Queen Mary
University of London

ECS404U: COMPUTER SYSTEMS & NETWORKS

2020/21 – Semester 1

Prof. Edmund Robinson, Dr. Arman Khouzani

Lab 5: *Constructing an Adder, Constructing a Memory Unit, Memory Hierarchy, Text Representation in Binary, Marking Exercise*

October 19, 2020

Deadline for submitting your proof of work: ***Next week's Thursday, at 10:00 AM UK time***

Student Name:

Yujeong JUNG

Student ID:

200699145

Brief Feedback to student (commendations, areas for improvement):

Learning Objectives

- Understand how by combination of logic gates (and hence, transistors), we can construct device units that can carry out arithmetic operations with the specific example of addition.
- Be familiar with the basic circuitry of an SRAM memory cell.
- Have looked at the timings of different sorts of memory.
- Have seen some examples of student answers and matched them to their feedback.

1 Building an Adder

In our previous lab, we learned how to use transistors to build the first building blocks of computers, i.e., logic gates (like `and`, `or`, etc.). Although very impressive, it still doesn't feel like a computation! In this exercise, we see how we can build a unit that can carry out what is unmistakably a computation: addition of two numbers. The trick is to observe that in binary, the addition can be broken into a series of logic operations, which we already know how to make using transistors.

1.1 Half adder

Learning Objective Ensure you understand how to construct a half adder for single bit addition out of two logic gates (and hence, putting next to what we learned from last week, using transistors).

1. Write out the binary addition table, and split it into the least significant bit and the carry bit.

A	B	A+B	least bit	carry bit
1	1	10	0	1
1	0	01	1	0
0	1	01	1	0
0	0	00	0	0

   Identify the logical connectives that correspond to the least bit and the carry bit:

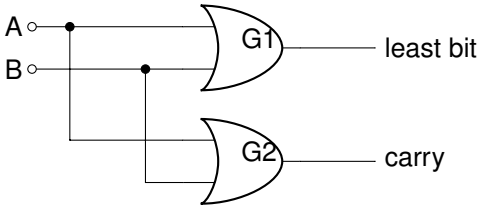
   least bit: xor

   carry bit: and

2. Explain how you would use logic gates, such as the ones you have just constructed, or similar, to build a half adder. The function of the half adder is to take two inputs A and B, and to produce two outputs, result and carry, the two bits of the result of adding the inputs. So if A and B are both 1, then the result should be 0 and the carry should be 1.

   In the diagram below, which implements a half adder, we have used or gates in place of the gates really needed. Which gates should we actually use?

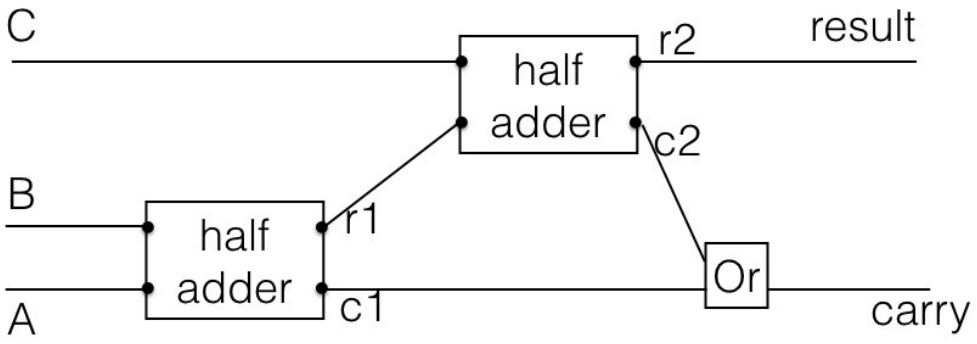
G1	xor
G2	and



1.2 Full adder

Learning Objective Ensure you understand the construction of a full adder out of two half adders.

3. The half adder from the last question will calculate the result of adding two bits. In order to build an addition circuit we also have to cope with the possibility of an incoming carry as a third input. A full adder will do that. We can build a full adder out of two half adders:

