KAZAKH BRITISH TECHNICAL UNIVERSITY

School of Information Technologies and Engineering

Laboratory work 5:

Yerkebulan Kussym
student ID: **21B030693**

Submitted to: Ayana Mussabayeva

Almaty, 2023

1 Preparation to Lab Work

2 Experiment

3 Test

1. What is a half-(full-) adder?

a digital circuit that performs addition of two single binary digits (bits). It has two inputs - one for each bit - and two outputs: a sum output and a carry output.

2. Show the truth table for half-(full-) adder

Input A	Input B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

3.Show the algebraic expressions for sum and carry for half-(full-) adder?

Let A and B be the input bits, and S and C be the sum and carry outputs, respectively. S = A XOR B (where XOR represents the exclusive-OR operation) C = A AND B The algebraic expressions for the sum and carry outputs of a full-adder can be derived in a similar way: Let A, B, and Cin be the input bits, and S and Cout be the sum and carry outputs, respectively. S = (A XOR B) XOR Cin Cout = (A AND B) OR ((A XOR B) AND Cin) In the expressions for Cout, the OR operation represents the carry generated by the addition of the two input bits, and the AND operation represents the carry propagated from the previous addition (if there is any).

4. What is a binary parallel adder?

A binary parallel adder is a digital circuit that performs addition of two binary numbers in parallel. This means that all the corresponding bits of the two numbers are added simultaneously, rather than sequentially as in a ripple-carry adder.

A binary parallel adder consists of multiple full-adders connected in par-

allel. The input bits are fed into the individual full-adders, and the carry outputs of each full-adder are connected to the carry input of the next full-adder. The sum outputs of each full-adder are combined to give the final sum output of the binary parallel adder.

The advantage of a binary parallel adder over a ripple-carry adder is that it can perform the addition of two n-bit numbers in a single clock cycle, whereas a ripple-carry adder takes n clock cycles to perform the same operation. This makes the binary parallel adder much faster than the ripple-carry adder for larger numbers. However, the binary parallel adder requires more hardware and is more complex than the ripple-carry adder.

5. What is a serial adder?

A serial adder is a digital circuit that performs addition of two binary numbers in a sequential or serial manner, one bit at a time. The two input binary numbers are fed into the circuit bit-by-bit, starting with the least significant bit (LSB), and the bits are added one by one until the most significant bit (MSB) is reached.

A serial adder typically consists of a single full-adder connected to a shift register. The input bits of the two binary numbers are fed into the shift register, and the output of each full-adder is connected to the next stage of the shift register. The output of the last full-adder is the sum of the two input numbers.

The advantage of a serial adder is that it requires less hardware and is simpler than a parallel adder. However, it takes longer to perform the addition of two n-bit numbers compared to a parallel adder, as it requires n clock cycles to complete the addition. Moreover, the serial adder is not suitable for addition of large numbers due to its slow speed. Serial adders are commonly used in low-power, low-speed applications such as in microcontrollers, sensors, and simple digital circuits.

6. How many adders are needed to construct 7-bit parallel adder?

To construct a 7-bit parallel adder, we need 7 full-adders, one for each bit position. Each full-adder adds the three corresponding bits from the two input numbers (A and B) and the carry from the previous full-adder (if there

is any), and produces a sum bit and a carry bit as outputs.

The 7 full-adders are connected in parallel, with the carry-out of each full-adder connected to the carry-in of the next full-adder. The carry-in of the first full-adder is typically set to zero or the carry-in from a previous operation.

Therefore, we need 7 full-adders to construct a 7-bit parallel adder.

7. How many bits have typical full-adders ICs got?

The number of bits in a typical full-adder IC (integrated circuit) depends on the specific IC in question. However, full-adder ICs are typically available in standard bit sizes, such as 4-bit, 8-bit, 16-bit, and so on.

For example, the popular 74LS283 4-bit full-adder IC contains four full-adders, each capable of adding two 4-bit binary numbers with a carry input. Other common full-adder ICs include the 74LS283 8-bit full-adder, the 74LS381 8-bit arithmetic logic unit (ALU) with two full-adders, and the 74LS283 16-bit full-adder.

There are also full-adder ICs available with different logic families and voltage levels, such as TTL (Transistor-Transistor Logic), CMOS (Complementary Metal-Oxide-Semiconductor), and ECL (Emitter-Coupled Logic). The specific number of bits in a full-adder IC will depend on the manufacturer and the intended use of the IC.

8. How to connect IC's full-adders if one package is not enough?

If you need to perform addition of binary numbers with more bits than can be accommodated by a single package of full-adder ICs, you can use multiple packages of full-adder ICs and connect them together in a way that enables them to work together to add the numbers.

One common way to connect multiple packages of full-adder ICs is to use a technique called "ripple carry addition." In this method, each full-adder IC in the chain takes a single bit of the two numbers being added, as well as a carry input from the previous full-adder, and produces a single bit of the sum and a carry output that is passed to the next full-adder in the chain.

