ISSN 2959-6157

Four-Bit Parallel Multiplier based on Digital Circuit Design

Yinsong Yan

Southwest Jiaotong University, Chengdu, China
Corresponding author: el22y3y@leeds.ac.uk

Abstract:

This paper presents an investigation into a four-bit parallel multiplier based on an enhanced digital circuit design. The advent of computers and digital electronic systems has led to a significant increase in the importance of multiplication operations in a wide range of applications.In this paper, the circuit is designed using the Logisim, and the designed circuit is converted into a model and input into Modelsim for simulation. Finally, the feasibility of the circuit is verified through the Modelsim model. The objective of the circuit is to implement the multiplication operation of two four-bit inputs. The model designed in Logisim can facilitate the process of code writing in Modelsim simulations. Furthermore, once the simulation is complete, the simulation results can be compared with the circuit diagram to ascertain their alignment with reality. The entire model is based on the concept of a full adder, and the delay is reduced by disconnecting the serial feed chain, which results in a 25% reduction in delay compared to a full adder with two columns. This approach effectively achieves the desired reduction in delay.

Keywords: full adder, digital circuit, multiplier

1.Introduction

In light of the accelerated advancement of computers and digital electronic systems, the significance of multiplication as a pivotal component of the four fundamental operations cannot be overstated. A four-bit parallel multiplier is a digital circuit that is commonly employed in combinational logic circuits. Its principal function is to facilitate the multiplication of two four-bit binary numbers. In comparison to adders and subtractors, multipliers are capable of performing more intricate operations and are pervasively utilized in computer processors, image processing and other domains [1]. The two input binary numbers are multiplied bit by bit and summed up by adding and shifting, resulting in an eight-bit binary number as the product. Furthermore, while pursuing efficient computing speeds, it is also important to focus on circuit area and power consumption in order to meet the demand for low power consumption in embedded systems and mobile devices [2]. It has been demonstrated that the simulation and construction of circuit diagrams of digital circuits can be an effective method for both learning and testing the knowledge acquired about digital circuits [3]. To gain a deeper understanding of the multiplier, a circuit diagram of the four-bit binary multiplier was created using Logisim. Once completed, the diagram was subjected to logical verification, whereby the inputs were altered and the outputs were converted to decimal numbers. This process was undertaken to ascertain the viability of the multiplier and to facilitate the adaptation of the schematic diagram of its package, which would subsequently inform the simulation phase. Once the circuit diagram of the device has been completed, the device is simulated in Modelsim. The operation logic of the multiplier is then compiled through the Verilog language. Following this, the logic of the multiplier is simulated and verified by writing the test code, and the output of the multiplier can be detected through the function image simulated by the test code.

The initial section of this document delineates the introductory aspects, while the subsequent section elucidates the gate circuit design and an analysis of the circuit's structural composition. The subsequent section employs a series of simulation experiments conducted through Modelsim to procure the concluding experimental data and conclusions. The final section presents a comprehensive summary of the entire paper.

2.Gate Circuit Design

2.1 Design ideas

In order to design an operation that can realise the multiplication of two four-bit binary numbers, the two binary numbers are set as the multiplicand $A_3A_2A_1A_0$ and multiplier $B_3B_2B_1B_0$, respectively. The two binary numbers, the multiplied number and the multiplier, are represented by high and low levels, respectively. The result of the multiplication operation is then directly output as an eight-bit binary number. The technical methodology is as Figure 1:

				A3B0	A2B0	A1B0	A0B0
			A3B1	A2B1	A1B1	A0B1	
		A3B2	A2B2	A1B2	A0B2		
+	A3B3	A2B3	A1B3	A0B3			
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Fig.1 Theoretical model of parallel multiplier

Two four-bit binary numbers are divided into separate one-bit binary numbers for the purpose of performing arithmetic operations by means of a divider. The addition of each column is performed using a full adder, with the use of a one-bit full adder. This is achieved by obtaining an array of several gates to ensure that the progressive output of bit i of the jth step is the progressive input of bit i+1 of the next step. Additionally, a row of serially progressive adders is incorporated at the end. Each one-bit binary number of the eight individual outputs is integrated into an eight-bit binary number by means of a line splitter (Figure 2).

