ECE385 Experiment #5

Eric Meyers, Ryan Helsdingen

Section ABG; TAs: Ben Delay, Shuo Liu

February 24th, 2016

emeyer7, helsdin2

I. Introduction

The purpose of this lab was to design and construct a 2s compliment 8-bit multiplier that uses a shift-and-add algorithm. The user will input their desired multiplicand and multiplier into switches and these will be stored in two shift registers (A and B). The multiplier is built upon a control unit with a state machine, so once the "run" button is pressed, the machine will cycle through multiple states and output the value in the combined 16-bit value "AB".

II. 8-bit Multiplication Example

Table I shows each step of the multiplication process given inputs Multiplier B=7 and Multiplicand S=-59. The result is -413 in decimal and 1111 1110 0110 0011 in binary. As you can see, bolded at the bottom of Table I is the expected result loaded into A and B. Shift functions occur eight times, right-shifting the bit. Add functions occur before each shift and only if the previous M bit is equal to 1. During an add function, S gets added to A. The only exception is before the eighth shift which would be a subtract function rather than an add function. S would get subtracted from A. This exception, as you can see, is not needed for this example.

TABLE I: Reworked 8-bit Multiplication Problem.

Function	X	A	B	M
Clr_A_Ld_B	0	0000 0000	0000 0111	1
ADD (1)	1	1100 0101	0000 0111	1
SHIFT (1)	1	1110 0010	1000 0011	1
ADD (2)	1	1010 0111	1000 0011	1
SHIFT (2)	1	1101 0011	1100 0001	1
ADD (3)	1	1001 1000	1100 0001	1
SHIFT (3)	1	1100 1100	0110 0000	0
SHIFT (4)	1	1110 0110	0011 0000	0
SHIFT (5)	1	1111 0011	0001 1000	0
SHIFT (6)	1	1111 1001	1000 1100	0
SHIFT (7)	1	1111 1100	1100 0110	0
SHIFT (8)	1	**1111 1110**	**0110 0011**	1

III. Description of Circuit

The purpose of the 8-bit multiplier is to take two 8-bit user-defined words, multiply them together and place the resulting product in the two combined registers.

Initially both storage registers will begin with 00000000 loaded into them. On the Altera Board, the operator will control the last eight general purpose switches to indicate what the word they wish to load is, then press the "Clear_A_Load_B" push button to the right. This will then take the switch values and load them into Register B. Therefore after this operation is performed, A will still contain 00000000, and B will contain XXXXXXXX (X being the value the user defines).

Then once the user indicates the multiplier (Register B), they will then indicate the multiplicand by changing the order of the eight switches. Once this is done, then the user will press the "Run" switch and the control unit will cycle through all states within the FSM and come to a halt. The product of the two values will be displayed in register AB(concatenated A + B) and the X bit will be displayed on the green LED next to the HEX values.

The FSM is a part of the control unit and will be explained below and the "X-bit" will also be described in detail below.

IV. Purpose of Modules

Please refer to Figure 5 to understand the how each module interacts with one another. The multiplier is

broken down into four primary modules as listed below:

- Shift Register
- Full Adder/Subtractor
- Control Unit
- X Register

The first of these modules is the shift register and it's purpose is to store the contents of each 8-bit word the user specifies. The concatenation of the two registers together (AB) contains the 16-bit product of the multiplication. The shift register shifts right when enabled by the control unit, pushing the least significant bit out through its "shift_out" output. The data also is parallel loaded in through its input in the module and has parallel outputs used to display on LEDs on the Altera Board.

The next module is the Full Adder/Subtractor which essentially does exactly what it states: both adds and subtracts (depending on the control unit). The inputs are 16 data bits (8 bits for A and 8 bits for B) and a "subtract" bit which indicates whether or not subtraction must be performed.

Because subtraction is adding the 2s complement of a number, the team decided to use a trick learned back in Lab 3. To get the 2s complement, one must flip all of the bits and add one. To conditionally flip all of the bits, the team used an XOR gate with inputs of B and the "subtract" bit. This is because if the "subtract" bit is high, then the B value must be flipped, and if it is low then it stays the same. Then this logic value must be fed into the nine inputs of the full adders along with the "subtract" bit on the first full adder. Please reference Figure 8 to get a better understanding of this.

This setup will take the 2s complement of the number and add it to the A value if the "subtract" bit is high, but if the "subtract" bit is low then it will simply regularly add the two A and B values.

