline is often data-driven, meaning that each stage processes data as soon as it is available, rather than being synchronized by a global clock.
Lack of Centralized Clock:
Asynchronous pipelines do not rely on a central clock signal. Instead, each stage operates independently based on handshaking signals and local conditions.

2.PROPOSED SYSTEM

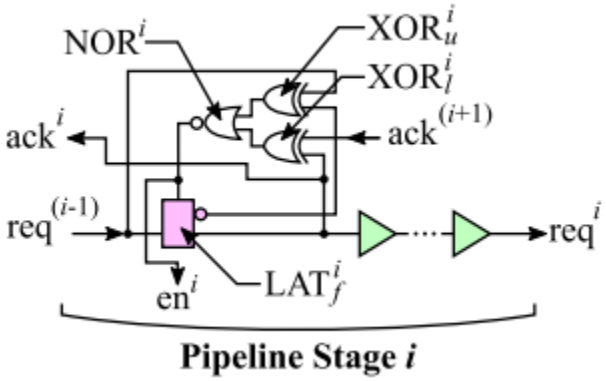

Fig2.Proposed Asynchronous pipeline controller.

2.1.State-of-the-Art Pipeline Controller Template

Fig. 2 shows a part of the state-of-the-art asynchronous pipeline controller template in [14] to be installed on each pipeline stage. The structure consists of two XOR gates (XORi u and XORi l), one NOR gate (NORi), and a resettable transparent latch (pink box, LATi f), all of which are connected to support the communication protocol on that pipeline stage. Procedures of the

transactions are as follows; let us assume that all request and acknowledgment signals are 0 initially. Then, XORi u and XORi l are 1 and 0, respectively, and thus, NORi is 0, which makes LATi f close. As req(i−1) becomes 1 at the latch input, XORi u goes to 0, causing NORi to 1, which makes LATi f transparent. After that, XORi l becomes 1, causing NORi to 0, which makes LATi f close again. This short time interval (i.e., the sum of NORi and XORi l delays) during which LATi f is transparent, thus reducing glitches, is the most significant advantage of this controller.1 After a relatively long time through the delay circuit, the request signal reqi of logic value 1 goes to the controller on pipeline stage i +1 and comes back as the acknowledgment signal ack(i+1) , which completes the transaction on pipeline stage i. Then, it initiates the next event on pipeline stage i with the opposite polarities, i.e., the request and acknowledgment signals of logic value 1.

III. RESULTS AND ANALYSIS DISCUSSION

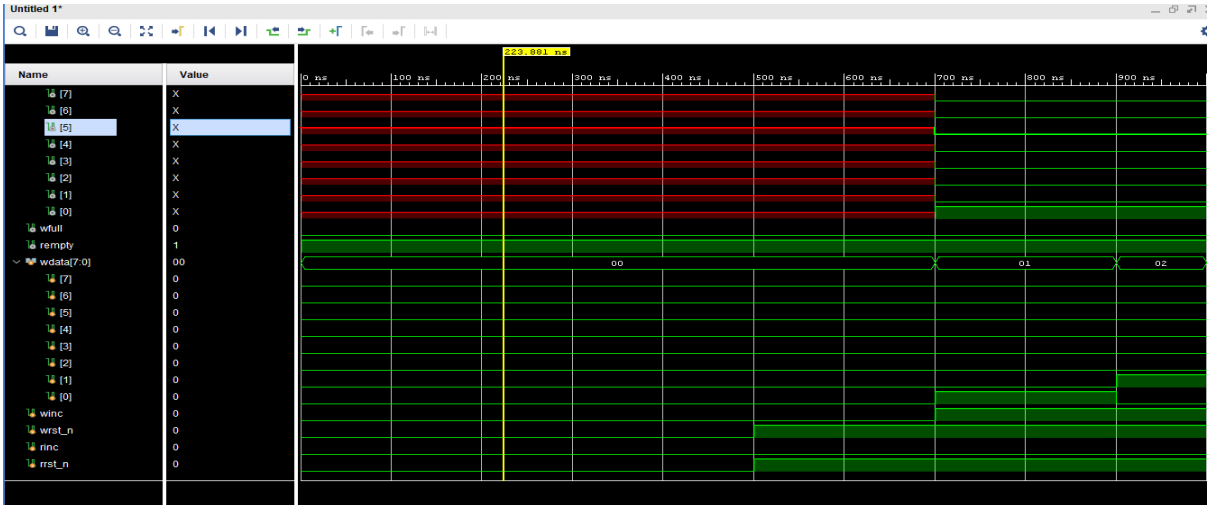

FIG.1.SIMULATION OUTPUT.

