FINE-GRAIN CLOCK GATING TECHNIQUE FOR A POWER SAVING 32-BIT PIPELINED MULTIPLIER IN ACTIVE MODE

Vo Minh Huan*, Pham Van Khoa
Ho Chi Minh University of Technology and Education

ARTICLE INFO

Received: 30/11/2023
Revised: 22/3/2024
Published: 22/3/2024

KEYWORDS

Coarse grain
Fine grain
Look ahead clock gating
Multiplier
Pipelined technique

ABSTRACT

Regularly, the clock gating enable signal is generated based on coarse grain clock gating where the clock enable signal is generated from system view. This study proposes the fine grain clock gating where the clock enable signal is generated from block view. The clock enable signal is self-generated based on look ahead technique which is applied to 32-bit pipelined multiplier. A pipelined multiplier breaks multiplication process into multiple stages, where each stage performs a small part of overall multiplication. The clock gating enable signal is self-generated from each pipeline stage which can be looked ahead to gate clocks to flipflops. The proposed 32-bit look ahead clock gating (LACG) pipelined multiplier using fine-grain technique shows the ability to efficiently save power consumption compared to the normal 32-bit pipelined multiplier using coarse grain technique. The study proves results on 32-bit pipelined adder and 32-bit pipelined multiplier in terms of power consumption, utilization area and functionality. The testbench is performed by five different testcases. The simulation results show the proposed multiplier saves power consumption up to 13.2% in case of random input test case compared to the normal multiplier. However, the proposed multiplier still has more utilization area overhead than the normal pipelined multiplier.

KỸ THUẬT CLOCK GATING CHỈNH TINH CHO MẠCH NHÂN ĐƯỜNG ỐNG 32 BIT TIẾT KIỆM CÔNG SUẤT TRONG CHẾ ĐỘ TÍCH CỰC

Võ Minh Huân*, Phạm Văn Khoa
Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh

THÔNG TIN BÀI BÁO

Ngày nhận bài: 30/11/2023
Ngày hoàn thiện: 22/3/2024
Ngày đăng: 22/3/2024

TỪ KHÓA

Chỉnh thô
Chỉnh tinh
Clock gating nhìn thấy trước
Bộ nhân
Kỹ thuật đường ống

TÓM TẮT

Thông thường, tín hiệu cho phép clock gating được tạo dựa trên clock gating chỉnh thô ở đó tín hiệu cho phép clock được tạo tại mức hệ thống. Nghiên cứu này đề xuất clock gating chỉnh tinh ở đó tín hiệu cho phép clock gating được tạo ra tại mức khối. Tín hiệu cho phép clock gating được tự tạo dựa trên kỹ thuật nhìn trước về khả năng clock gint được áp dụng cho mạch nhân đường ống 32 bit. Mạch nhân theo đường ống chia quá trình nhân thành nhiều giai đoạn, trong đó mỗi giai đoạn thực hiện một phần nhỏ của mạch nhân. Tín hiệu cho phép clock gating được tự tạo từ mỗi giai đoạn đường ống, tín hiệu này có thể được nhìn thấy trước để tắt clock đến flipflop. Mạch nhân đường ống nhìn thấy trước clock gating (LACG) 32 bit dùng kỹ thuật chỉnh tinh được đề xuất cho thấy khả năng tiết kiệm điện năng tiêu thụ một cách hiệu quả so với mạch nhân theo đường ống 32 bit thông thường dùng kỹ thuật chỉnh thô. Nghiên cứu chứng minh kết quả trên bộ cộng đường ống 32 bit và mạch nhân đường ống 32 bit về mức tiêu thụ điện năng, diện tích sử dụng và chức năng. Testbench được thực hiện bởi năm trường hợp kiểm tra khác nhau. Kết quả mô phỏng cho thấy mạch nhân được đề xuất tiết kiệm điện năng tiêu thụ lên tới 13,2% trong trường hợp thử nghiệm ngõ vào ngẫu nhiên so với mạch nhân thông thường. Tuy nhiên, mạch nhân được đề xuất vẫn có nhiều chi phí diện tích sử dụng hơn mạch nhân theo đường ống thông thường.