The next module is the Control Unit which dictates how the Finite State Machine operates and how the entire circuit functions. This unit takes in 3 inputs (Run, Reset, and Clear_A_Load_B) that are designated by the operator. These inputs then determine the outputs of the control unit which are Load, Shift, Add, and Shift. These four outputs are determined by the Finite State Machine described in "Section V: State Diagram".

Essentially, the outputs are telling each module if they are in use or not. So for example, if the multiplier needs to perform an "Add" or "Subtract" operation, it will output a high value on the respective output and this will be then fed into the proper module which will then be activated.

Last but not least is the "X Register" Module which contains the status of the "X" bit. The "X" bit is essentially the most significant bit in the A register. This register tells whether the result of the multiplication is positive or negative and is mainly used to view the output of the multiplication on the Altera Board by feeding it into a green LED to see its status.

V. State Diagram

RYAN SECTION

VI. Schematic/Block Diagram

Please refer to "Section X : Figures" of this document to view the Schematic/Block Diagrams. Figure 6 displays the entire block diagram of the multiplier, with block labels and interconnections. The multiplier, as stated before is broken down into several modules. Those modules are as follows with the figure references aside them:

- Shift Register - Figure 7
- Full Adder/Subtractor - Figure 8
- Control Unit - Figure 9 (Top Half Only)
- Control Unit - Figure 10 (Bottom Half Only)
- X Register - Figure 11

VII. Pre-Lab Simulation Waveforms

Please refer to "Section X : Figures" of this document to view the Pre-Lab Simulation Waveforms. There was a total of four simulations performed. All options were explored on this multiplier and the following inputs were used:

- +7 * -59 - (Figure 1)
- -7 * +59 - (Figure 2)
- +7 * +59 - (Figure 3)
- -7 * -59 - (Figure 4)

VIII. Post Lab

1) Refer to the Design Resources and Statistics in IQT.30-32 and complete the following design statistics table.

These numbers are shown below in Table **??** .

Come up with a few ideas on how you might optimize your design to decrease the total gate count and/or to increase maximum frequency by changing your code for the design.

Answer:

TABLE II: Reworked 8-bit Multiplication Problem.

LUT	95
DSP	0
Memory (BRAM)	0
Flip-Flop	0
Frequency	239.52 MHz
Static Power	98.52 mW
Dynamic Power	0 mW
Total Power	147.8 mW

IX. Conclusion

In this lab, the team was able to successfully program and test a working multiplier on the DE2 FPGA board.

X. Figures

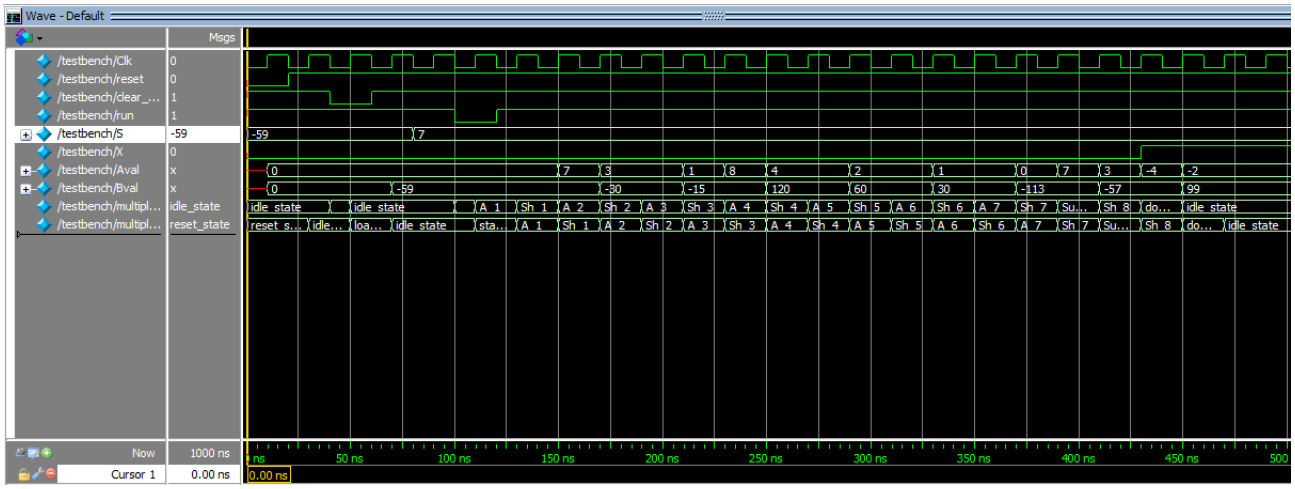

Fig. 1: ModelSim Simulation Output (+7 * -59)