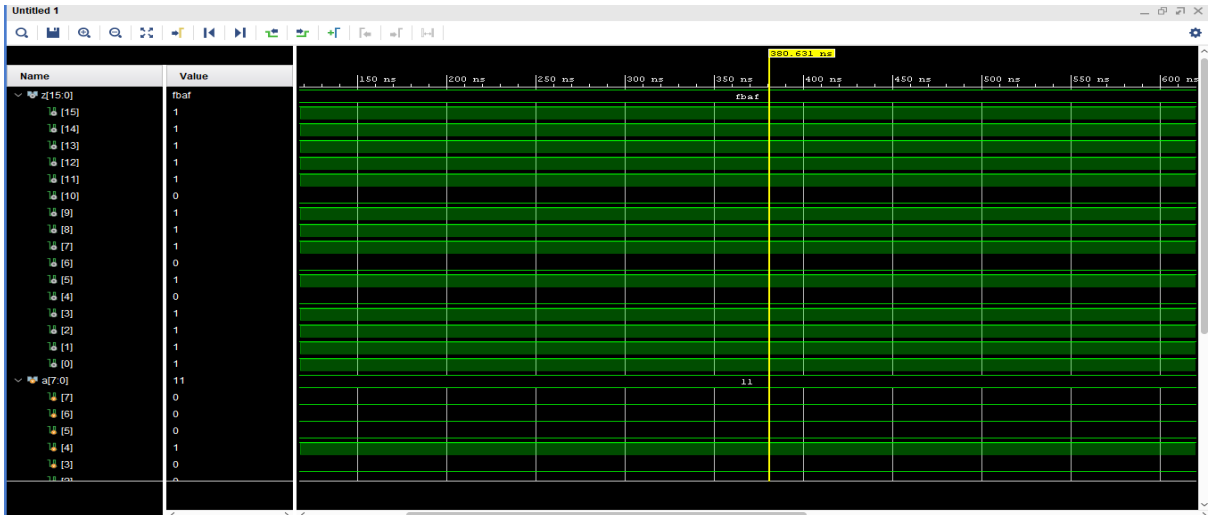

FIG2.SIMULATION OUTPUT

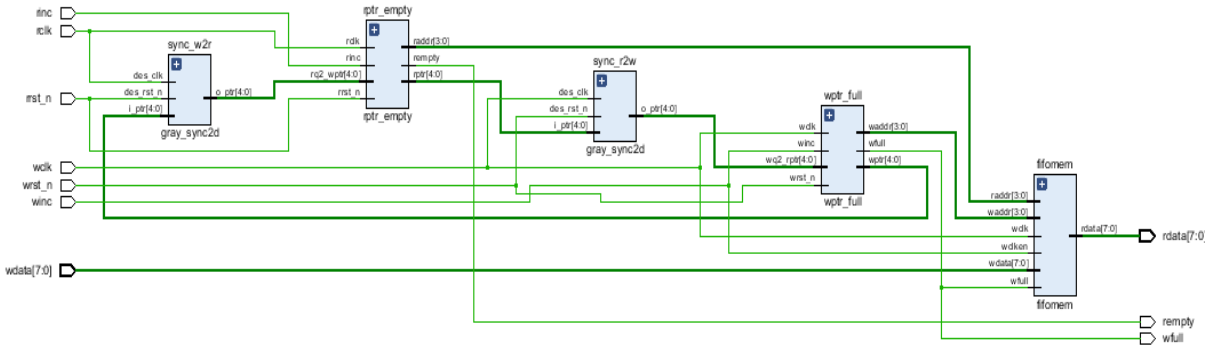

FIG3.RTL SCHEMATIC DIAGRAM

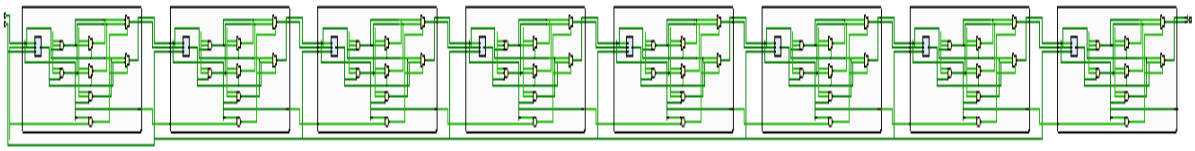

FIG4. RTL SCHEMATIC DIAGRAM

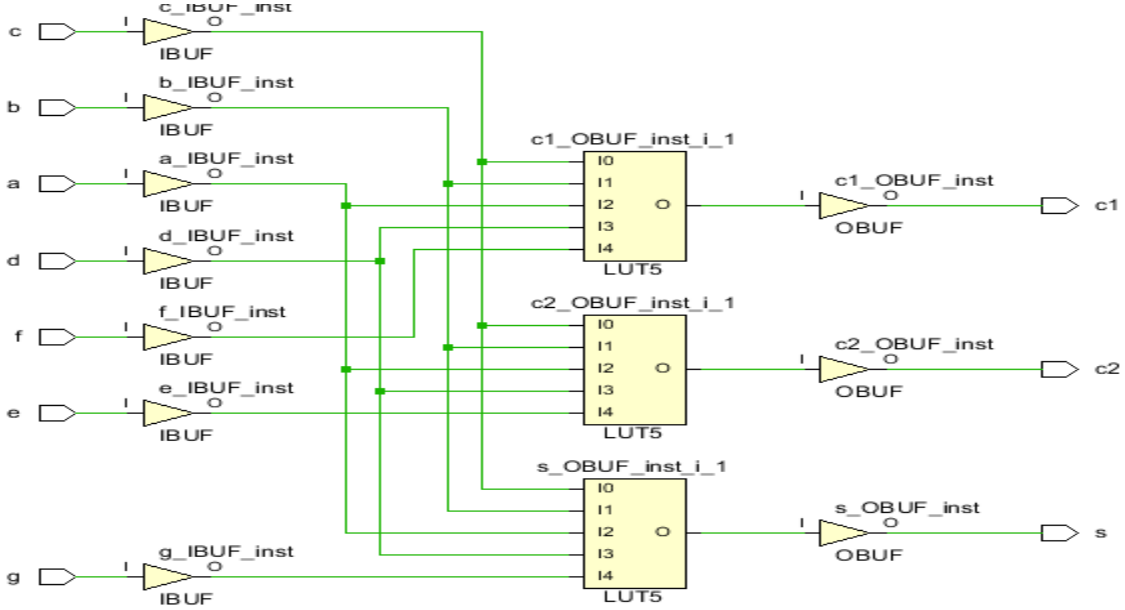

FIG5.SYNTHESIS DESIGN

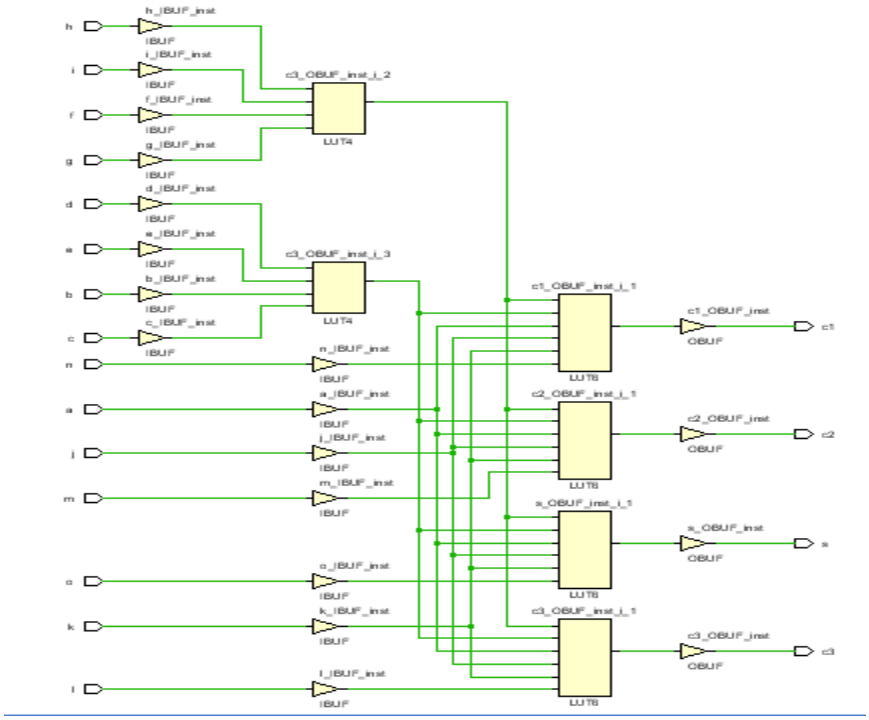

FIG6.SYNTHESIS DESIGNS

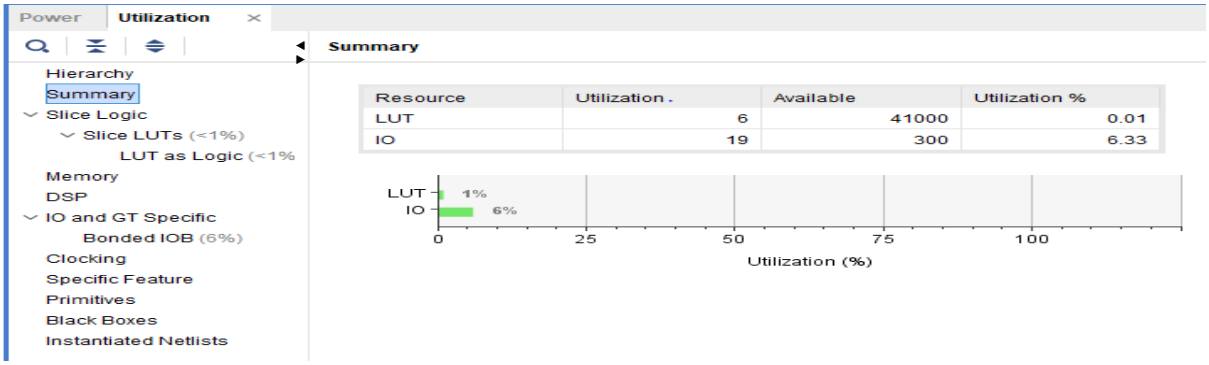

