2. Projects #2: Làm bài tập tại mục 8.4 (HDTH KTMT bằng mô phỏng) với một số yêu cầu bổ sung.

(Bùi Tiến Đạt; Hà Minh Hải; Trần Đức Hiếu)

(After finishing: save your logisim simulation circuit as: Project#2.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

Hãy thiết kế bằng mô phỏng bảng hiển thị điện tử kích thước 8 hàng x 32 cột và 2 hiện dòng chữ:

- Dòng 1: "DAI HOC QUOC GIA HA NOI" ở chính giữa dòng trên cùng
- Dòng 2: "TRUONG DAI HOC CONG NGHE" ở chính giữa dòng thứ 2.

Yêu cầu từng ký tự của dòng chữ thứ nhất rồi đến dòng thứ 2 được hiện lên lần lượt, sau khi hiển thị đến ký tự cuối cùng của dòng thứ 2 thì xóa toàn bộ và lặp lại quá trình trên.

;#===

3. Projects #3: Làm bài tập tại mục 9.7 (HDTH KTMT bằng mô phỏng)

(Dương Thị Thúy Hằng; Lê Minh Tâm; Phạm Ngọc Anh Trang)

(After finishing: save your logisim simulation circuit as: Project#3.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

Hãy thiết kế một đồng hồ điện tử hiện số, với các yêu cầu sau:

1. Thời gian được hiển thị dưới dạng: hh : mm : ss, trong đó:

- hh là số chỉ giờ (hour), tăng từ 0 đến 11 rồi lại trở về đếm từ 0.
- mm là số chỉ phút (minute), tăng từ 0 đến 59 rồi lại trở về đếm từ 0.
- ss là số chỉ giây (second), tăng từ 0 đến 59 rồi lại trở về đếm từ 0.
- Giữa các số chỉ thời gian nói trên có dấu ":", nhấp nháy 1 giây 1 lần.

2. Mỗi chữ số được hiển thị bằng 1 phần tử "Hex Digit Display" thuộc nhóm Input/Ouput trong thư viện của logisim.

;#===

4. Projects #4: Làm bài tập tại mục 9.7 (HDTH KTMT bằng mô phỏng) với một thay đổi nhỏ trong yêu cầu.

(Nguyễn Quang Huy; Hoàng Sơn Tùng; Nguyễn Ngọc Thanh Tùng)

(After finishing: save your logisim simulation circuit as: Project#4.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

Hãy thiết kế một đồng hồ điện tử hiện số, với các yêu cầu sau:

1. Thời gian được hiển thị dưới dạng: hh : mm : ss, trong đó:

- hh là số chỉ giờ (hour), tăng từ 0 đến 23 rồi lại trở về đếm từ 0.
- mm là số chỉ phút (minute), tăng từ 0 đến 59 rồi lại trở về đếm từ 0.
- ss là số chỉ giây (second), tăng từ 0 đến 59 rồi lại trở về đếm từ 0.
- Giữa các số chỉ thời gian nói trên có dấu ":", nhấp nháy 1 giây 1 lần.

2. Mỗi chữ số được hiển thị bằng 1 phần tử "Hex Digit Display" thuộc nhóm Input/Ouput trong thư viện của logisim.

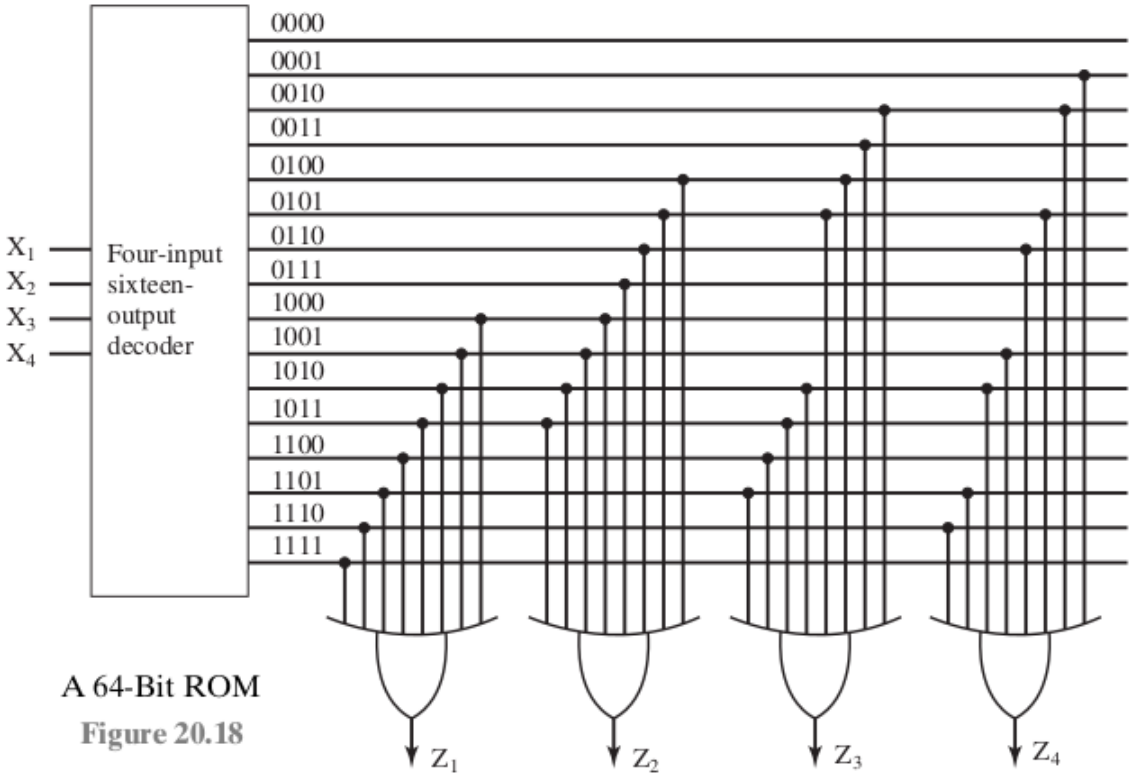
6. Projects #6: Use Logisim digital circuit simulator to construct a 8x8-bit ROM shown on the Figure 20.18 A 64-Bit ROM.