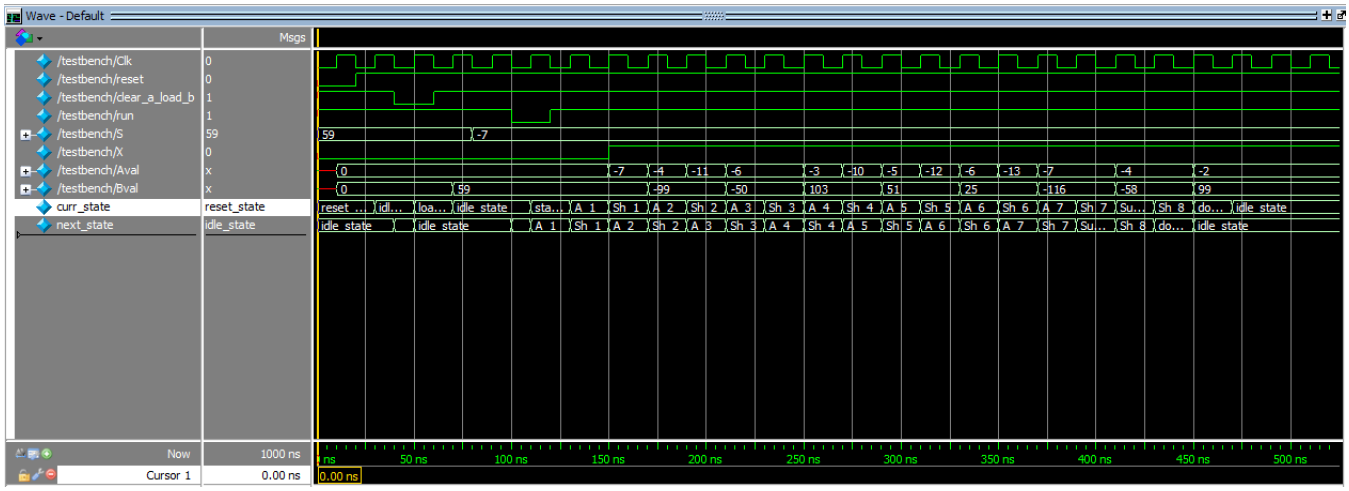

Fig. 2: ModelSim Simulation Output (-7 * +59)

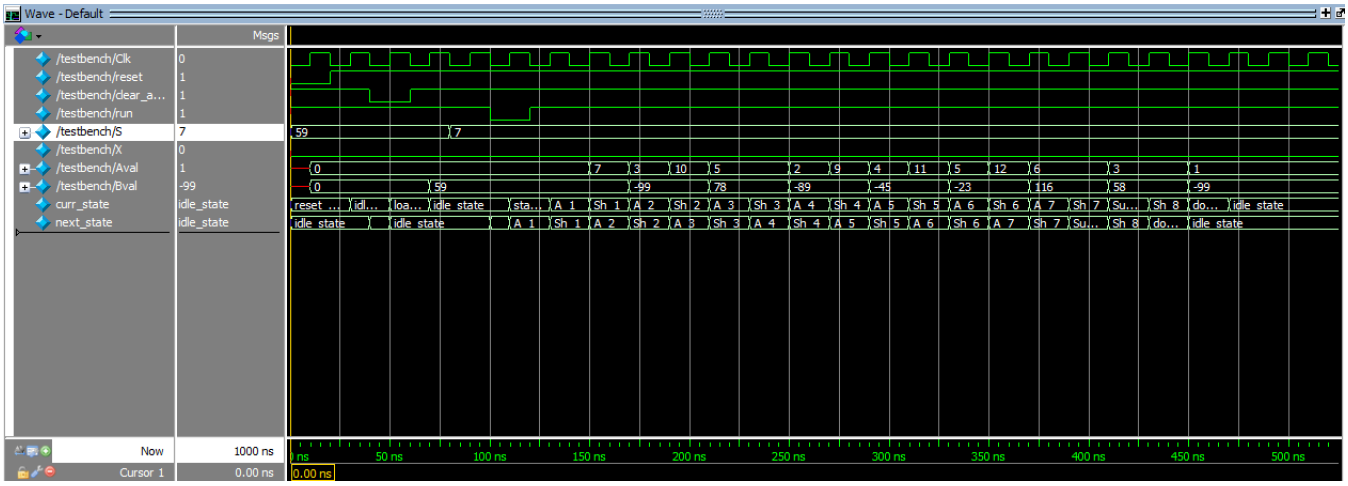

Fig. 3: ModelSim Simulation Output (+7 * +59)