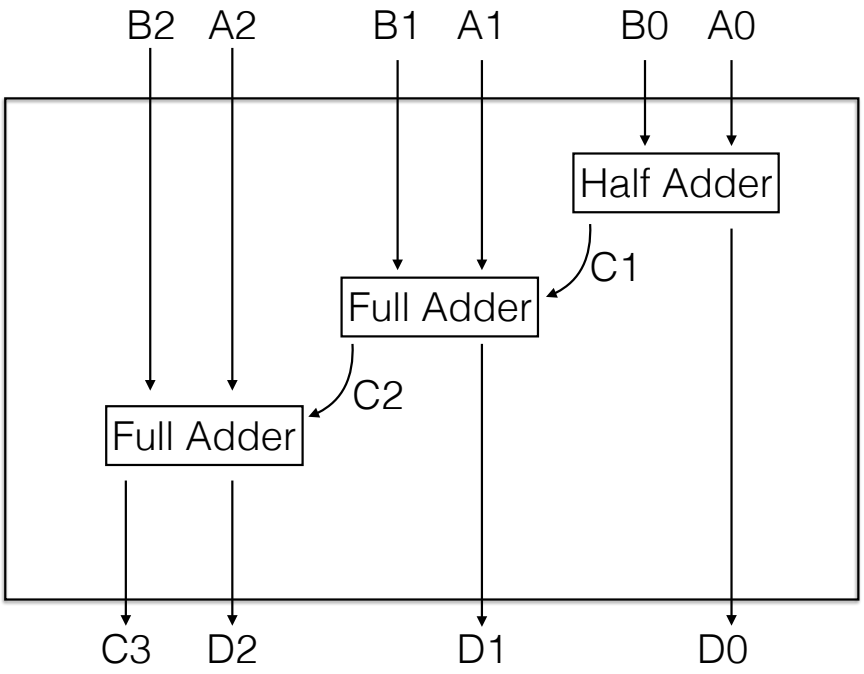


Verify that this works by checking the bit values at r1,c1,r2,c2, result and carry for the following inputs:

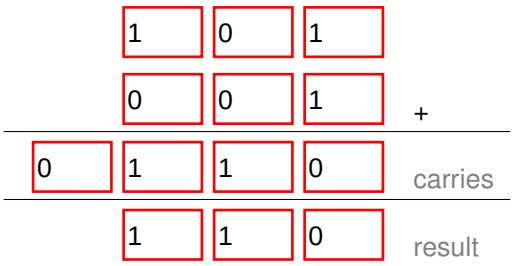
A	B	C	r1	c1	r2	c2	result	carry
1	1	1	0	1	1	0	1	1
1	1	0	0	1	0	0	0	1
0	1	1	1	0	0	1	0	1

1.3 A simple 3-bit adder

4. We can chain single bit adders together in order to build a full n-bit adder.



Explain how this corresponds to long addition by relating it to paper-based long addition using the example of $101 + 001$. Fill in the values of the bits in the wires of the adder above, and check that the result is a valid long addition.



2 SRAM Memory

So far, we have seen how to make units that can carry out logic and arithmetic operations. We made this marvellous achievement using transistors as the ultimate building blocks of all computers! But wait, computers need more: in order to do carry out more complicated operations, we need to store temporary intermediate results. For example, the moment we change the input bits to an adder, the outputs will change too. So the previous result is gone. Not good! Technically, we have implemented the parts of ALU: Arithmetic Logic Unit. But we also need Memory, to be able to at least temporarily store the results. Here, we see how to make a memory unit, amazingly, using transistors again! Transistors, for the win!

To achieve this, we first start by investigating the kind of logic gates we have done before. Next, we study a circuit that can be used to store information 1 bit of information. In particular, as a one bit memory, our device needs to do three things:

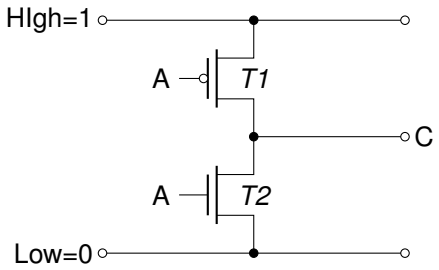
1. We should be able to “write” (i.e. “set”) the value of the bit;
2. It should hold the value of the bit after the bit is written, forever, until the next time a new value is written to it;
3. We should be able to “read” the stored value whenever we want;

We will see how our construction achieves each of these tasks.

5. The diagram below gives a simple gate design with a single input. For each value of the input A (1 or 0, representing high and low), say whether the transistors T1 and T2 are open (no connection) or closed. (connection). Hence say which of the four conditions holds:

   - C is connected to High and not Low, hence C=1
   - C is connected to Low and not High, hence C=0
   - C is connected to neither High nor Low, hence C is undetermined
   - C is connected to both High and Low, hence there is a destructive short

   A gate is valid if C is always determined and there is never a short. Say whether each gate is valid or not, and if not why not.