(Nguyễn Duy Khánh; Khổng Thị Mai Loan; Trần Việt Tiến; Đỗ Quốc Trung; Hà Minh Tuấn)

(After finishing: save your logisim simulation circuit as: Project#6.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010, page 20-22). (Hãy sử dụng bộ mô phỏng mạch điện số Logisim để xây dựng một bộ nhớ ROM 8x8 bit được thể hiện trên Hình 20.18 64-bit ROM. (Sách: ...)).

- Add necessary functional units to the circuit to display address and the output number in Hexa and Decimal format. *(Hãy bổ sung các đơn vị chức năng cần thiết vào mạch điện để hiển thị địa chỉ và số đưa ra dưới dạng Hexa và dạng thập phân).*
- Simulate read operation. *(Hãy mô phỏng hoạt động đọc).*
- Write "User's Guide" to your simulation circuit along with exported image of your simulation circuit. *(Hãy viết "Hướng dẫn cho người sử dụng" đối với mạch mô phỏng của bạn cùng với hình được kết xuất của mạch mô phỏng của bạn).*



A 64-Bit ROM

Figure 20.18

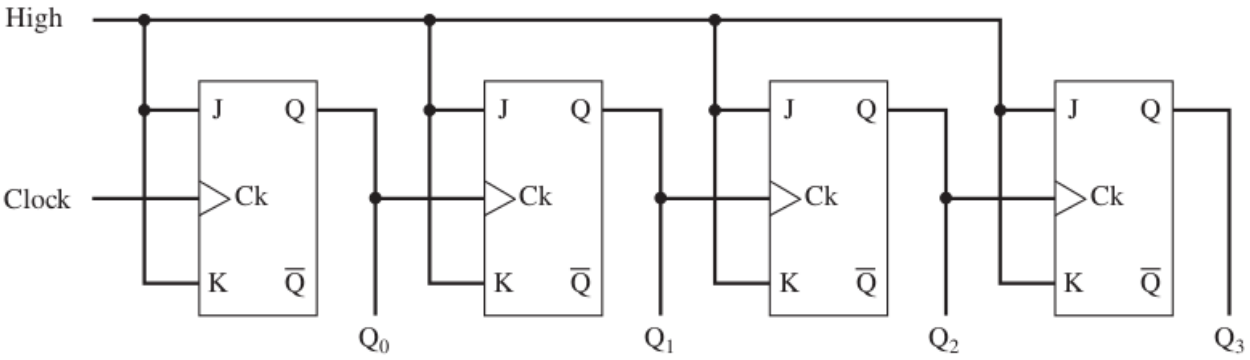
7. Projects #7: Use Logisim digital circuit simulator to construct a 8-bit Ripple Counter similar to the one on the Figure 20.30a Ripple Counter (on the next slide)

(Phạm Quang Anh; Nguyễn Huy Hoàng; Vũ Công Thi)

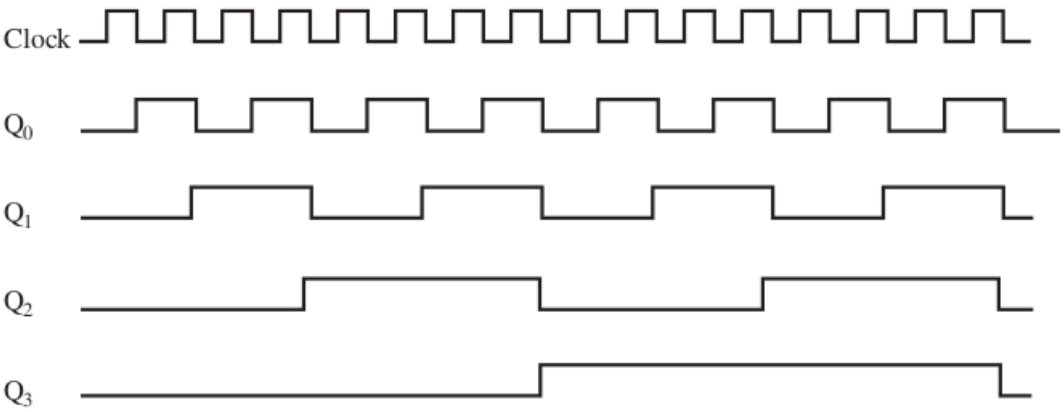
(After finishing: save your logisim simulation circuit as: Project#7.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010). *(Hãy sử dụng bộ mô phỏng mạch điện số Logisim để xây dựng một bộ đếm kiểu gợn sóng 8 bit, tương tự như bộ đếm trên Hình 20.30a Ripple Counter (xem trên slide sau). (Sách: ...)).*

- Add necessary functional units to the circuit to display the output number in Hexa and Decimal format. *(Hãy bổ sung các đơn vị chức năng cần thiết vào mạch điện để hiển thị số đưa ra dưới dạng Hexa và dạng thập phân).*
- Simulate operation of your counter. *(Hãy mô phỏng hoạt động của bộ đếm của bạn).*
- Write "User's Guide" to your simulation circuit along with exported image of your simulation circuit. *(Hãy viết "Hướng dẫn cho người sử dụng" đối với mạch mô phỏng của bạn cùng với hình được kết xuất của mạch mô phỏng của bạn).*



(a) Sequential circuit



(b) Timing diagram

Figure 20.30 Ripple Counter

8. Project #8: To solve problem 20.6 (Book: William Stallings)

(Đỗ Văn Bằng; Nguyễn Hữu Hòa; Trần Quang Huy)

(After finishing: save your logisim simulation circuit as: Project#8.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010; page 20-40): *(Hãy giải bài tập 20.6 (Sách: ...; Trang 20-40):)*

Given a NOR gate and NOT gates, draw a logic diagram that will perform the three-input AND function. *(Hãy xây dựngmột mạch điện thực hiện chức năng AND có 3 đầu vào, chỉ sử dụng các cổng logic NOR và NOT).*