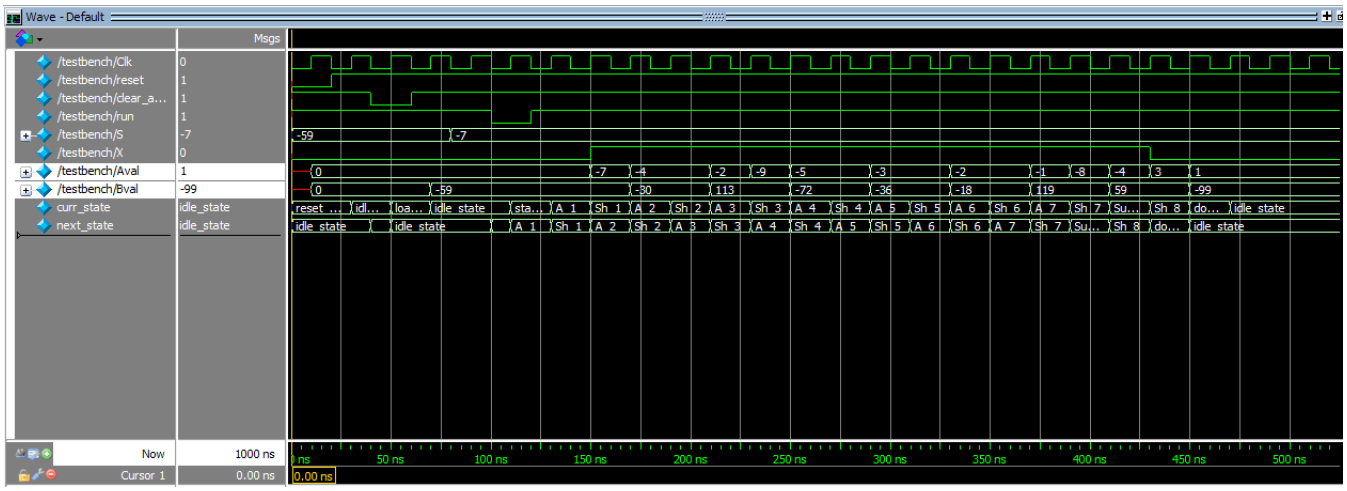

Fig. 4: ModelSim Simulation Output (-7 * -59)