A	T1	T2	C
0 or 1	closed or open	closed or open	0 or 1 or undef or short
1	open (disconnected)	closed (connected)	0
0	closed (connected)	open (disconnected)	1

Valid? yes

This circuit is called an *inverter*. Why do you think that might be?

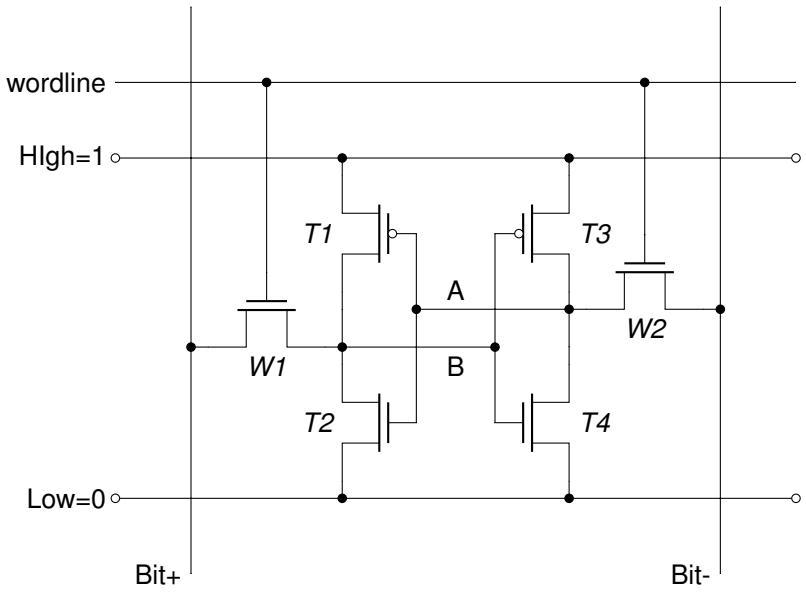
Ans:

In this circuit there is only input at A and the output is at C.
The output at C is connected middle of rail between T1 and T2

What logical connective does it implement?

Ans: and

6. This question moves on from logic gates. Transistor-based circuits can also be used to store data. The diagram below shows an SRAM memory cell which can store a single bit. The data is effectively held using the gates of transistors T1,T2,T3,T4. Transistors W1 and W2 control access to the word line.



(a) Identify two inverters (not gates - in the sense of being gates that compute the negation of their input) in the diagram by listing the transistors in each:

First inverter	T1	T2
Second inverter	T3	T4

(b) How are the input and output of these inverters linked?

This circuit has 6 transistor; two CMOS inverts and two access MOSFETs. (nmos).
SRAM is a structure with one transistor on each side of the flipflop.
So, inputs are at A and B, and outputs are at W1 line and W2 line.

(c) What does this tell you about the potentials at A and B?

This tell about the predictable status.
It means that if we input 0,0, then its behavior will be unpredictable.
But, if we input the one of 0,1 or 1,0 or 1,1, its behavior will be predictable completely.

(d) When the wordline is driven low, are either of the bit-lines connected to either of A or B?

No, it can't be connected.

(e) When the wordline is driven low, describe the two possible stable states of the cell in terms of the potentials at A and B and the consequent states of the transistors T1, T2, T3 and T4?

When A=1, B=0, T3 and T2 are on but T1 and T4 are off.
When A=0, B=1, T3 and T2 are off but T1 and T4 are on.

(f) Confirm the stability of each of the states you have just given, i.e. check that A and B are both connected to the appropriate potentials.

When A is 1 and B is 0, wordline has high bit as driver then the structure of circuit is flipped and be stable.

(g) When the wordline is driven high, are either of the bit-lines connected to either of A or B?

Either of bit-lines are connected to either of A or B.

(h) When the wordline is driven high, Bit- is driven at 0 and Bit+ is driven High, then what happens to the transistors T1-T4?

T2 and T3 will be off and T1 and T4 will be on.

3 Comparative timings for memory levels

7. Recall access timings for memory levels as below. Complete the table to include the average access times, given as the average of the two figures. Also include the number of clock cycles this represents, assuming a cpu working at 2GHz.

Technology	Access time (min)	Acces time (Max)	Access time (avge)	Clock cycles
SRAM semiconductor	0.5ns	2.5ns	1.5 ns	10
DRAM semiconductor	50ns	70ns	60 ns	100
Flash semiconductor	5,000	50,000ns	27,500 ns	100,000
Magnetic disk	5,000,000ns	20,000,000ns	12,500,000 ns	1,000,000,0

By what factor is SRAM access faster than DRAM?