DOI: https://doi.org/10.34238/tnu-jst.9321

* Corresponding author. *Email: huanvm@hcmute.edu.vn*

1. Introduction

In a digital system, data is represented in binary form and processed using digital circuits. The ALU is a center component of these circuits that is responsible for performing basis arithmetic operations such as addition, subtraction, multiplication, and division as well as logical operations such as AND, OR, NOT, and XOR. The high-speed digital system consists of digital signal processing algorithms such as FFT and FIR which the multiplier is main arithmetic logic. In detail, 8.72% of all instructions are multiplier in a typical scientific program [1]. The multiplier is a fundamental arithmetic component that is responsible for performing arithmetic, and logic operations on binary data. The multiplier decides ALU performance. The role of multiplier in the ALU is to perform multiplication operations on binary data. Multiplication is required in many computer applications including graphics processing, scientific calculations, and cryptography. By implementing a multiplier in the ALU, the CPU can perform multiplication quickly and efficiently [2].

A normal multiplier also known as a single cycle multiplier, performs the entire multiplication operation in a single clock cycle [3]. It means that the input operands are multiplied in a single stage and results are immediately available at the output. While this approach is simple and straightforward, it can be relatively slow and resource-intensive for large operands. On the other hand, a pipelined multiplier breaks multiplication process into multiple stages, where each stage performs a small part of overall multiplication [4], [5]. The input operands are split into smaller pieces which are processed in parallel. The results of each stage are then added together to produce the final product. The advantage of a pipelined multiplier is that it can perform multiplication operations much faster than a normal multiplier. This is because the pipelined architecture allows for parallel processing of the operands, reducing the overall processing time required. Thus, a pipelined multiplier is often used in high-speed applications such as in digital signal processing, graphics processing, and cryptography to perform multiplication operations quickly and efficiently [4].

Pipeline technique is applied to different fields [5] – [8]. In general, a pipeline technique involves breaking down a complex task or process into a series of smaller, more manageable steps or stages, with data flowing from one stage to another by breaking down a complex task into smaller stages, each with a specific purpose. The pipeline technique allows for parallelization and efficient resource utilization. Different stages of the pipeline can be executed concurrently, reducing overall processing time and increasing throughput. By applying the pipeline technique to the multiplier design, the multiplication operation can be divided into several stages, allowing for parallelism and improved efficiency. By breaking down the multiplication process into these stages, each stage can be implemented as a separate pipeline stage with its own set of circuitries. This enables parallel execution of multiple multiplications and improves the overall throughput and performance of the multiplier. The pipeline stages can be overlapped so that while one multiplication is in progress, the subsequent multiplication can begin in the following stages, minimizing idle time and maximizing utilization [4], [9].

Clock gating, a widely employed method in numerous low-power circuits, serves the purpose of curtailing dynamic power dissipation [10], [11]. Its fundamental principle revolves around interrupting the clock signal when the circuit remains inactive. The crux of the clock gating technique lies in deactivating the clock signal to prevent unnecessary signal propagation through the circuit during periods of inactivity or standby mode. This measure is taken to avert inefficient power consumption stemming from needless signal switching when not required. The approach encompasses diverse clock gating methodologies like Synthesis Based Clock Gating, latch data-driven clock gating, and Auto-Gated Flip-Flops, each applied across a spectrum of applications including counters, digital clock circuits, linear feedback shift registers, and others [10] – [15].

Clock gating is a prevalent technique predominantly applied at the architectural design level, managing the activation and deactivation of the clock for the entire System-on-Chip (SoC). Its primary purpose is to conserve power by initiating sleep mode or waiting for external system requests [10], [11]. At the block level, clock gating governs the activation or deactivation of specific blocks within the system, allowing them to transition between idle and active states. However, when the clock signal is distributed across multiple sub-clock domains with differing frequencies or distinct operational principles within a block, the efficacy of implementing clock gating diminishes at the block level. This occurs because not all functional subblocks within a block enter idle or active states simultaneously. Therefore, the optimal power-saving potential of clock gating is realized only when the entire block transitions to sleep or idle mode. At certain instances, specific functional subblocks might remain active to maintain the system's necessary state, rendering traditional clock gating ineffective.

To refine power conservation by exerting control over clock gating at a finer granularity, each subblock should possess its clock control. This would enable individual subblocks to be independently clocked and self-selected based on control signals, allowing for precise adjustments to power consumption levels. The study applies this proposed concept to the pipelined system to build a lower power multiplier using fine-tuning clock gating technique each subblock. The study uses look ahead clock gating technique to calculates the clock self-enable signals of each FF before each cycle, based on the current cycle data of those FFs on which it depends. The content of the study is not simply using the usual clock gating technique. The author focuses on proposing a fine-grain clock gating method, used to optimize the power consumption on the clock network. Conventional clock gating technique only applies to a large system such as a SoC system. If the system is idle, the clock signal will be gated so as not to increase the switching of the clock network. The fine grain clock gating technique proposed in this paper focuses on reducing switching performance in small blocks (subblocks) rather than in a large system that is called a coarse grain clock gating technique. Given that, in a large active SoC block, there are bound to be a few subblocks that are not functional but still consume clock signals to switch, these blocks should be clocked off to reduce switching on clock network in active mode.