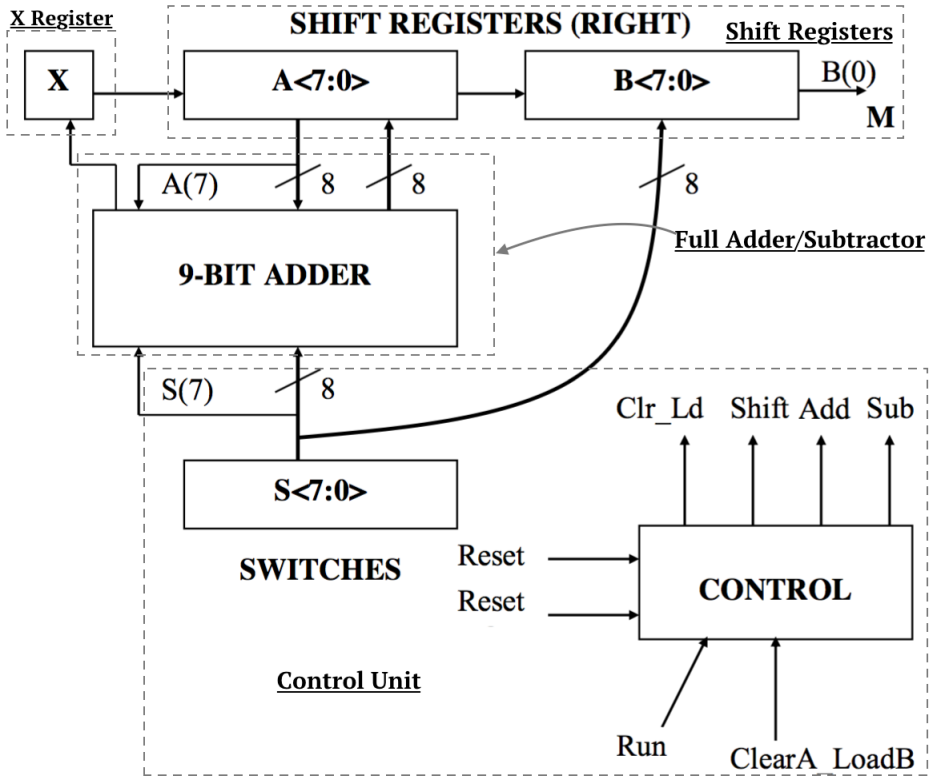

Fig. 5: Incomplete Module Block Diagram

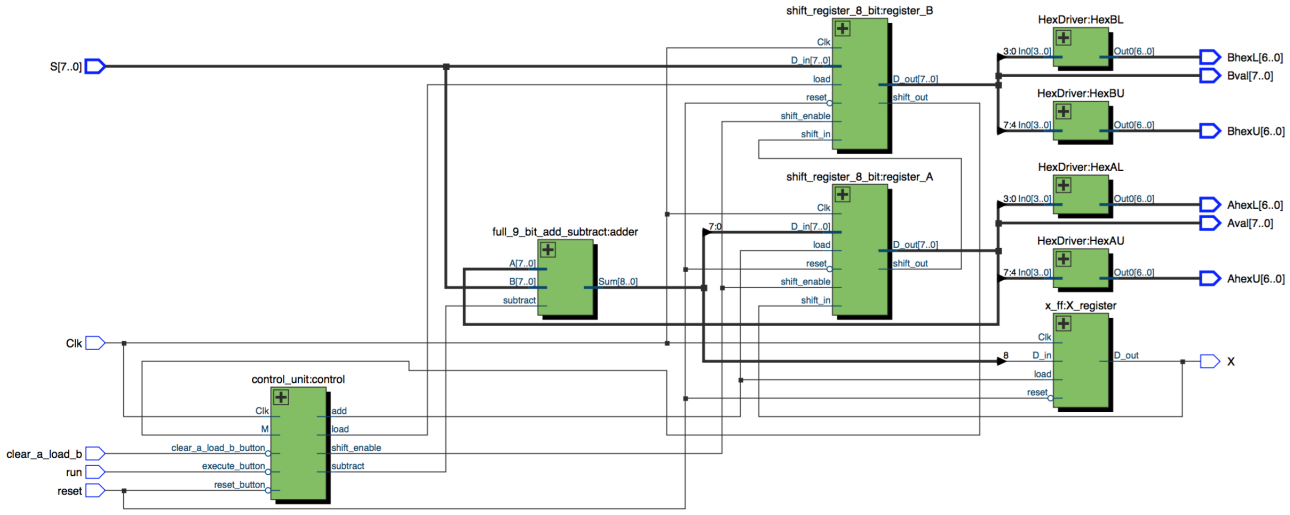

Fig. 6: Full Multiplier Block Diagram