9. Project #9: To solve problem 20.8 (Book: William Stallings)

(Nguyễn Kiều Hưng; Phan Quang Hưng; Lê Cao Tùng Lâm)

(After finishing: save your logisim simulation circuit as: Project#9.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010; page 20-40) *(Hãy giải bài tập 20.8 (Sách: ...; Trang 20-40):)*

A combinational circuit is used to control a seven-segment display of decimal digits, as shown in Figure 20.36. The circuit has four inputs, which provide the four-bit code used in packed decimal representation (010 = 0000, ..., 910 = 1001). The seven outputs define which segments will be activated to display a given decimal digit. Note that some combinations of inputs and outputs are not needed. *(Một mạch tổ hợp được sử dụng để điều khiển một bộ hiển thị kiểu 7 đoạn đối với các số thập phân, như được cho trên hình 20.36. Mạch điện có 4 lối vào, chúng tạo ra mã 4 bit được sử dụng trong cách biểu diễn số thập phân kiểu nén (010 = 0000, ..., 910 = 1001). Bảy lối ra xác định đoạn nào sẽ được kích hoạt để hiển thị một số thập phân đã cho. Chú ý rằng có một số tổ hợp các lối vào và lối ra không được sử dụng.)*

a) Develop a truth table for this circuit. *(Hãy xây dựng một bảng chân lý cho mạch điện này)*

b) Express the truth table in SOP form. *(Hãy biểu diễn bảng chân lý dưới dạng SOP)*

~~c) Express the truth table in POS form.~~

d) Provide a simplified expression. *(Hãy đưa ra một biểu thức được rút gọn)*

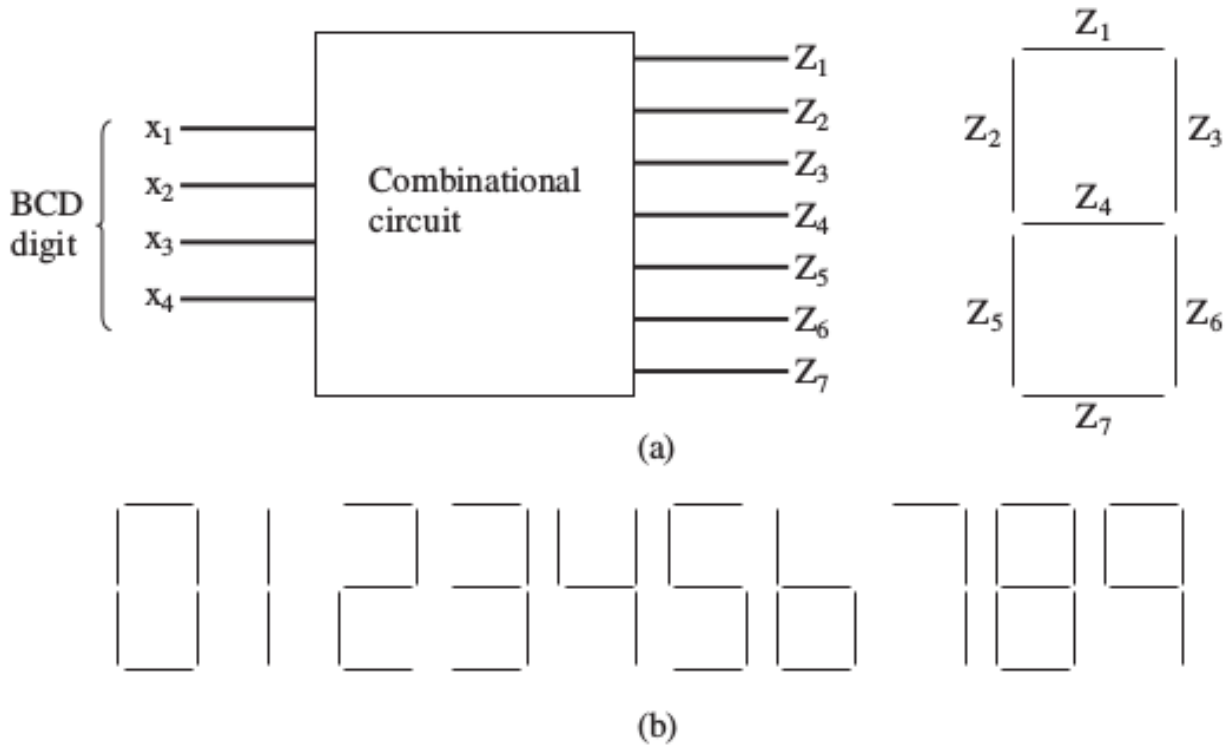


Figure 20.36 Seven-Segment LED Display Example

Hình vẽ 20.36 dùng cho Project #10 và Project #11

11. Project #11: To solve problem 20.8 (Book: William Stallings)

(Nguyễn Minh Đức; Nguyễn Hợp Quang; Đinh Bảo Vương)

(After finishing: save your logisim simulation circuit as: Project#11.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010; page 20-40) *(Hãy giải bài tập 20.8 (Sách: ...; Trang 20-40):)*

A combinational circuit is used to control a seven-segment display of decimal digits, as shown in Figure 20.36. The circuit has four inputs, which provide the four-bit code used in packed decimal representation (010 = 0000, ..., 910 = 1001). The seven outputs define which segments will be activated to display a given decimal digit. Note that some combinations of inputs and outputs are not needed. *(Một mạch tổ hợp được sử dụng để điều khiển một bộ hiển thị kiểu 7 đoạn đối với các số thập phân, như được cho trên hình 20.36. Mạch điện có 4 lối vào, chúng tạo ra mã 4 bit được sử dụng trong cách biểu diễn số thập phân kiểu nén (010 = 0000, ..., 910 = 1001). Bảy lối ra xác định đoạn nào sẽ được kích hoạt để hiển thị một số thập phân đã cho. Chú ý rằng có một số tổ hợp các lối vào và lối ra không được sử dụng.)*

a. Develop a truth table for this circuit. *(Hãy xây dựng một bảng chân lý cho mạch điện này)*