CPU

By what factor is RAM access faster than Flash?

volatile

By what factor is Flash access faster then Magnetic Disk?

capacity

4 Coursework exercise

8. The 2016 coursework 1 contains the following part question:

> 1(b) Explain why 11110101 represents -11 in 8 bit 2's complement, and why 00001001 represents +9. Compute the binary multiplication of 11110101 times 00001001 and verify that the result represents -99 (remember to only take the least significant 8 bits. Marks will be given for correct working and explanation.

This exercise is to help you to understand how to answer coursework and exam questions.

Here are three student solutions, followed by three feedback examples. None of the solutions are perfect. **Match the submission to the feedback.** The feedback should tell you what is being assessed. You should not need the model answer to make comparisons.

Student	Feedback
A	3
B	2
C	1

Student A:

B)

The first binary set determines whether it is a positive number or a negative number, '1111' says that the number is negative, so then '0000' shows it's positive, so when looking at the two examples, '-11' is negative due to the '1111' and the '+9' is positive due to the '0000'.

The next binary set is where the confusion may set it, but, to briefly explain it is the normal '0000', '0001', '0010' but in reverse, so instead of '1111' correlating to '16' it actually correlates to '1', this means that '0000' actually correlates to '16', this is only the case when it is a negative number, so the '-11' is counted from bottom to top (*using the image on the previous page*), so instead of '11' being '1011' it actually becomes '0101' in reverse.

The '9' however stays the same because it is positive.

1111 0101 = – 11

0000 1001 = +9

When multiplying a positive and a negative number it will always results in a negative number, so then multiplying '-11' and '+9' it will come out as a negative number and as '11 * 9 = 99' it means that '-11 * +9 = - 99'

1111 0101 x 0000 1001 = -99

Student B:

b) i) 1111 0101 – It starts at 1, which means it represents a negative. In this case, all the 1's and 0's need to be switched around resulting as 0000 1010. Then add 1 to the value which results as 0000 1011. The binary is then converted to decimal using this table.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

8 21
0000 1011
$2^0 + 2^1 + 2^3 = -11$

1 + 2 + 8 = -11

ii) 0000 1001 – it starts to 0 therefore it represents a positive. In this case the value does not need to be switched around.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

8 1
0000 1001
$2^0 + 2^3 = 9$
1 + 8 = +9

iii)
```
1111 0101
0000 10001   x
_____________
1111 0101
0000 0000
0000 0000
1111 0101        (+)
_________________
100010011101
```

100010011101 – first 8 digits from the right

In 2's complement, you always consider the first 8 digits as it is 8 bits.

01100010 + 1
= 01100011
$=2^0+2^1+2^5+2^6$
=1+2+32+64
=1+2+96
=99
therefore 1001110 – left most digit is 1, initially it is a negative
therefore this results as -99

Student C:

1. Get the binary representation in one's complement, minus 1 from the binary initial number: 1111 0101 - 1 = 1111 0100 2. Get the binary representation of the positive number, flip all the bits in the binary one's complement representation - replace the bits set on 1 with 0-s and the bits set on 0 with 1-s: !(1111 0100) = 0000 1011 3. Positive value of base 10 integer number: 0000 1011(2) = 0 * 27 + 0 * 26 + 0 * 25 + 0 * 24 + 1 * 23 + 0 * 22 + 1 * 21 + 1 * 20 = 0+0+0+0+8+0+2+1= 8+2+1= 11(10) 1111 0101 converted from binary two's complement to signed integer in decimal system (in base 10) = -11(10)

Feedback Examples:

—bf Feedback 1:

b)
Assessment: Good explanation of how to translate negative 2's complement to decimal.
No explanation of how to translate positive 2's complement to decimal was given.
Binary multiplication not attempted.
Verification of result not attempted.
Marker's feedback: Only explained -11, no binary multiplication or verification

Marks: 4

Feedback 2:

b)
Assessment: Good explanation of how to translate negative 2's complement to decimal.

Good explanation of how to translate positive 2's complement to decimal.
Binary multiplication was clear and accurate
Verification of result was clear and accurate
Marker's feedback: Good explanations and working

Marks: 12

Feedback 3:

b)
Assessment: Good explanation of how to translate negative 2's complement to decimal.
Good explanation of how to translate positive 2's complement to decimal.
Binary multiplication not attempted.
Verification of result not attempted.
Marker's feedback: Good explanations for negative and positive numbers, but no binary multiplication or verification was attempted

Marks: 8

End of questions