This study applies this fine grain clock gating technique based on look ahead clock gating technique to the 32-bit pipelined multiplier. It is possible to combine clock gating and multiplier circuits to create more power efficient designs. The next section presents the methods to propose fine grain clock gating technique and designs the look ahead clock gating technique to the 32-bit Pipelined multiplier. The simulation result is presented in Section 3. Finally, the study presents the conclusion.

2. Material and methods

2.1. Fine grained clock gating technique

Figure 1a illustrates the conventional clock gating technique. In this setup, a global clock signal (GCLK) undergoes multiplication with the enable signal EN via an AND gate, governing the functionality of logic subblocks 1, 2, and 3. When the EN signal equals 0, the AND gate output becomes logic 0, leading to the deactivation of the GCLK clock. Consequently, the network clock is deactivated, depriving the logic blocks of operational pulses. Conversely, when the EN signal equals 1, the logic blocks synchronize their operations with the GCLK signal, enabling the execution of switching functions within the logic circuit.

During normal operation in active mode, it's common for logic subblocks (subblocks 1, 2, and 3) not to be simultaneously active to fulfill their functions. For instance, if logic subblock 1 remains inactive while subblocks 2 and 3 continue to operate, serving the system's functions, it becomes advantageous to implement clock gating specifically at logic block 1 to conserve the

system's switching power, as demonstrated in Figure 1b. In this setup, a lookahead clock gating self-generator block generates three distinct enable signals (EN[0], EN[1], and EN[2]) based on the current operational status of the system. These signals, along with the GCLK signal, serve as inputs to AND gates (as depicted in Figure 1b). The resulting output from these AND gates controls the activation and deactivation of the GCLK pulse for the three-block logic circuit. By dynamically enabling or disabling logic subblocks 1, 2, and 3 in response to the system's working status, even while the circuit is operational, a substantial amount of wasted switching energy within the system can be effectively conserved.

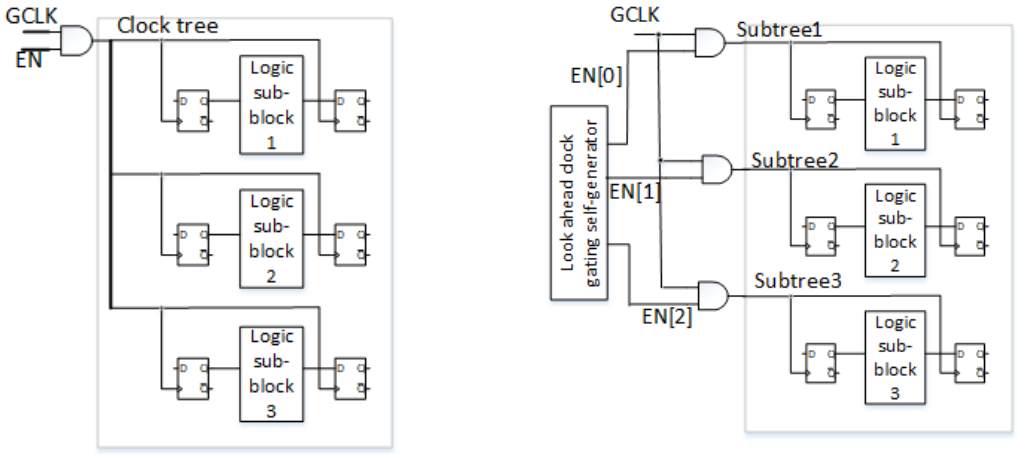

Figure 1. *(a) Block diagram of normal coarse grain clock gating technique b) block diagram of the proposed fine grain clock gating technique [16]*

When assessing the architectural area cost between two distinct clock gating methodologies, it's evident that the fine-grain clock gating approach involves supplementary hardware components. This includes the incorporation of a lookahead clock gating signal generator block along with a clock gating implementation block comprising AND gates. This highlights that the fine-grain clock gating module necessitates extra hardware resources to enable the segregation of the EN control signal and the subsequent implementation of the fine-grain clock gating architecture using AND gates.