~~b. Express the truth table in SOP form.~~

c. Express the truth table in POS form. *(Hãy biểu diễn bảng chân lý dưới dạng POS)*

d. Provide a simplified expression. *(Hãy đưa ra một biểu thức được rút gọn)*

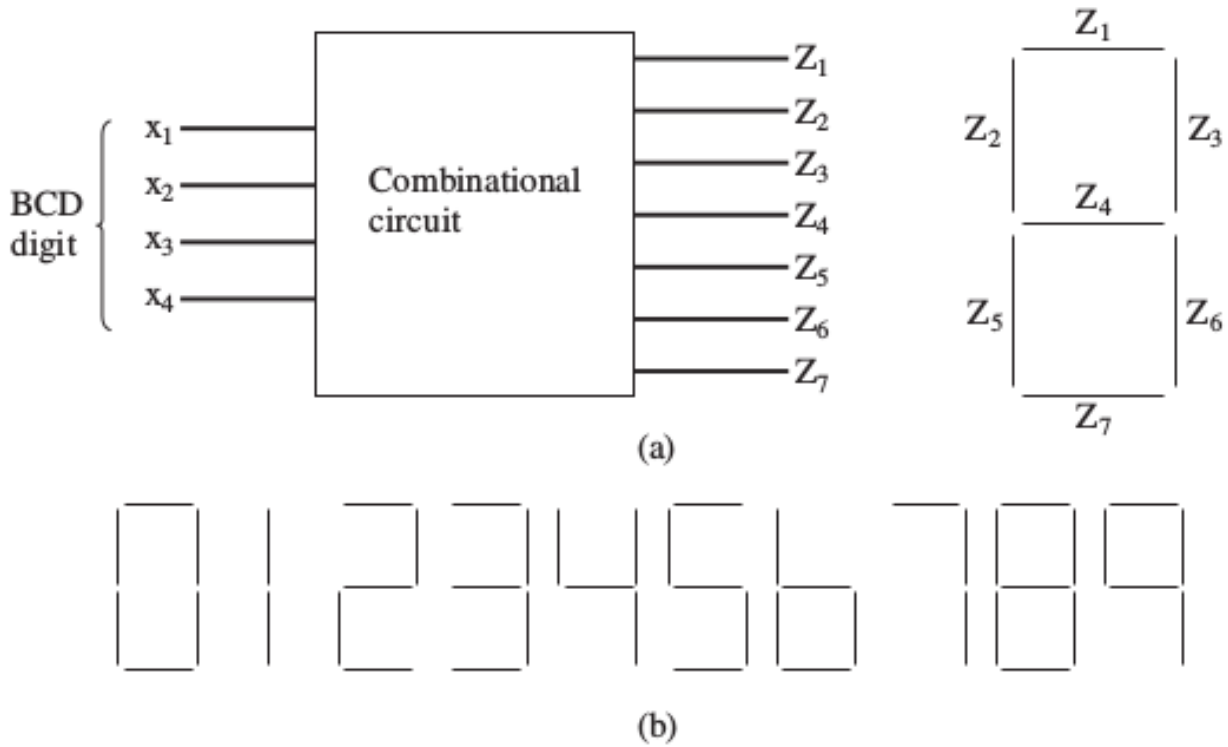


Figure 20.36 Seven-Segment LED Display Example

Hình vẽ 20.36 dùng cho Project #10 và Project #11

12. Project #12: To solve problem 20.10 (Book: William Stallings)

(Phạm Đức Duy; Đỗ Thành Đạt; Nguyễn Hữu Đạt)

(After finishing: save your logisim simulation circuit as: Project#12.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010; page 20-40): Add an additional line to Figure 20.15 so that it functions as a demultiplexer. *(Hãy giải bài tập 20.10 (Sách: ...; Trang 20-40): Hãy bổ sung một đường truyền vào Hình 20.15 sao cho mạch điện có thể hoạt động như một bộ phân kênh)*

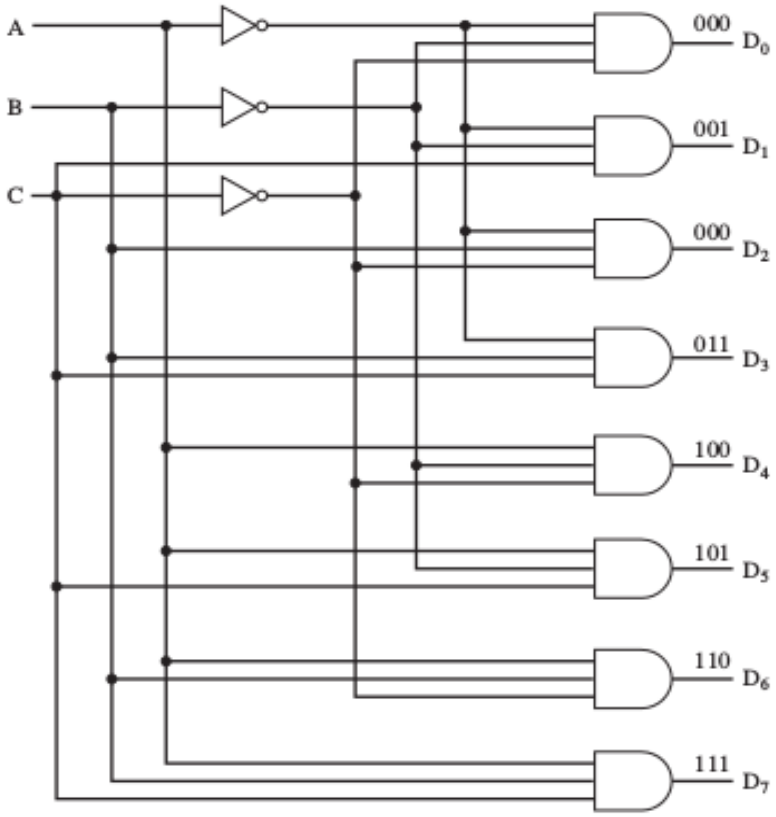


Figure 20.15 Decoder with 3 Inputs and $2^3 = 8$ Outputs

13. Project #13: To solve problem 20.11 (Book: William Stallings)

(Đào Tiến Dũng; Phan Thế Giang; Trần Hoàng)