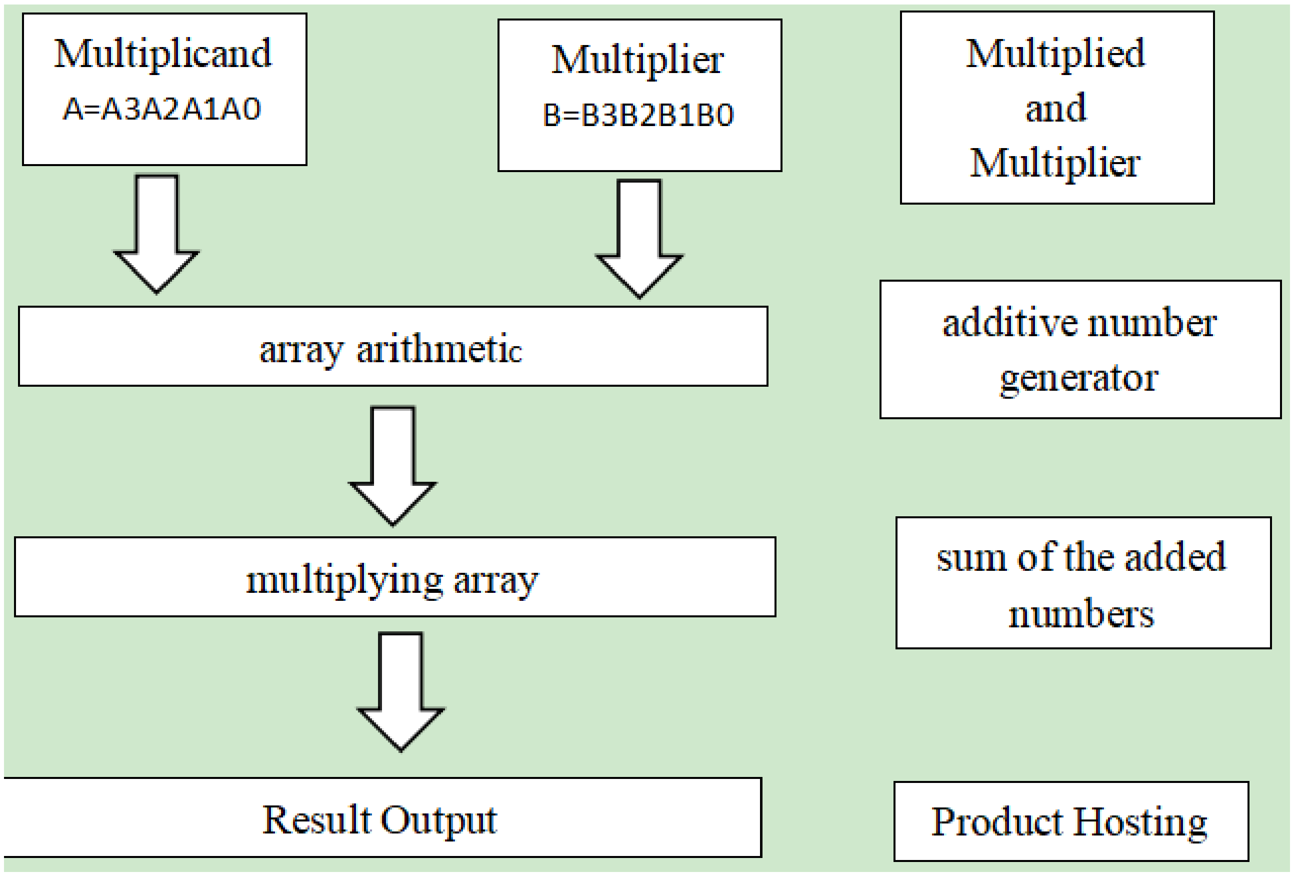

Fig. 2 Hardware structure of the parallel multiplier

2.2 Gate structure of a parallel multiplier

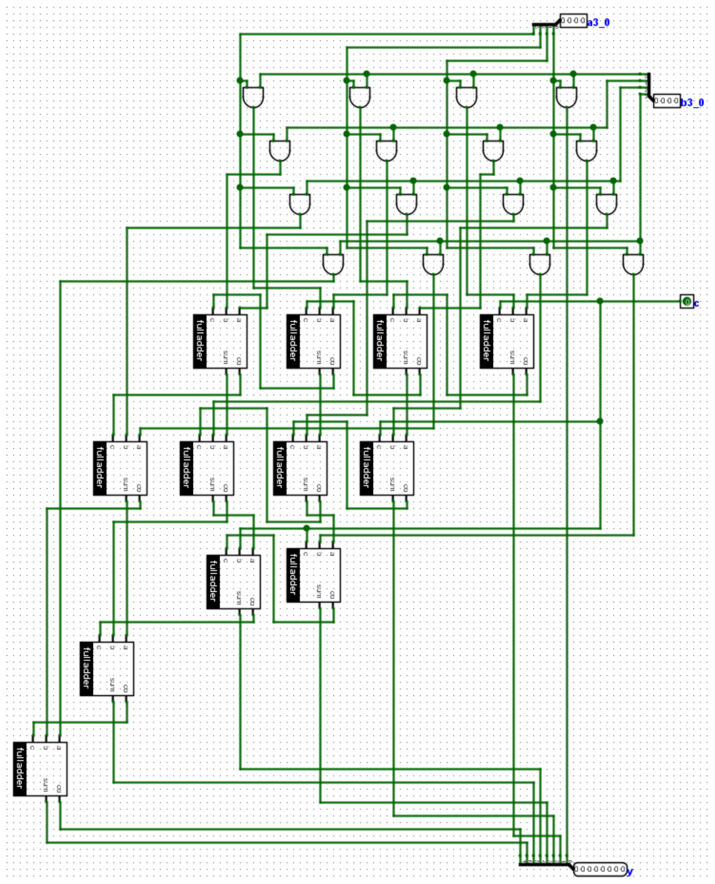

Fig. 3 Gate Circuit Diagram

This paper presents a circuit diagram created using the Logisim software [4]. The multiplier and the two four-bit binary numbers to be multiplied are divided into eight separate one-bit binary numbers by two identical four-bit shunts. Each one-bit binary number is then integrated by an AND gate, and the intermediate result is obtained by adding the intermediate results step by step using all levels of the full adder module and storing the intermediate result in the sum. The rounding signals are present in the co. The original two four-bit binary numbers and one eight-bit binary number are decomposed into separate one-bit binary numbers through the use of three shunts: two four-bit and one eight-bit [5]. The resulting one-bit binary number is then substituted into the circuit for the operation. Furthermore, Figure 3 illustrates the full adder, which will be discussed in greater detail in the subsequent section.The function of component c in the circuit is to detect whether the circuit is functioning correctly. When the circuit is in test mode, the potential of component c should be set to a high level to ascertain whether the circuit is able to establish a typical pathway. In normal operation, the potential of the circuit should be set to a low level, thereby ensuring that all the adders are able to perform the multiplication of two four-digit binary multipliers. The following section delineates the specific functions of each component.

2.2.1 Input side

The two inputs, designated as A and B, should be set to the west, with the number of input bits set to four. The names of A and B should be set to the east. The two fan-in and fan-out values should be set to four. The four-bit separator should be set to the west and south. The appearance should be set to the right-handed direction. The two four-bit binary numbers should be decomposed by the two four-bit separators to yield eight binary numbers, which will be used in subsequent operations.

2.2.2 Array arithmetic

Subsequently, the array computation is performed. The array computation permits the generation and propagation of multiple levels of rounding, whereby each gate is employed to calculate the rounding. In the event that both inputs are 1, a rounding signal is generated and transmitted to the subsequent level of the adder. Furthermore, the logic gates operate in conjunction with the adder, generating and propagating progressions in this manner to guarantee the accurate transmission of the progression signal to the subsequent stage.

2.2.3 .Multiplying array

The multiplication array is realized through the use of multiple full adders. The circuit diagram of the full adder is provided below for reference.