To connect multiple full-adder ICs using ripple carry addition, follow these steps:

Connect the A inputs of each full-adder IC in the chain to the correspond-

ing bit of the first number being added, and connect the B inputs to the corresponding bit of the second number being added. Connect the carry-in input of the first full-adder IC to a known value (usually 0). Connect the sum output of each full-adder IC to the corresponding bit of the output result. Connect the carry output of each full-adder IC to the carry-in input of the next full-adder IC in the chain. Connect the carry output of the last full-adder IC in the chain to a designated carry output pin on the circuit. Repeat these steps for each additional package of full-adder ICs, connecting the carry output of the last full-adder IC in the previous package to the carry-in input of the first full-adder IC in the next package. The total number of full-adder ICs required to add two n-bit numbers using ripple carry addition is n, assuming that each full-adder IC is a single-bit full-adder.

Note that while ripple carry addition is a simple and common technique for connecting multiple full-adder ICs, it is not the fastest or most efficient method for adding large numbers, as the carry signal has to propagate through the entire chain of full-adders. Other techniques, such as carry-lookahead addition or carry-select addition, can be used to speed up the process of adding large numbers.

9. What is a look-ahead carry generator?

A look-ahead carry generator is a digital circuit that is used to speed up the process of binary addition by generating carry signals in advance, rather than waiting for the carry signal to ripple through a chain of adders.

In a typical ripple-carry adder, the carry signal is generated at each stage by the combination of the input bits and the carry signal from the previous stage. This means that the carry signal has to ripple through the entire chain of adders, which can lead to significant delay when adding large numbers.

A look-ahead carry generator, on the other hand, generates the carry signals for each stage in advance, based on the input bits and the carry-in signal. This allows the carry signal to be propagated through the adder much more quickly, as the carry signals for each stage are available at the same time as the input bits.

There are several different techniques that can be used to implement a look-ahead carry generator, but one common approach is to use a series of logic gates to generate the carry signals for each stage based on the input bits and the carry-in signal. This can involve a significant amount of logic, but it results in much faster addition times for large numbers.

Look-ahead carry generators are commonly used in high-speed arithmetic circuits, such as microprocessors and digital signal processors, where fast addition is critical for performance.

10. What functions can look-ahead carry generator produce?

A look-ahead carry generator produces carry signals for each stage of a binary adder. The carry signals are used to propagate carry bits from one stage to the next, allowing the adder to correctly add multi-bit binary numbers.

In general, a look-ahead carry generator can produce the following functions:

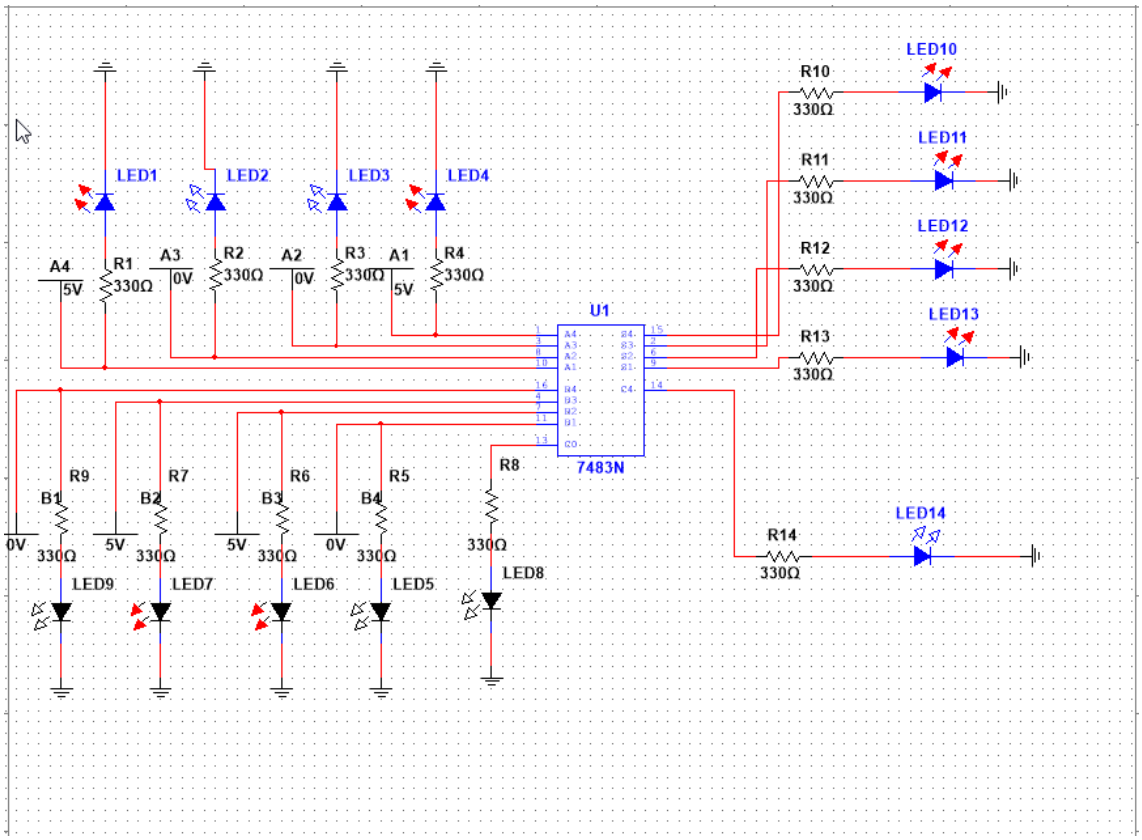
Generate Carry (G): This function indicates whether a carry will be generated at the current stage, based on the input bits and the carry-in signal.