(After finishing: save your logisim simulation circuit as: Project#13.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010; page 20-40): The Gray code is a binary code for integers. It differs from the ordinary binary representation in that there is just a single bit change between the representations of any two numbers. This is useful for applications such as counters or analog-to-digital converters where a sequence of numbers is generated. Because only one bit changes at a time, there is never any ambiguity due to slight timing differences.The first eight elements of the code are:

Project #13: Hãy giải bài tập 20.11 (sách: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010; trang 20-40): Mã Gray là một mã nhị phân cho số nguyên. Nó khác với cách biểu diễn số nhị phân thông thường ở chỗ, chỉ có một bit thay đổi giữa 2 biểu diễn của 2 số bất kỳ. Điều này là có ích cho các ứng dụng chẳng hạn như các bộ đếm hoặc các bộ chuyển đổi tương tự – số, trong đó có một dãy các con số được sinh ra. Bởi vì mỗi lần (sinh ra một con số) chỉ có một bit thay đổi, nên không bao giờ có chuyện nhập nhằng (nhầm) do sự sai khác về thời gian là ít. Tám thành phần đầu tiên của mã là: (trên bảng)

Binary Code	Gray Code
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

14. Project #14: To solve problem 20.12 (Book: …)

(Nguyễn Minh Khôi; Dương Hải Minh; Nguyễn Quý Thịnh)

(After finishing: save your logisim simulation circuit as: Project#14.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010; page 20-41): Design a 5 * 32 decoder using four 3 * 8 decoders (with enable inputs) and one 2 * 4 decoder.

(Hãy giải bài tập 20.12 (Sách: ...; Trang 20-41): Hãy thiết kế một bộ giải mã 5x32 bằng cách sử dụng 4 bộ giải mã 3x8 (có các lối vào enable) và một bộ giải mã 2x4)

15. Project #15: To solve problem 20.13 (Book: William Stallings)

(Nguyễn Tuấn Linh; Phan Tất Phúc; Nguyễn Ngọc Anh Vũ)

(After finishing: save your logisim simulation circuit as: Project#15.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(Book: Computer Organization and Architecture: Designing for Performance, Eight Edition, William Stallings, 2010; page 20-41): Implement the full adder of Figure 20.20 with just five gates. (Hint: Some of the gates are XOR gates.).

(Hãy giải bài tập 20.13 (Sách: ...; Trang 20-41): Hãy xây dựng một bộ full adder như trên hình 20.20, nhưng chỉ được sử dụng 5 cổng. (Gợi ý: Một số trong các cổng là cổng XOR))