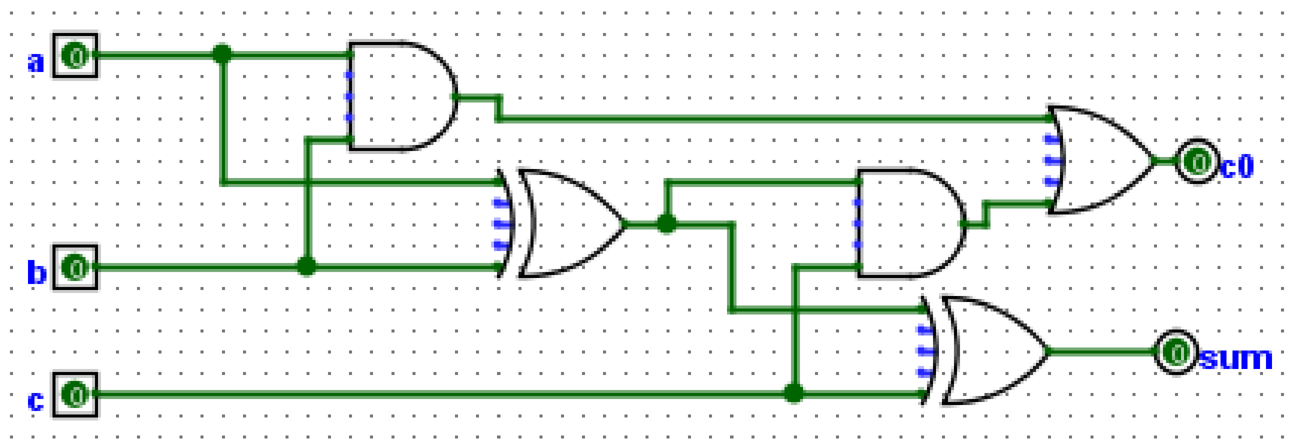

Fig. 4 Full Adder Circuit Diagram

The adder is composed of one or more gates, two different gates, and two or more gates [6]. The progression c0 receives the initial portion of the A and B sequence through the primary gate, the subsequent segment through the secondary gate, and the final result of the operation of the distinct-or gates a and b. Subsequently, the output of the progression is conveyed through an or gate. The sum is obtained by first using the first ISO gate to obtain an ISO of A and B, and then using the second ISO gate to obtain a sum by performing an ISO operation on the result of the first step with C. Once the full adder circuit diagram has been drawn, the circuit of the full adder is encapsulated; that is to say, the corner labels are labelled (see Fig.5) and the output and input ports are adjusted according to the actual use of the circuit. Once the design of the full adder has been completed, the multiplication array can be assembled. This allows multiplication to be converted to multi-stage addition by summing the results of the array operations and passing them through the stages. In this stage of the multiplication array, the partial products of all the array operations are added together to eventually form eight one-bit binary numbers, which in turn give each bit of the final result. The c0 to c15 elements of the multiplicative array represent the product between the bits a and b, which is also the partial product obtained from the array computation as previously described. The specific concrete mode of operation of partial product is as follows:c0 =a1&b0,c1=a2&b0,c3=a3&b0,while for the same reason-c15=a3&b3 [7]. Given that both A and B are four-bit binary numbers, the total number of bits in the partial products is sixteen. These partial products are subsequently added step by step by the adder to form the final eight-bit output.

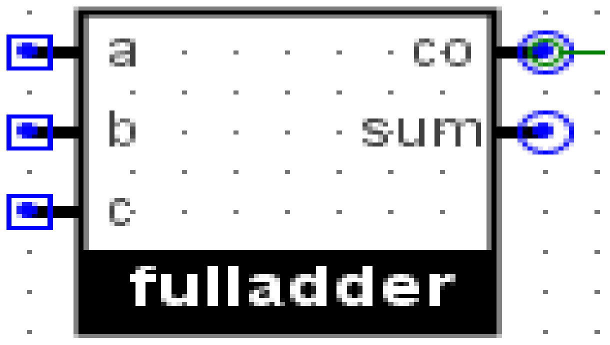

Fig. 5 Full adder package

2.2.4 Output of results

The output orientation should be set to west, the number of output bits to eight, and the label ‘y’ should be used to indicate the output of this four-bit parallel multiplier. Before the eight-bit output, a separator should be connected to set the orientation to north, fan in and fan out to eight, and appearance to the left-hand direction. The eight one-bit binary numbers should then be combined by this eight-bit separator into one eight-bit binary number.