Typically, clock gating involves utilizing the enable signal in conjunction with the clock signal through an AND gate, depicted in both Figures 1a and 1b. Figure 1a showcases the standard clock gating technique utilizing a single main clock (GCLK) to generate the clock tree signal steering the flip-flops. When employing this method (Figure 1a), a low EN signal results in bringing the clock tree signal to a non-switching low level. Consequently, the flip-flop data lines governed by the clock tree remain inactive. Consider the scenario in Figure 1a, where the clock tree signal operates normally, yet subblock1 logic transitions to an idle state. In this setup, to deactivate the clock tree, the system must wait for both subblock 2 and subblock 3 logics to become idle, causing unnecessary switching power wastage in subblock 1. Conversely, in Figure 1b, the global clock tree (GCLK) domain is partitioned into subtree domains (subtree1, subtree2, and subtree3) for their respective flip-flop data lines. Each subtree is associated with an enable signal fine-tuning the switching activation of its data lines. Consequently, while some subtree clock domains, like subtree2 and subtree3, may be active, allowing transitions, other subtree clock domains, e.g., subtree1, remain idle without switching. Therefore, subtree1 deactivates, preserving energy, while subtree2 and subtree3 continue normal operation, along with the main clock domain (GCLK) maintaining its regular function.

2.2. LACG technique

The comparison between input D and output Q is performed by an XOR gate. The output of XOR gate is fed to the AND gate to generate the required clock signal (CLK_G) for the flipflop. When both D and Q states are the same, the clock signal will remain low, and no power is consumed to switch. When their states are different, the clock signal will be the original clock signal and can make the necessary transitions to change the flip flop's state. The same technique can be used for falling edge flipflops using an XNOR or an OR gate as shown in Figure 2 (a).

The fine grain clock gating technique can gate clock signal and turn off the clock signal according to the Look Ahead Clock Gating method. The clock enable signal generator uses an XOR gate and an AND gate to self-generate the gating clock signal. The clock enable signal self-generator performs a preview of the signal changes at adjacent times to gating the signals into the FFD subblock in the circuit. The Q output of the FFD under consideration will be XORed with the D input (Q_(n-1)) of the preceding FFD. The output of the XOR will be AND with CLK. If there is no data change, CLK will be turned off and vice versa. Because of gating each FFD, the AND gate is used as the gating logic.

Figure 2(b) shows an 8-bit adder. A 32-bit adder using the pipeline method is designed from this 8-bit adder. Four 8-bit adders are combined to create a 32-bit adder. The 8-bit full adder has a design based on the carry ripple adder which is presented with input A [7:0] and B [7:0], carry input Cin and carry output Cout, Summation output S [0:7]. The summation of the most significant bits is only available after the carry signal has rippled through the adder from the least significant stage to the most significant stage. In Figure 2(b), an 8-bit LACG register shifts the input data including 8 LACG D-flipflops. The clock signal is applied flip flop D to control it as data is changed. The proposed idea is to build an adder using the clock signal that implements an adder with the look ahead clock gating method. The clock is used to control the data translation into the FFDs. If there is no change in the input signal, the clock will be turned off, the input signal and the carry bit will not be shifted. Conversely, when there is a change in input data, the FFD can gating the input data and allowing the carry bit to shift. The last carry bit (the 7th bit) of the previous Full Adder is shifted to the next Full Adder to generate a concatenated signal that works continuously.

2.3. The proposed multiplier

After designing the 8-bit Full Adder, we concatenate four 8-bit adders together to create a 32-bit adder as shown in Figure 3(a). The additional flip flops are designed to create a full 32-bit pipelined Adder. The flip flops have the function of buffering data, creating synchronization at the adder's output, and will shift as soon as the output response is detected. Thus, these flipflops separate from other adders so that they can function continuously without interruption. At the same time, these flip flops are also specially designed according to LACG technique gating the clock signals to reduce the power of this adder.

The next result will be reached after each clock pulse. The result of the summation of two numbers A [31:0] and B [31:0] is output S [31:0]. Therefore, the operation execution time will be significantly reduced compared to the adder circuit without a pipeline design. The results of the without pipelined multiplier operations must wait a time equal to the response time of the first result while the resources in the circuit are only maintained not used.

Figure 3 (b) shows block diagram of 32-bit LACG Pipelined Multiplier. The proposed multiplier consists of two inputs that B [31:0] inputs are multiplied by A [31:0] inputs. The logic result is synthesized from the AND gate set. The logic result at the AND gate is fed into the 32-bit pipelined adders. The resulting cascades of the previous adder are then fed into the next adder following the multiplication of the succeeding stages up to the end of the 31st bit. In addition, the first A0 bits and the S0 bits of the previous stages will be buffered to output in a synchronous

manner. Concurrently, the carry bit of the previous adder will be sent to the adder of the next stages to perform the addition. By doing so, we get a 32-bit multiplier at the end. Besides, the inside block is also designed according to fine grain clock gating method based on look ahead technique to reduce the switching power of the circuit.