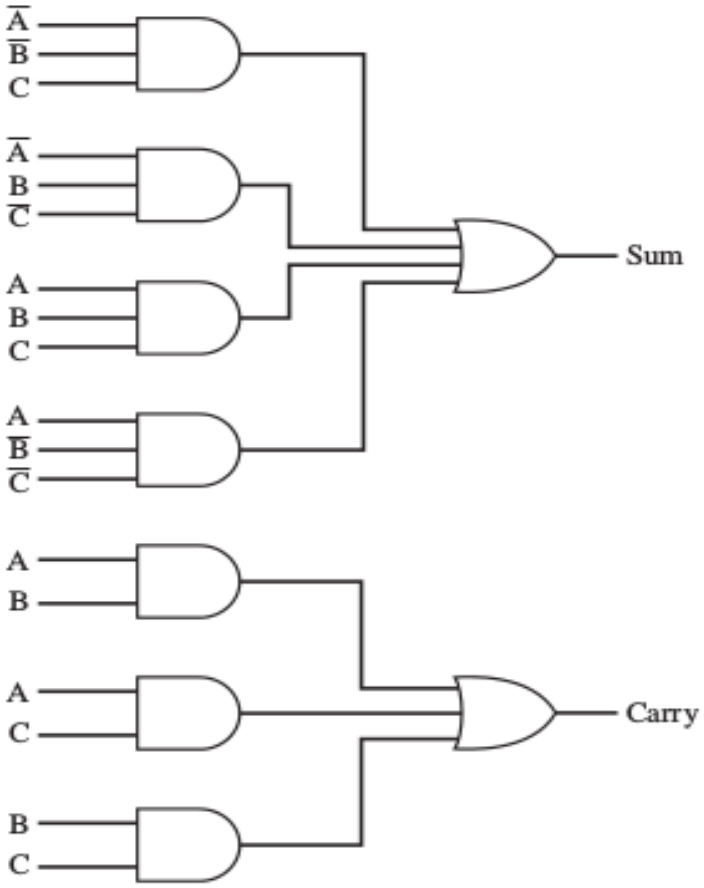


Figure 20.20 Implementation of an Adder

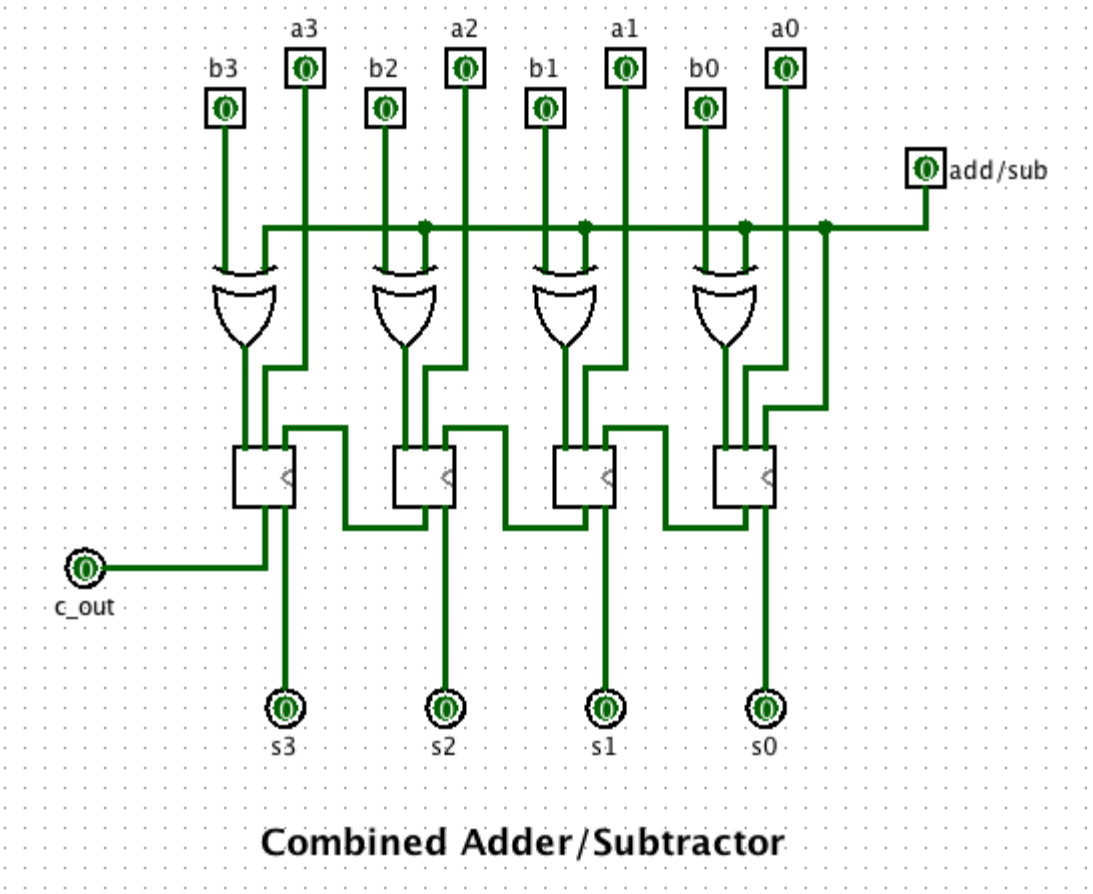
16. Project #16: To make a 4-bit ALU with functions: add, sub, dec and inc

(Trần Bá Hoà; Ngô Minh Hoàng; Phạm Khánh Ly)

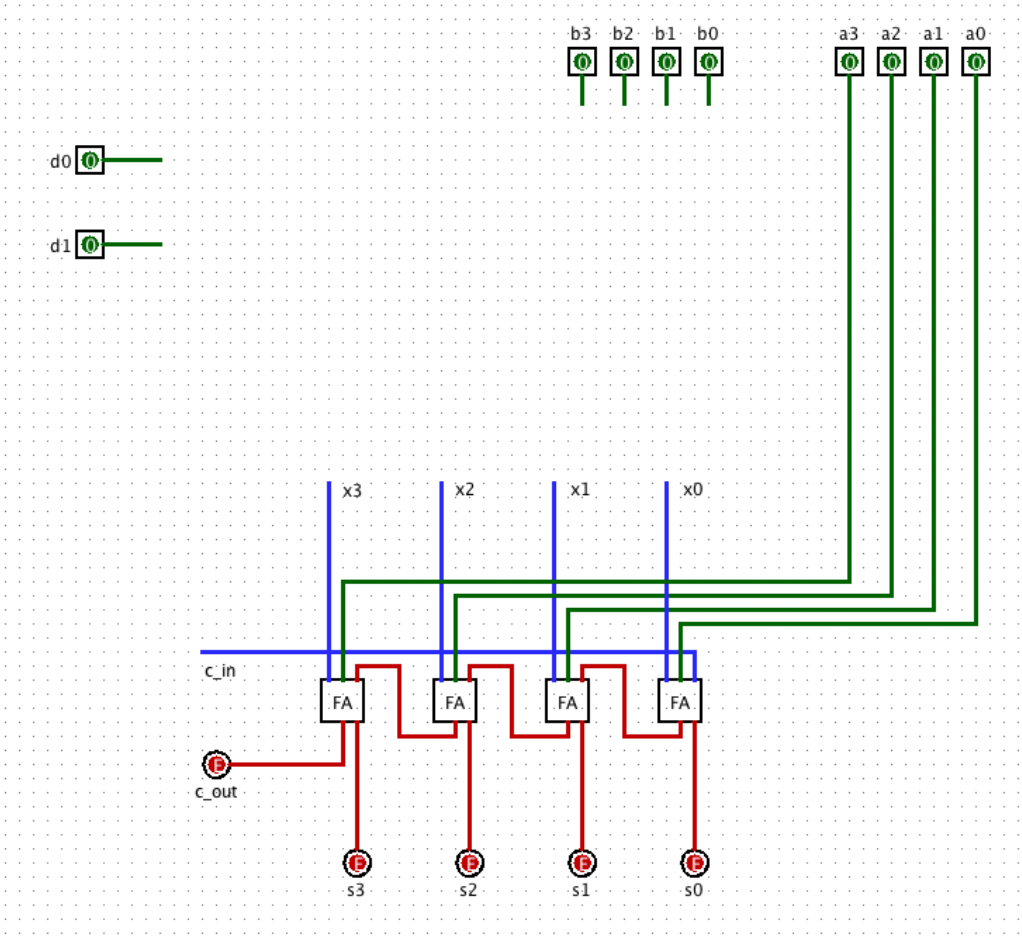
(After finishing: save your logisim simulation circuit as: Project#16.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

(After finishing: save your logisim simulation circuit as: Project#16.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

For this problem, you will build a 4-bit Arithmetic Logic Unit (ALU) that adds, subtracts, decrements and increments in Logisim. The design is based on the Combined Adder/Subtractor shown in class:



We can think of this circuit as having 1 control line that determines whether the circuit adds or subtracts. Our 4-bit ALU will have two control lines (d1 d0) that determines whether the ALU will add, subtract, decrement or increment:



This functionality can be built using just the 4-bit ripple-carry adder. Your problem is to figure out how to connect x3 x2 x1 x0 to d1 d0 and b3 b2 b1 b0. Follow these steps:

First, we write down what we want to accomplish:

- d1 d0 = 00 (ADD)

We want to add b3 b2 b1 b0 to a3 a2 a1 a0 and provide the sum in s3 s2 s1 s0. This can be accomplished by passing each b_i unchanged thru to x_i.