Propagate Carry (P): This function indicates whether a carry signal from the previous stage will be propagated to the current stage, based on the input bits.

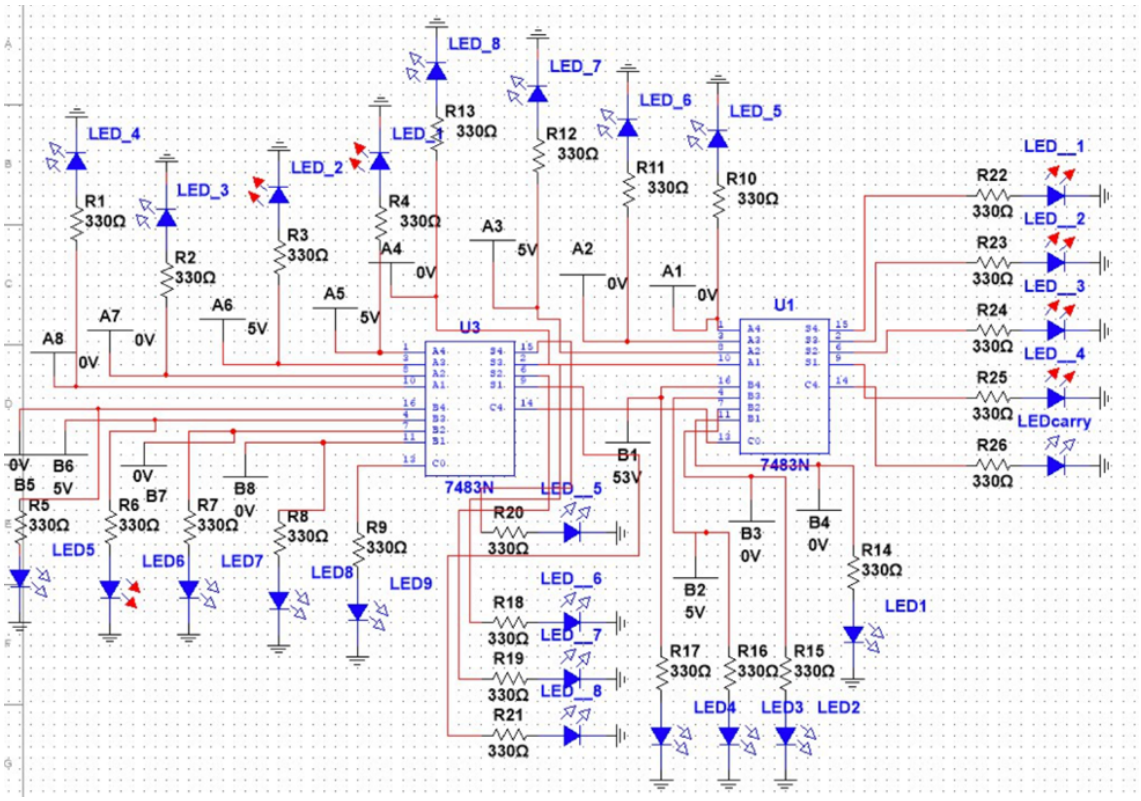
Kill Carry (K): This function indicates whether a carry signal from the previous stage will be killed (or blocked) at the current stage, based on the input bits.

Generate and Propagate (G and P): This function indicates whether both a carry will be generated and a carry signal from the previous stage will be propagated to the current stage, based on the input bits and carry-in signal.

The functions produced by a look-ahead carry generator depend on the specific design of the circuit and the number of bits being added. In general, the goal is to minimize the amount of delay in the carry propagation, which can be achieved by generating the carry signals for each stage in advance based on the input bits and carry-in signal.



Numbers for addition								inputs									outputs				
A1	A2	A3	A4	B1	B2	B3	B4	L1	L2	L3	L4	L5	L6	L7	L8	L9	L 10	L 11	L 12	L 13	L 14
1	1	0	0	1	1	0	1	on	on	off	off	on	on	off	on	off	1	0	0	1	1
0	0	0	1	1	1	1	1	off	off	off	on	on	on	on	on	off	0	0	0	0	1
1	0	0	1	0	1	1	0	on	off	off	on	off	on	on	off	off	1	1	1	1	0



Numbers for addition																outputs								
A1	A2	A3	A4	A5	A6	A7	A8	B1	B2	B3	B4	B5	B6	B7	B8	L 1	L 2	L 3	L 4	L 5	L6	L7	L8	L(c)
0	0	1	0	1	1	0	0	1	1	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1

1.B

2.E

3.E

4.B

5.D

6.C

7.E

8.C

9.A

10.C