FIG7.UTILIZATION OF LUTS AND Ios.

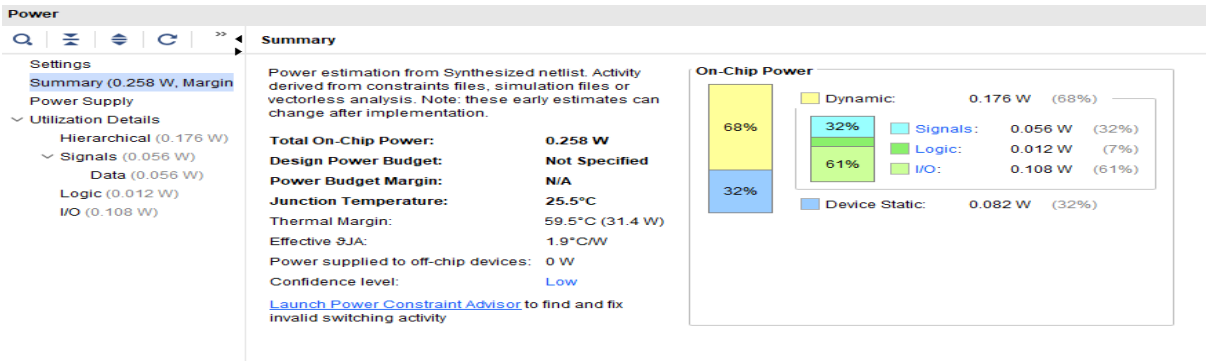

FIG9.POWER REPORT

CONCLUSION

This project is addressed the synthesis problem of two-phase bundled-data asynchronous pipeline controllers. To lighten the pipeline controllers, we developed a new logic synthesis concept called delay path sharing and reusing, by which we could significantly reduce the amount of the costly delay buffers. Precisely, the following conditions hold: 1) we proposed a technique of synthesizing a pipeline controller in a way to share delay buffers among the setup timing paths for minimally allocating them and 2) we devised an area-efficient delay circuit structure called DPU by extending the delay path sharing and proposed an in-depth synthesis flow of an asynchronous pipeline controller using DPUs.