- d1 d0 = 01 (SUB)

We want to subtract b3 b2 b1 b0 from a3 a2 a1 a0 and provide the difference in s3 s2 s1 s0. We can use the same trick that the Combined Adder/Subtractor used: make x_i the complement of b_i and set the carry-in (c_in) to 1. That effectively adds the two's complement of b3 b2 b1 b0 to a3 a2 a1 a0.

- d1 d0 = 10 (DEC)

We want to subtract 1 from a3 a2 a1 a0 and provide the result in s3 s2 s1 s0. In this case, the values in b3 b2 b1 b0 are ignored. We can subtract 1 by adding 1 1 1 1 to a3 a2 a1 a0, since 1111 is −1 in 4-bit two's complement. Thus, each x_i should simply be set to 1 in this case. The carry-in should be 0.

- d1 d0 = 11 (INC)

We want to add 1 to a3 a2 a1 a0 and provide the result in s3 s2 s1 s0. As in the case for DEC, we ignore b3 b2 b1 b0. To add 1, we simply set each x_i to 0 and make c_in 1.

From the description above, we can produce a truth table for the values of x_i and c_in, given the inputs d1, d0 and b_i:

d1	d0	bi	xi	c_in
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

What to do:

- On a piece of paper to be turned in with Problems 1 and 2, write down a Boolean formula for x_i in terms of d1 d0 and b_i. Simplify the formula as much as you can. Clearly indicate your final formula for x_i.
- Write down a formula for c_in. Simplify the formula as much as you can. Clearly indicate your final formula for c_in.
- Download the starting point for your circuit: hw3-3.circ.
- In Logisim, connect the ripple-carry adder's inputs, x3 x2 x1 x0, to d1 d0 and b3 b2 b1 b0 according to your formula from the previous step. (Yes, you will have four copies of the same circuit. You can use copy-paste to do this more quickly.)
- In Logisim, connect c_in to d1 d0 and b0 according to your formula for c_in from above.
- Test your ALU and make sure that it works as advertised.
- Save your circuit diagram, transfer the file to GL and submit from GL as usual:
- `submit cs313 hw3 hw3-3.circ`
- Submit the work done on paper in class along with your work for Problems #1 & #2.

Implementation notes:

- When you simplify your Boolean formula, you want to reduce the number of gates. This might not correspond to a "simpler" formula as defined in high school and middle school Algebra.
- Give some thought to the layout of your circuit. Neatness will count in grading.
- Logisim gates come in three sizes: narrow, medium and wide. Pick the appropriate size.
- Logsim gates can have negated inputs (bubbles). This can greatly simplify your circuit's layout because you do not need to make room for NOT gates. To negate a particular input, select the gate using the arrow tool. Then, in the panel on the lower left, choose "Yes" to negate a particular input line.

17. Project #17: To make a function solver for the function f(x,y) = 6*y-8+x in 4bit arithmetic

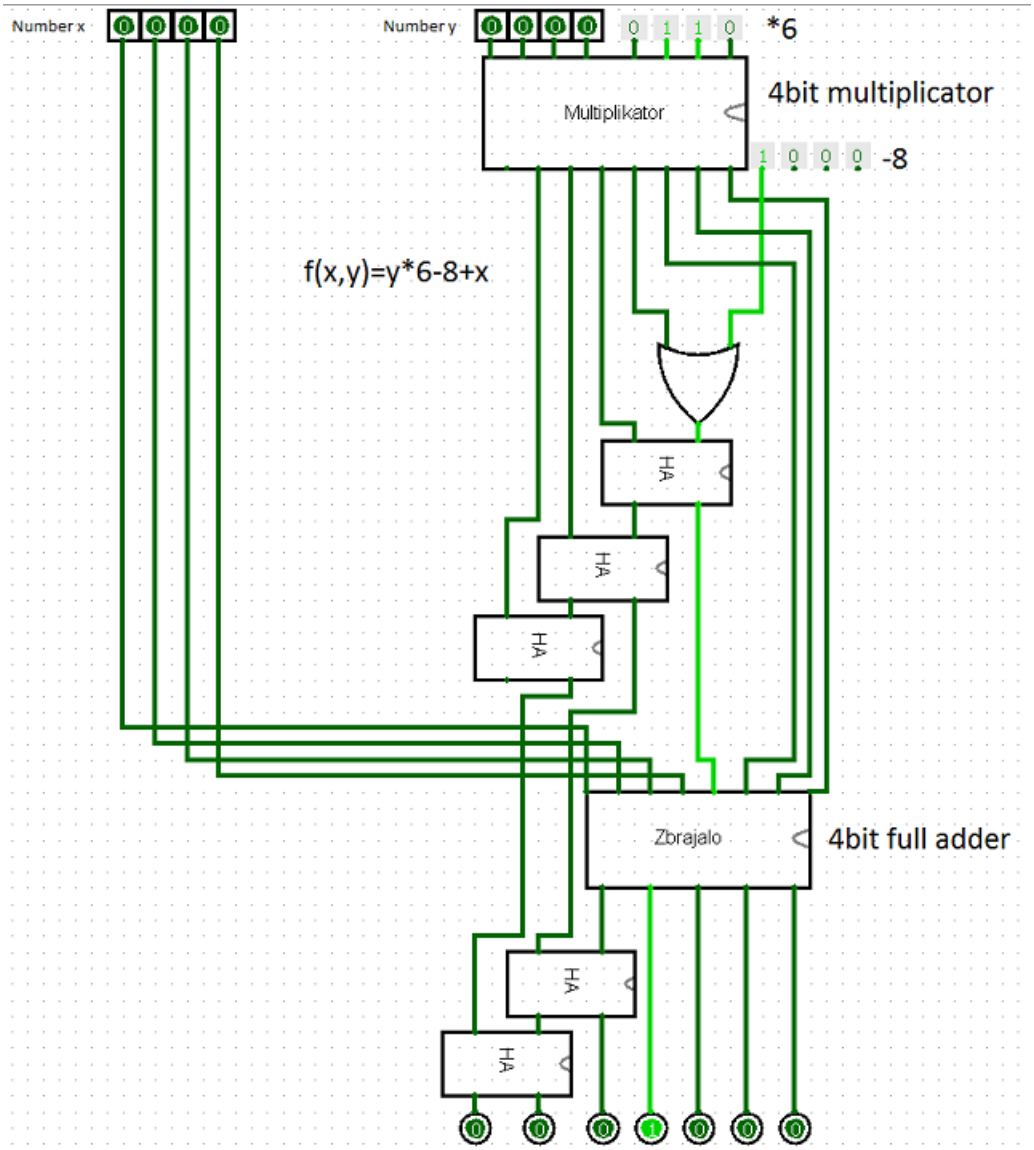
(Đỗ Thị Hồng Ngát; Lâm Hà Thái; Phạm Minh Tuấn)