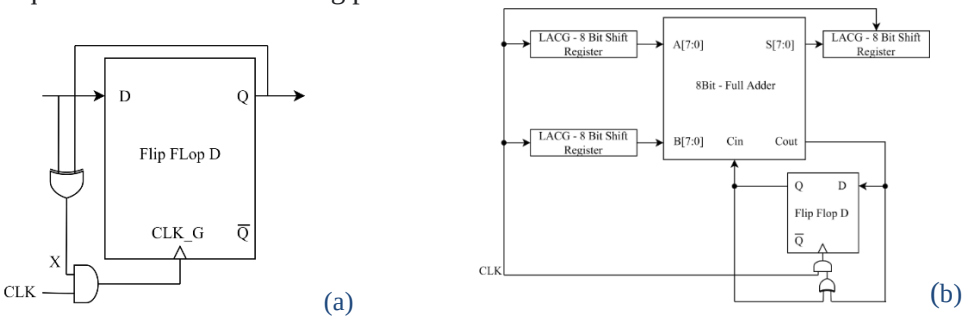

Figure 2. *(a) Look Ahead Clock Gating (LACG)– D Flip Flop, (b) 8-Bit LACG Full Adder*

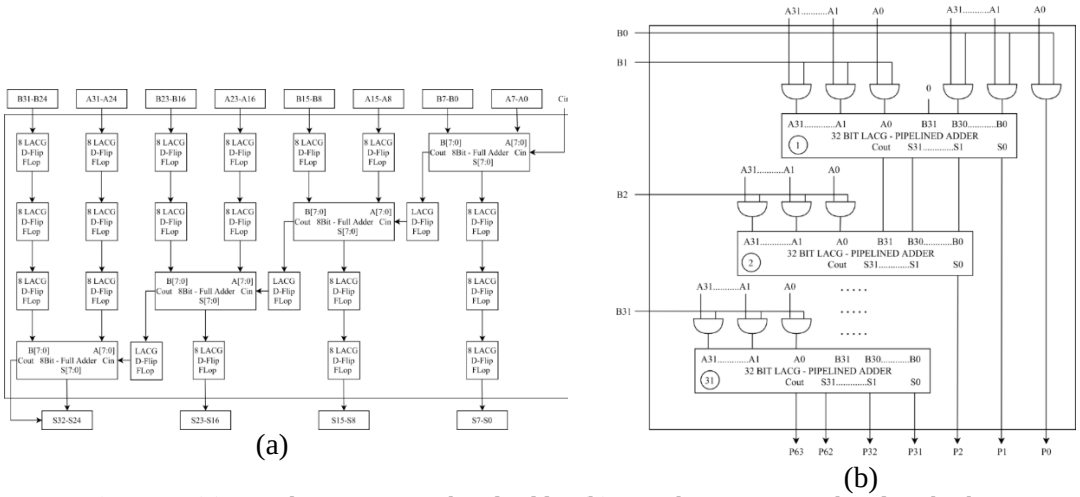

Figure 3. *(a) A 32-bit LACG Pipelined Adder (b) A 32-bit LACG Pipelined Multiplier with fine grain clock gating technique*

3. Simulation Results

3.1. Waveform simulation result

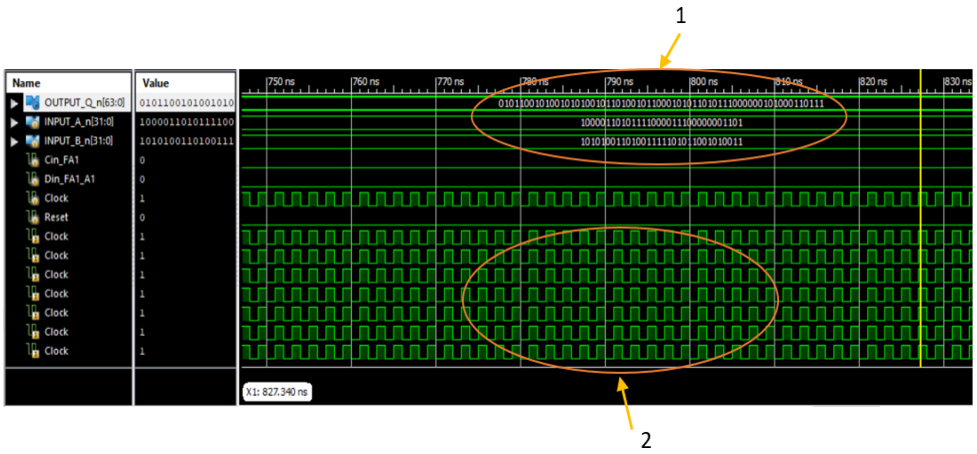

Figure 4. *Output waveform of 32-bit pipelined multiplier with coarse grain clock gating technique in case of randomly generated A, B inputs*