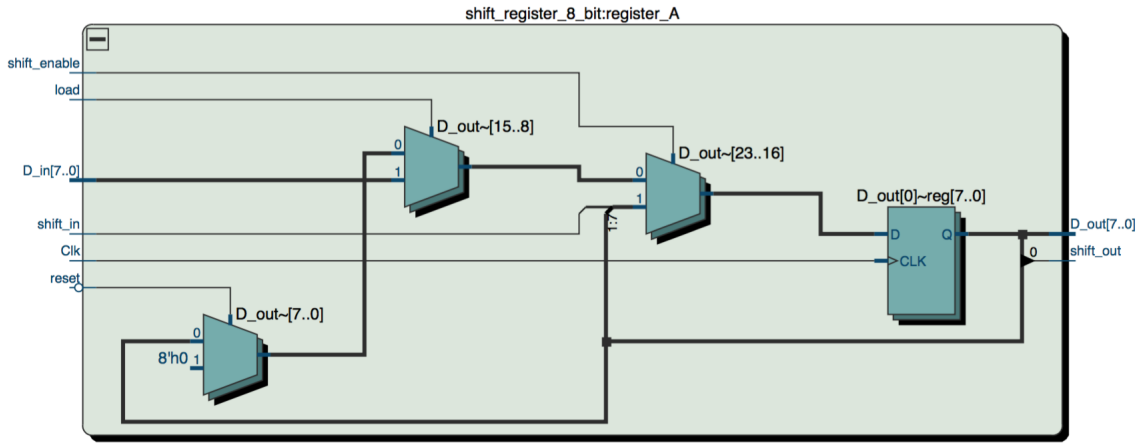

Fig. 7: Shift Register Block Diagram

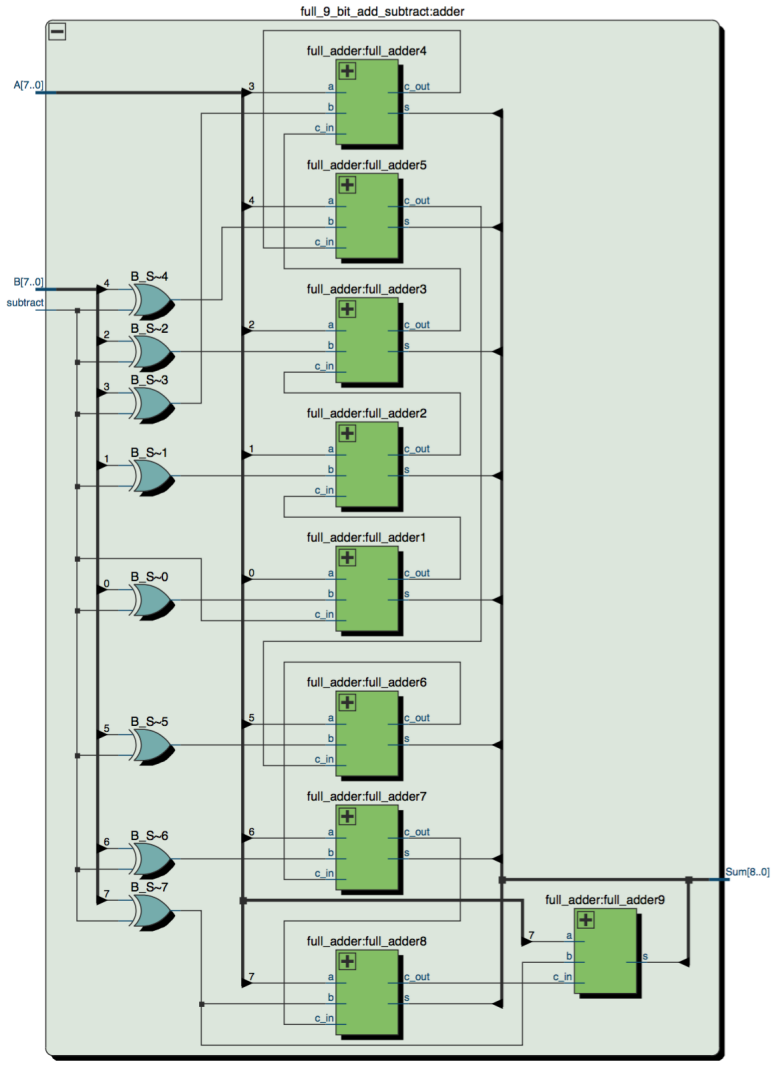