3.Simulation

3.1 Simulation Methods

The code for the four-bit parallel multiplier has been written in Modelsim in Verilog language and compiled in hardware language based on the circuit diagram drawn in Logisim [8]. Firstly, a project must be created and the module name set to ‘multiplier’ (as illustrated in Figure 6). Two four-bit binary numbers, a and b, must be introduced as the multiplier and multiplied numbers, respectively, as well as the input signals of the module. The eight-bit binary number, y, must be introduced as the output of the four-bit binary number. The rest must be introduced as the reset signal input, and the s and co must be introduced as the twelve-bit wide intermediate lines for storing the data inside the four-bit parallel multiplier. The s and co are introduced as twelve-bit wide intermediate lines within the four-bit parallel multiplier for the storage of partial sums and rounding, while the c is introduced as a sixteen-bit wide intermediate line situated in the centre of the four-bit parallel multiplier for the storage of the result of the multiplication operation of each bit.

```
module multiplier(reset,a,b,y);
  output [7:0] y;
  input [3:0] a,b;
  input reset;
  wire [11:0] s,co;
  wire [15:0] c;
```

Fig.6 Simulation parameter introduction

Following the establishment of the parameters in accordance with the circuit diagram generated by Logisim, the 16- and 12-bit gates and full adders are compiled in a step-by-step manner, utilizing the assign statement for each multiplication result and the full adder statement for the pairwise sum and rounding co calculations.

3.2 Testing

Following the completion of the code for the four-bit parallel multiplier, test code is additionally written to test the functionality of the code [9]. This is named a file as tb_multiplier, in which the Verilog language is utilised to write two for loops about i and j (as shown in Fig. 7). Additionally, the operations of the intermediate processes of the four-bit parallel multiplier are compiled.

```
for(i=0;i<16;i=i+1)begin
  for(j=0;j<16;j=j+1) begin
     #200 a=i;b=j;
  end
end
```

Fig.7 The for loop for i and j

3.3 Simulation results

Once the code and test code have been compiled, the simulation file can be used to simulate the multiplier [10]. The Modelsim simulation function can be employed to obtain the simulation results of a four-bit parallel multiplier (see Fig 8).