REFERENCES

[1] P. A. Beerel, R. O. Ozdag, and M. Ferretti, A Designer's Guide to Asynchronous VLSI. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[2] N. C. Paver, "The design and implementation of an asynchronous microprocessor," Ph.D. dissertation, Dept. Comput. Sci., Univ. Manchester, Manchester, U.K., 1994.

[3] M. Ferretti and P. A. Beerel, "High performance asynchronous design using single-track full-buffer standard cells," IEEE J. Solid-State Circuits, vol. 41, no. 6, pp. 1444–1454, Jun. 2006.

[4] D. Hand et al., "Blade—A timing violation resilient asynchronous template," in Proc. 21st IEEE Int. Symp. Asynchronous Circuits Syst., May 2015, pp. 21–28.

[5] J. Simatic, A. Cherkaoui, F. Bertrand, R. P. Bastos, and L. Fesquet, "A practical framework for specification, verification, and design of selftimed pipelines," in Proc. 23rd IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC), May 2017, pp. 65–72.

[6] S. M. Nowick and M. Singh, "High-performance asynchronous pipelines: An overview," IEEE Des. Test. Comput., vol. 28, no. 5, pp. 8–22, Sep. 2011.

[7] I. Sutherland and S. Fairbanks, "GasP: A minimal FIFO control," in Proc. 7th Int. Symp. Asynchronous Circuits Syst. (ASYNC), 2001, pp. 46–53.

[8] M. Singh and S. M. Nowick, "The design of high-performance dynamic asynchronous pipelines: High-capacity style," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 11, pp. 1270–1283, Nov. 2007.

[9] K. M. Fant and S. A. Brandt, "NULL convention logic: A complete and consistent logic for asynchronous digital circuit synthesis," in Proc. Int. Conf. Appl. Specific Syst., Archit. Processors, Aug. 1996, pp. 261–273.

[10] A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," Proc. IEEE, vol. 94, no. 6, pp. 1089–1120, Jun. 2006.

[11] Z. Xia, S. Ishihara, M. Hariyama, and M. Kameyama, "Dual-rail/singlerail hybrid logic design for high-performance asynchronous circuit," in Proc. IEEE Int. Symp. Circuits Syst., May 2012, pp. 3017–3020.

[12] I. E. Sutherland, "Micropipelines," Commun. ACM, vol. 32, no. 6, pp. 720–738, Jun. 1989.

[13] M. Singh and S. M. Nowick, "MOUSETRAP: high-speed transition signaling asynchronous pipelines," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 6, pp. 684–698, Jun. 2007.

[14] K.-H. Ho and Y.-W. Chang, "A new asynchronous pipeline template for power and performance optimization," in Proc. 51st Annu. Design Autom. Conf. Design Autom. Conf. (DAC), 2014, pp. 1–6.

[15] N. Van Toan, D. M. Tung, and J.-G. Lee, "Energy-efficient and high performance 2-phase asynchronous micropipelines," in Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS), Aug. 2017, pp. 1188–1191.

[16] J. Myers et al., "A 12.4pJ/cycle sub-threshold, 16pJ/cycle near-threshold ARM Cortex-M0+ MCU with autonomous SRPG/DVFS and temperature tracking clocks," in Proc. Symp. VLSI Circuits, Jun. 2017, pp. C332–C333.

[17] A. Kapoor, N. Jayakumar, and S. P. Khatri, "A novel clock distribution and dynamic de-skewing methodology," in Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD), Nov. 2004, pp. 626–631.

[18] J. Heo and T. Kim, "Lightening asynchronous pipeline controller through resynthesis and optimization," in Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC), Jan. 2020, pp. 587–592.

[19] G. Gimenez, A. Cherkaoui, G. Cogniard, and L. Fesquet, "Static timing analysis of asynchronous bundled-data circuits," in Proc. 24th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC), May 2018, pp. 110–118.

[20] (2011). Silvaco 45 nm Open Cell Library. [Online]. Available: https://silvaco.co.kr/products/nangate/FreePDK45_Open_Cell_Library/

[21] (2017). IBM Ilog Cplex Optimizer 12.8.0. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

[22] G. Sagnol, "Picos, a python interface to conic optimization solvers," Zuse Inst. Berlin, Berlin, Germany, Tech. Rep. 12-48, 2012. [Online]. Available: http://picos.zib.de