Fig. 8: Full Adder and Subtractor Block Diagram

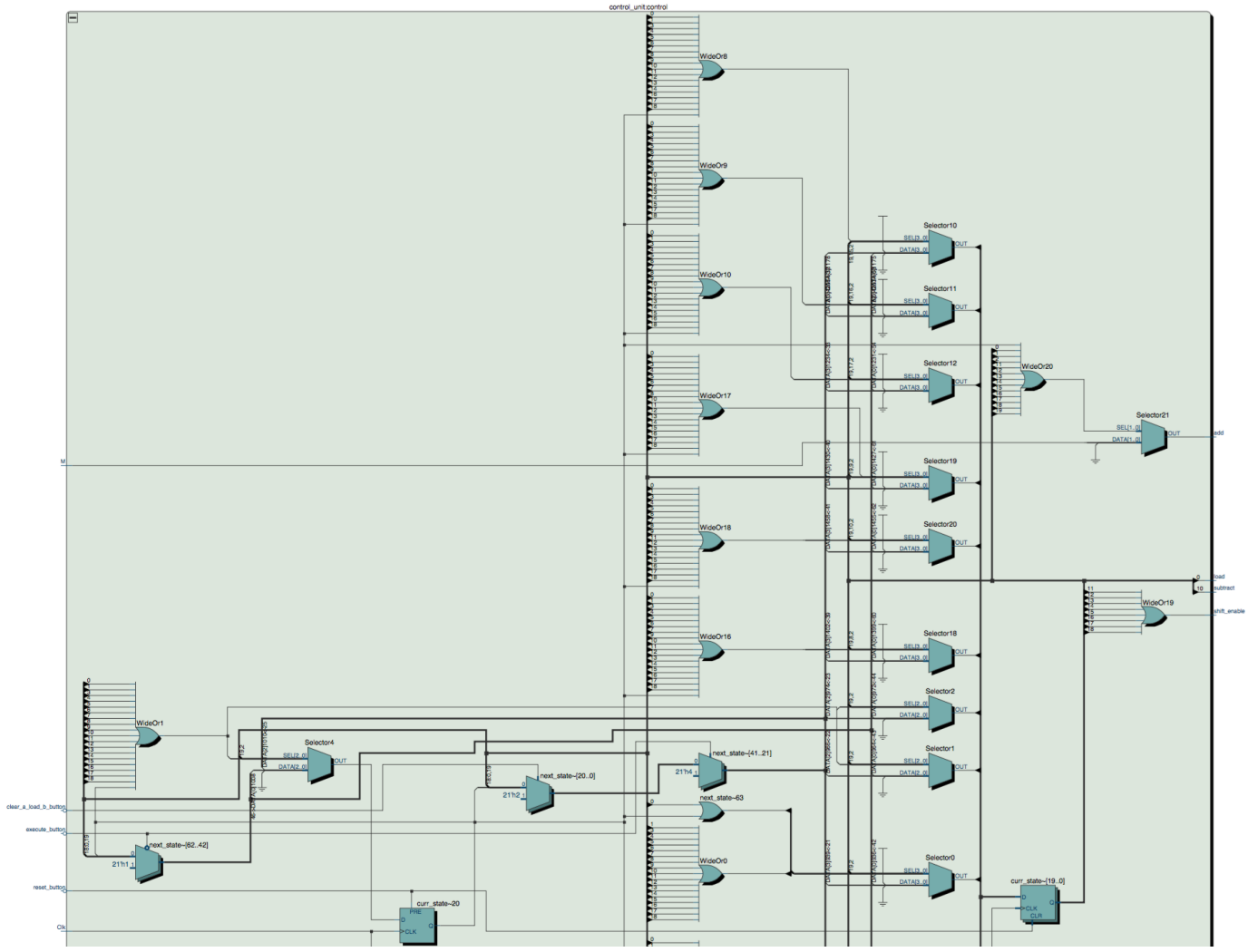

Fig. 9: Top Half Control Unit Block Diagram

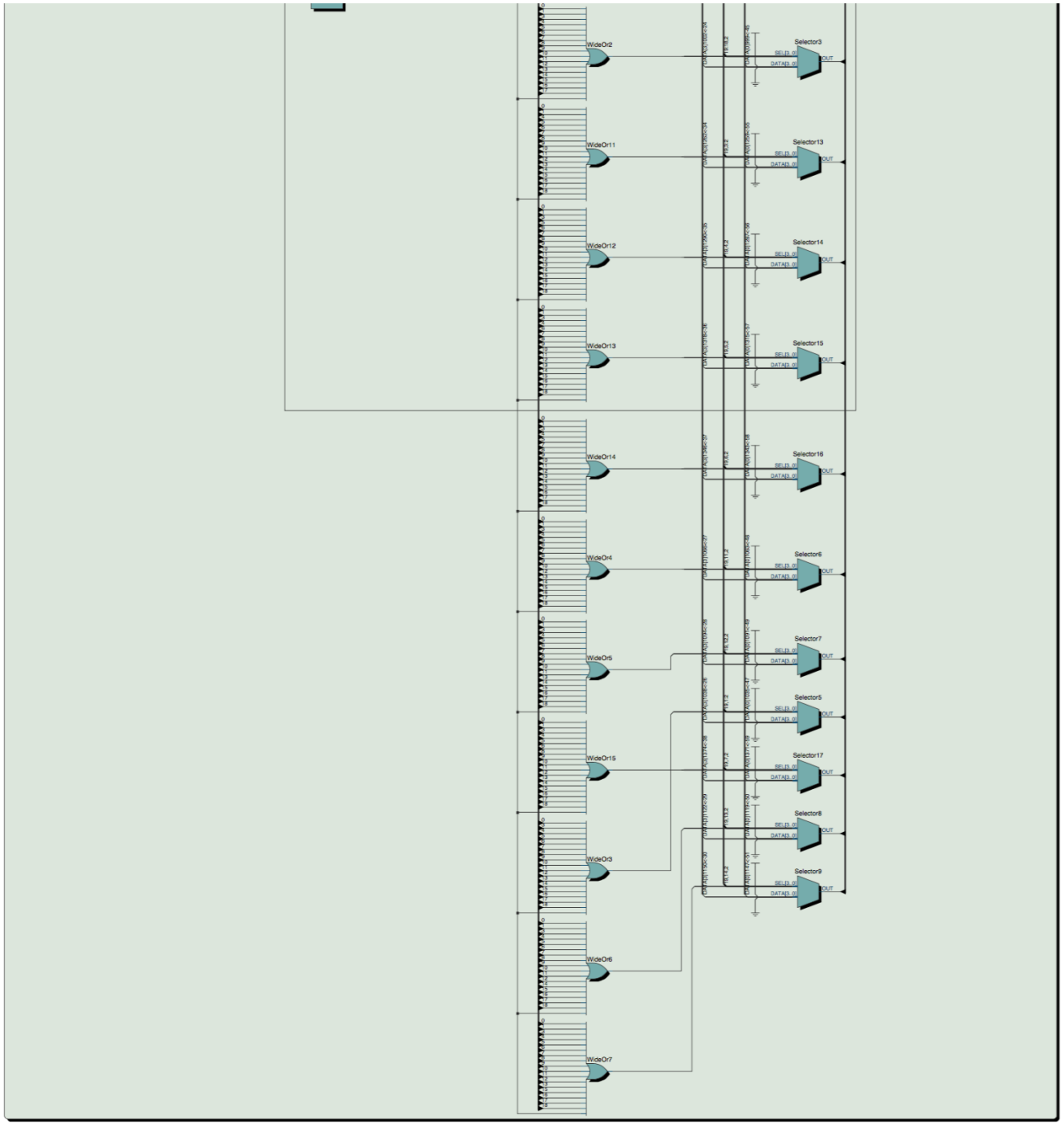