The simulation result is conducted in Xilinx ISE, Spartan 3E FPGA family with 500 MHz frequency. The top module of 32-bit LACG pipelined multiplier consists of the inputs input_A [31:0], input_B [31:0], clock, reset, Cin, and the 32-bit output Ouput_Q [31:0]. We use testbench to verify the functionality of multiplier designs. It is used to stimulate and observe the behavior of the design under various test cases and conditions. Testbenches are essential for validating the correctness and performance of digital designs. The testbench is used to properly evaluate the functionality of the 32-bit pipelined multiplier to perform basic calculations and to check the gating performance. We divide it into five testcases as follows: A, B will randomly be generated; A=1, B=1; A=1, B=0; A=0, B=1; and A=0, B=0.

Figure 4 shows a 32-bit pipeline multiplier designed from 32-bit pipeline adders using coarse grain clock gating technique. The result is simulated by 32-bit binary A, B inputs generated randomly. In Figure 4, the output (Output_Q) in region 1 performs the exact multiplication compared with the actual calculation. In the arrow region 2, we see that whether the input signal changes or does not change, the clock is still provided for the circuit to work.

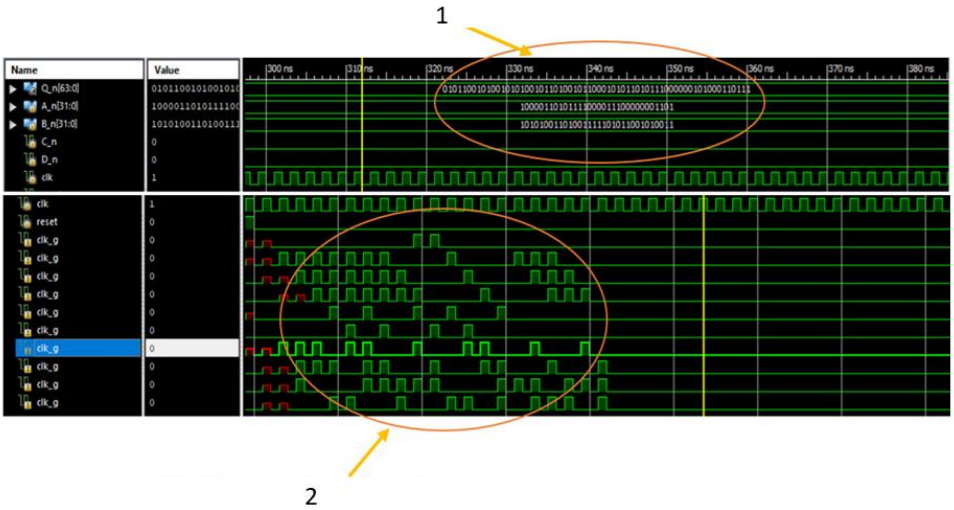

Figure 5. *The waveform of the 32-bit pipelined multiplier when applying fine grain LACG technique in case of randomly generated A, B inputs*

Figure 5 shows the output (Output_Q) result in the arrow region 1 is correctly compared with the actual calculation. On the arrow region 2 when the values change, the clock switching will be reduced to save power consumption in the circuit. Power consumption is measured on Xilinx software. Value Change Dump (VCD) simulation link file is created to calculate the power according to the provided input. We add the following 2 lines to the testbench simulation to create VCD file.

$dumpfile (“MUL_32BIT_G.vcd”);

$dumpvars (0,datn.uut);

The “dumpfile” command will dump the changes in the MUL_32BIT_G.vcd file. Changes are recorded in a file called a VCD file, which stands for value change dump. The VCD stores all information about value changes. The “$dumpvars” command is used to specify which variables should be dumped in the file mentioned by $dumpfile. When level is set to 0 and only module name is specified, it dumps all variables of that module and all variables in all lower-level modules initialized by a top module.

3.2. Performance measurement result

Table 1 shows the power comparison in different circuits consisting of 32-bit pipelined multiplier without Gating, 32-bit pipelined multiplier without Gating, 32-bit gated pipelined

adder, and 32-bit gated pipelined multiplier with look ahead clock gating technique. We use five testcases to verify the functionality as shown in Table 1. The results show the 32-bit gated pipelined multiplier with look ahead clock gating technique save power consumption up to 9.5%, 13.2%, 9.5%, 9.5% corresponding to random A, B input; A, B=1; A=1, B=0; A=0, 1B=1; and A=0, B=0, respectively.