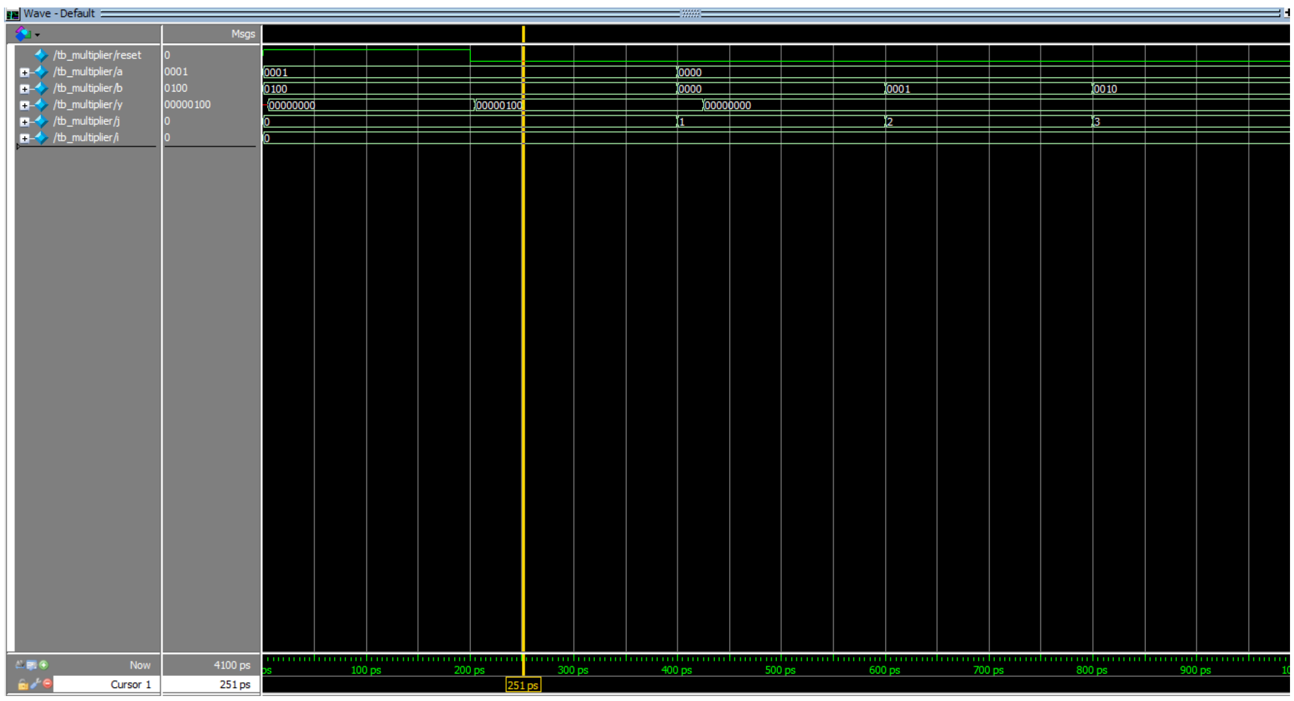

Fig.8 Simulation results

As illustrated in Figure 8, the input value of a is quasi-converted from a four-bit binary number to a decimal number of 1, the input value of b is converted from a four-bit binary number to a decimal number of 4, and the two numbers are presumed to result in 4. The output of y is then converted to a decimal number of 4, which can be obtained from the four-bit parallel multiplier that has the normal multiplication operation capability. Furthermore, the four-bit parallel multiplier mapping diagram can be obtained through the use of Modelsim's dataflow function.