(After finishing: save your logisim simulation circuit as: Project#17.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

A student (somewhere on the world) asked for help from UET Project#17-student group:

*I'm doing my homework for digital logic in logisim. I need to make function solver for this function: f(x,y) = 6*y-8+x in 4bit arithmetic. Here is my solution but it is not working for some inputs. I don't know where I failed. I will be thankful if you can help me somehow :) Thanks.*

Edit: it is working, for example for 0000 for X, and 0101 for Y, but it is not working for 0000 and 1101. I need to get solution for function above.



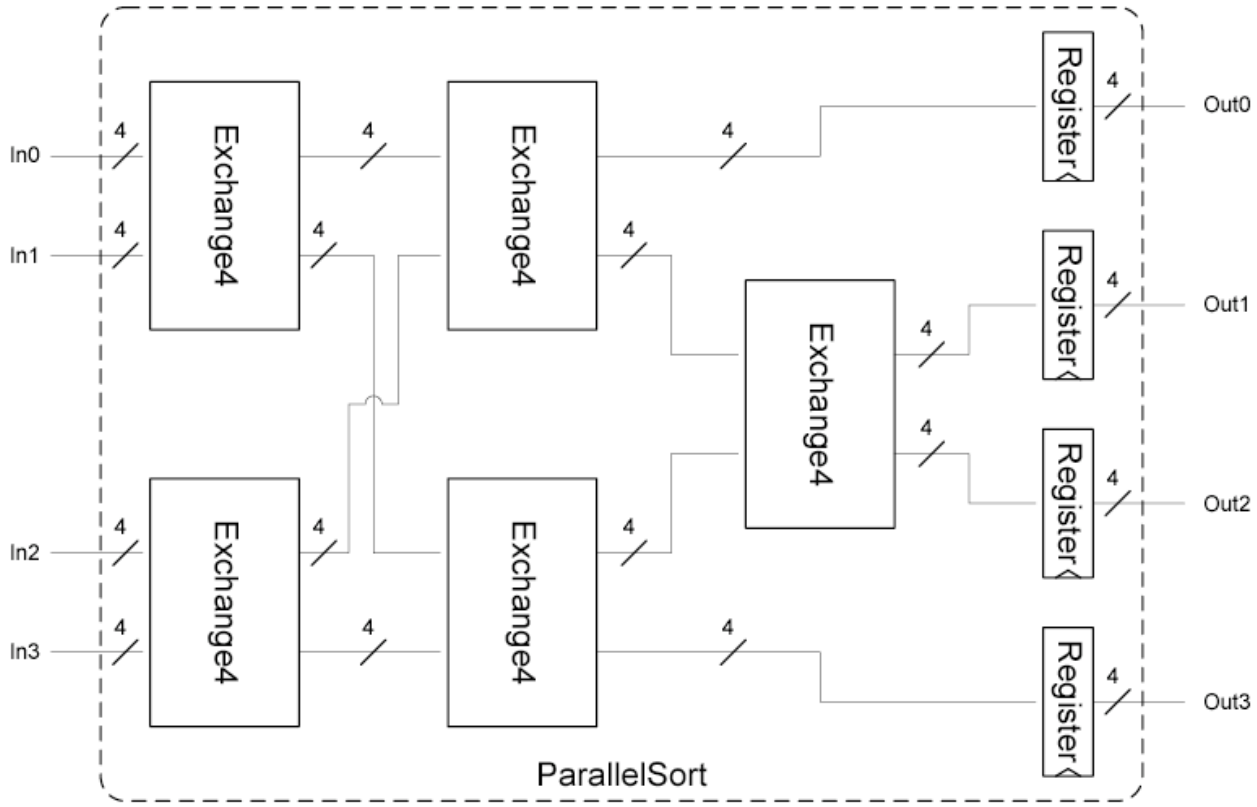
18. Project #18: ParaellelSort

(Nguyễn Trung Hiếu; Đặng Hữu Hoàn; Lê Khánh Toàn)

(After finishing: save your logisim simulation circuit as: Project#18.circ and then submit to your teacher via his email address: viet.nguyendinh@gmail.com)

You are to implement a 4-bit 4-input circuit that sorts its inputs in a single clock cycle (hence the name "parallel" or "combinational" sort).

Shown below is a schematic of the ParallelSort module. Your job is to translate this schematic into Logisim. We have already done some of the work for you; you may create new wires as needed, but you may not add inputs or outputs.



The schematic uses two other circuits, Exchange4 and Register. Registers are provided in one of the Logisim libraries; choose Project->Load Library->Memory to access them. You are to design Exchange4 yourself. It takes two 4-bit inputs A and B. Its two outputs are max(A,B) and min(A,B). We suggest that you use your subtractor from exercise 7.2 to implement Exchange4.

If you have a few extra minutes:

- Attempt to simplify the circuit by reducing the number of Exchange4 instances. Is it possible? Why or why not?
- Can you come up with a design for a sequential sorter which uses fewer Exchange4 instances by sharing them over time?