Table 1. *Power comparison of 32-bit pipelined adders and 32-bit pipelined multipliers*

	A and B are random	**A, B = 1**	**A = 1, B = 0**	**A = 0, B = 1**	**A = 0, B = 0**
32-bit pipelined adder with coarse grain clock gating	0.121 W	0.110 W	0.110 W	0.110 W	0.110 W
32-bit pipelined adder with fine grain clock gating	0.101 W	0.093W	0.092 W	0.092 W	0.092 W
32-bit pipelined multiplier with coarse grain clock gating	0.254 W	0.220 W	0.210 W	0.210 W	0.210 W
32-bit pipelined multiplier with fine grain clock gating	0.230 W	0.191 W	0.190 W	0.190 W	0.190 W

Table 2. *Summary of resources used for the 32-bit pipelined adder*

	Pipelined 32-bit Adder with coarse grain clock gating			**Pipelined 32-bit adder with fine grain LACG**		
Resource	**Used**	**available**	**Utilization [%]**	**Used**	**available**	**Utilization [%]**
Number of slice Flip Flop	147	126800	1%	147	126800	1%
Number of slice LUTs	47	63400	1%	100	63400	1%
Number of bonded IOBs	99	210	47%	99	210	47%
BUFG	1	32	3%	0	32	0%
Occupied Slice	127	15850	1%	159	15850	1%

Tables 2 and 3 refer to the comparison of logic resources used when designing two circuits: pipelined multiplier with coarse grain clock gating and fine grain clock gating pipelined multiplier. The comparison includes Number of slice Flip Flop, Number of slice LUTs, Number of bonded IOBs, Number of bonded IOBs, BUFG, and Occupied Slice. Number of slices Flip Flop represents the number of Flip Flops that the two circuits use to design. Both circuits use the same number of Flip Flops. Number of slice LUTs act as combinational logics. Number of the clock gated multiplier LUTs is more than the number of slice LUTs of multiplier without gating because of the extra logics of AND, XOR.

Table 3. *Resource summary for the 32-bit pipelined multiplier*

	Pipelined 32-bit Multiplier with coarse grain clock gating			**Pipelined 32-bit multiplier with fine grain LACG**		
Resource	**Used**	**available**	**Utilization [%]**	**Used**	**available**	**Utilization [%]**
Number of slice Flip Flop	1957	126800	1%	1957	126800	1%
Number of slice LUTs	1205	63400	1%	1725	63400	2%
Number of bonded IOBs	99	210	47%	99	210	47%
BUFG	1	32	3%	0	32	0%
Occupied Slice	1133	15850	7%	2109	15850	13%

Number of bonded IOBs represents the number of inputs and outputs of a circuit. The multiplier circuit consists of the specified number of IO pins. The internal logic structure can increase and decrease but the number of pins IO will remain fixed.

BUFG also known as Global Buffers. The primary purpose of the BUFG is to distribute clock signals to various components and modules within an FPGA design. Clock signals are crucial for synchronous operation and timing synchronization in digital circuits. The BUFG ensures that clock signals are propagated with minimal skew and delay across the entire FPGA, allowing for reliable and synchronized operation of different logic elements. In 32-bit pipelined multiplier without LACG, number of BUFG are 1 because same clock is used to entire circuit and distribute to each flipflop. The number of BUFG is 0 in the 32-bit pipelined multiplier with LACG because the clock of the model is not directly shared for all Flip Flops but will be controlled by different clock signals for each Flip Flop to signal gating purpose.

Occupied Slice represents the space occupancy of the logic in the FPGA kit. Because the number of logics used to design the 32-bit gating multiplier is more than the logic amount of the non-gating 32-bit multiplier, the space occupancy is also higher as shown in both Tables 2 and 3.

4. Conclusion

The study proposes the 32-bit pipelined multiplier using look ahead clock gating with fine grain technique where the clock enable signal is self-generated from block view instead of system view. The look ahead technique is proposed based on changes of inputs. The study uses XOR and NAND gate to create internally the clock enable signal. The study verifies the proposed clock gating pipelined multiplier and normal multiplier with coarse grain clock gating in five different testcases. The results of the waveform are correct for all five test cases in both pipelined multiplier and clock gating pipelined multiplier. Resource area of the fine grain look ahead gating multiplier uses more than multiplier with coarse grain clock gating. Since the number of logic resources in the fine grain gated circuit is larger than that of the coarse grain-gated circuit, the occupied area on the clock fine grain gated circuit is larger. The power of the proposed circuit is reduced up to 13.2% compared to that of the normal multiplier circuit in case of random input testcase.