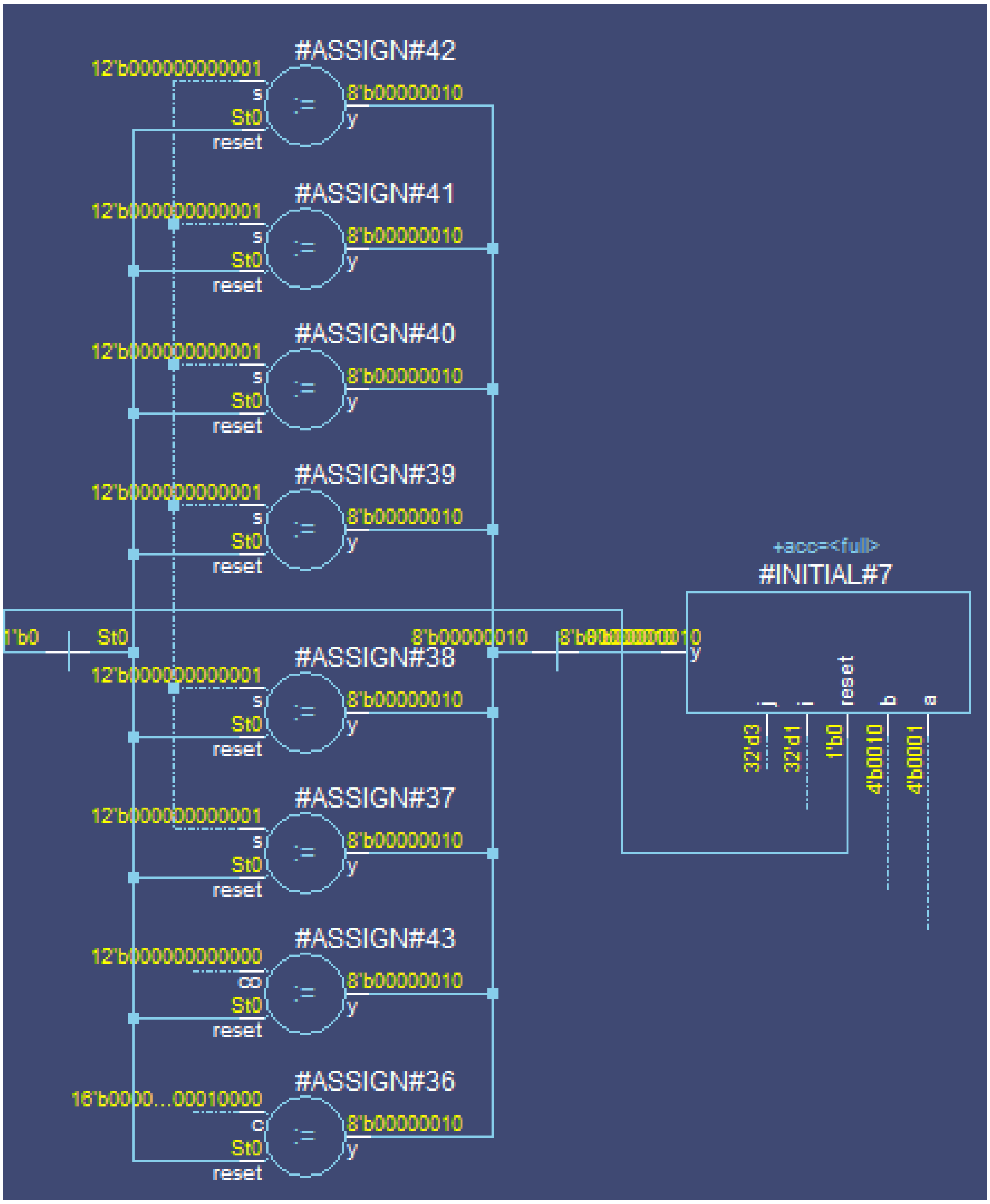

Fig.9 The dataflow

4.Conclusion

This paper presents the design and simulation of a four-bit parallel multiplier. The multiplication operation is performed by shift summing, which also serves to reduce the delay between the various stages of the gate circuit, the power consumption and the area occupied by the circuit. The circuit diagram was designed in Logisim, and the code was verified through the simulation feature of Modelsim. In future research, two avenues for improvement should be considered. The first is to further reduce the delay and power consumption by using the over-advancing adder on delay, power consumption and area. The second is to enhance the computing speed of the model to improve its prospects for application.

References

[1]Immareddy, S., & Sundaramoorthy, A. A survey paper on design and implementation of multipliers for digital system applications. Artificial Intelligence Review,2022,55(6), 4575–4603.

[2]Chen, J., & Tian, G. Analysis of low power consumption in digital integrated circuit design. Electronic Components and Information Technology,2023, 7(06), 144-147.

[3] Rokicki, T. G. Representing and modeling digital circuits (Doctoral
dissertation, Stanford University). ProQuest Dissertations & Theses. (Publication No. 9422115) ,1994.

[4] Burch, C. Logisim: A graphical system for logic circuit design and simulation. Journal on Educational Resources in Computing (JERIC), 2002,2(1), 5-16.

[5] Zhang, C., & Wu, J. Algorithmic study of FPGA-based multiplier. Electronic Fabrication,2015,(18), 13-14.

[6] Bhattacharyya, P., Kundu, B., Ghosh, S., Kumar, V., & Dandapat, A. Performance analysis of a low-power high-speed hybrid 1-bit full adder circuit. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2015,23(10), 2001-2008.

[7] Ashour, M. A., & Saleh, H. I. An FPGA implementation guide for some different types of serial–parallel multiplier structures. Microelectronics Journal, 2000,31(3), 161-168.

[8] Aguirre Morales, J. D., Marc, F., Bensoussan, A., & Durier, A. Simulation and modelling of long term reliability of digital circuits implemented in FPGA. Microelectronics Reliability, 2018,88–90, 1130-1134.

[9] Das, S. R., Li, J. F., Nayak, A. R., Assaf, M. H., Petriu, E. M., & Biswas, S. N. Circuit architecture test verification based on hardware software co-design with ModelSim. IETE Journal of Research, 2013,59(2), 132–140.

[10] Instructional Simulation System for Selected Topics of Logic Design using ModelSim and EWB. Al-Mustansiriyah Journal of Science, 2014, 25(1)