Fig. 10: Bottom Half Control Unit Block Diagram

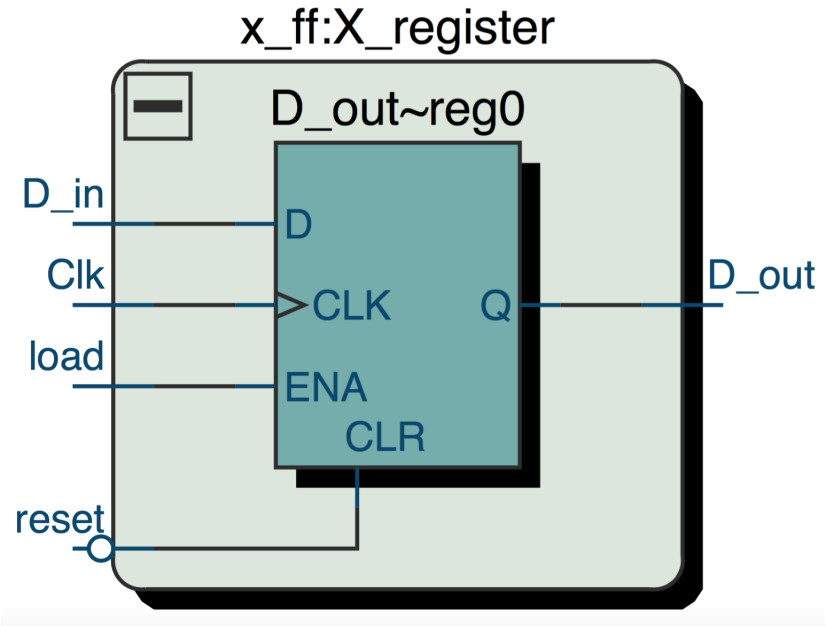

Fig. 11: X-Bit Flip Flop Block Diagram