Acknowledgements

This work belongs to the project funded by Ho Chi Minh City University of Technology and Education, Vietnam.

REFERENCES

[1] G. Cometta and J. Cortadella, "Asynchronous multipliers with variable-delay counters," *ICECS 2001. 8th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.01EX483*), Malta, 2001, vol.2, pp. 701-705, doi: 10.1109/ICECS.2001.957572.

[2] N. Kandasamy, N. Telagam, and C. Devisupraja, “Design of a Low-Power ALU and Synchronous Counter Using Clock Gating Technique,” in *Advanced Computing and Intelligent Engineering*, Springer, Singapore, 2018, vol. 564, doi: 10.1007/978-981-10-6875-1_50.

[3]. R. Rathod, P. Ramesh, P. S. Zele, and K. Y. Annapurna, "Implementation of 32-Bit Complex Floating Point Multiplier Using Vedic Multiplier, Array Multiplier and Combined integer and floating point Multiplier (CIFM)," *2020 IEEE International Conference for Innovation in Technology (INOCON)*, Bangluru, India, 2020, pp. 1-5, doi: 10.1109/INOCON50539.2020.9298363.

[4] T. Mendez and S. G. Nayak, “Design of a Low-power Computational Unit using a Pipelined Vedic Multiplier,” *2023 International Conference for Advancement in Technology (ICONAT)*, 2023, pp. 1-6.

[5] S. Verma, A. A. Angelina, and V. S. K. Bhaaskaran, "Multiphase pipelining in domino logic ALU," *2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2)*, Chennai, India, 2017, pp. 309-315, doi: 10.1109/ICNETS2.2017.8067952.

[6] Z. Pei, L. Shang, S. Jung, and C. Pan, "Deep Pipeline Circuit for Low-Power Spintronic Devices”, *IEEE Transactions on Electron Devices*, vol. 68, no. 4, pp. 1962-1968, April 2021, doi: 10.1109/TED.2021.3059601.

[7] S. Yoshikawa, S. Sannomiya, M. Iwata, and H. Nishikawa, "Pipeline Stage Level Simulation Method for Self-Timed Data-Driven Processor on FPGA," *2020 8th International Electrical Engineering Congress (iEECON*), Chiang Mai, Thailand, 2020, pp. 1-5, doi: 10.1109/iEECON48109.2020.229515.

[8] Z. Xia, M. Hariyama, and M. Kameyama, "Asynchronous Domino Logic Pipeline Design Based on Constructed Critical Data Path," *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. 23, no. 4, pp. 619-630, April 2015.

[9] R. Pal, J. Ghosh, and A. Saha, "Novel Self-Pipelining Strategy for Efficient Multiplication," *2019 Devices for Integrated Circuit (DevIC)*, Kalyani, India, 2019, pp. 298-301, doi: 10.1109/DEVIC.2019.8783651.

[10] P. Sahu and S. K. Agrahari, "Comparative Analysis of Different Clock Gating Techniques," *5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, 2020, pp. 1-6.

[11] S. Chindhu and N. Shanmugasundaram, "Clock Gating Techniques: An Overview," *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*, Tiruchengode, India, 2018, pp. 217-221, doi: 10.1109/ICEDSS.2018.8544281.

[12] R. J. -H. Sung and D. G. Elliott, "Clock-Logic Domino Circuits for High-Speed and Energy-Efficient Microprocessor Pipelines," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 5, pp. 460-464, May 2007, doi: 10.1109/TCSII.2007.892212.

[13] S. Wimer and A. Albahari, "A Look-Ahead Clock Gating Based on Auto-Gated Flip-Flops," *IEEE Transactions on circuits and systems-i: Regular papers*, vol. 61, no. 5, pp. 1465-1472, May 2014.

[14] M. B. Junghare and A. S. Shinde, "A Clock Gating Technique Using Auto Gated Flip Flop for Look Ahead Clock Gating," *International Journal of Science and Research (IJSR)*, vol. 4, no. 7, pp. 1465-1472, July 2015.

[15] R. Manjith and C. Muthukumari, "Dynamic Power Reduction in Sequential Circuits Using Look Ahead Clock Gating Technique," *World Academy of Science, Engineering and Technology International Journal of Electronics and Communication Engineering*, vol. 9, no. 2, pp. 252-258, 2015.

[16] M. H. Vo, "Fine-tuned Clock gating technique reducing application dynamic power consumption in ping pong game," (in Vietnamese), *LQDTU Journal of Science & Technique - Section on Information and Communication Technology*, no. 11, pp. 64 – 70, April 2018.