Digital Electronics and Microprocessor Systems (ELEC211)

Dave McIntosh and Valerio Selis

dmc@liv.ac.uk

v.selis@liv.ac.uk

Digital 6: Review, look ahead and digital problem-solving

UNIVERSITY OF LIVERPOOL

Outline

- Recap of Shannon's Expansion
- Notes about flip-flops
- More on Field Programmable Gate Arrays (FPGAs)
- Counters, Adders and 7-segment decoders

Previous material

PLA, PAL ✓

FPGAs ✓

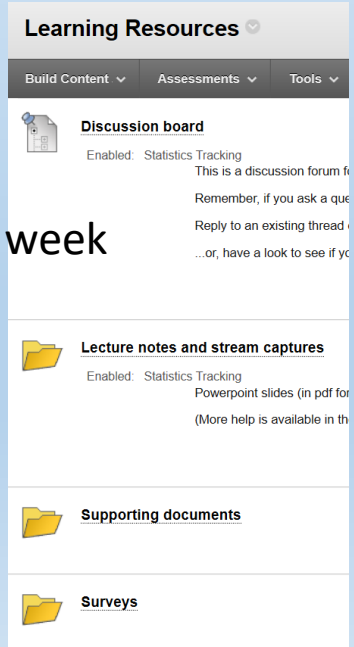
Flip-flops ✓

Karnaugh maps ✓

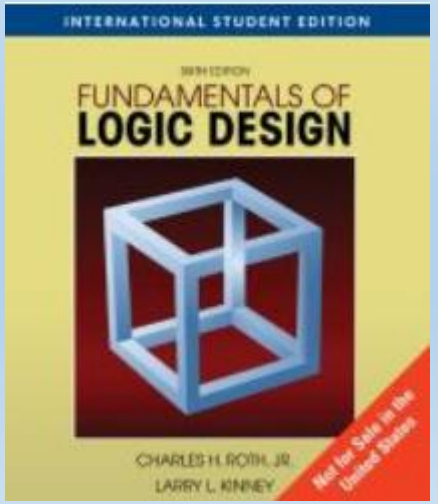
Use VITAL!:

- Stream lectures
- Handouts
- Notes and Q&A each week
- Discussion Board
- Exam resources

www.liv.ac.uk/vital



Course textbook – please borrow and use it!



UNIVERSITY OF LIVERPOOL

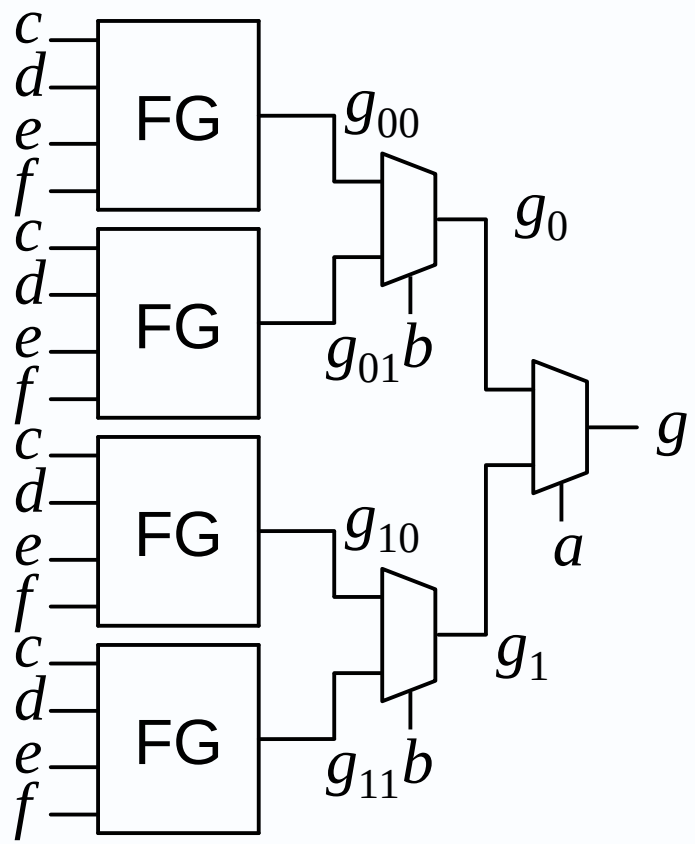
Claude E. Shannon

C. Shannon, 1916 -2001, American mathematician & electronic engineer, known as the ‘Father of Information Theory’.

Also, in 1937 he developed digital circuit design theory as an MSc student at MIT; described as "possibly the most important, & also the most famous, MSc of the century."

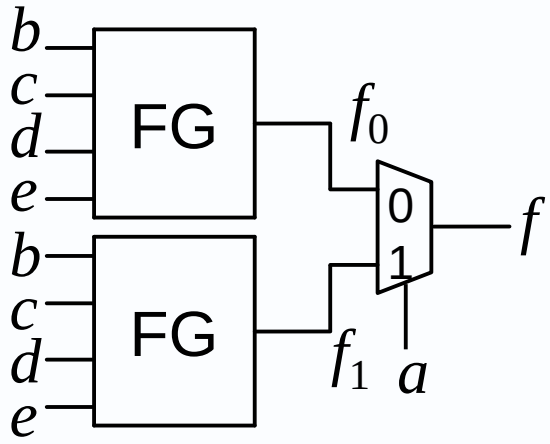
Realization of 6-variable functions with Function Generators

Any 6-variable function can be realized using four 4-variable function generators and three 2-to-1 mux.



$$g(a,b,c,d,e,f) = a'b'g(0,0,c,d,e,f) + a'bg(0,1,c,d,e,f)$$
$$+ ab'g(1,0,c,d,e,f) + abg(1,1,c,d,e,f)$$

Realization of 5-variable functions with Function Generators



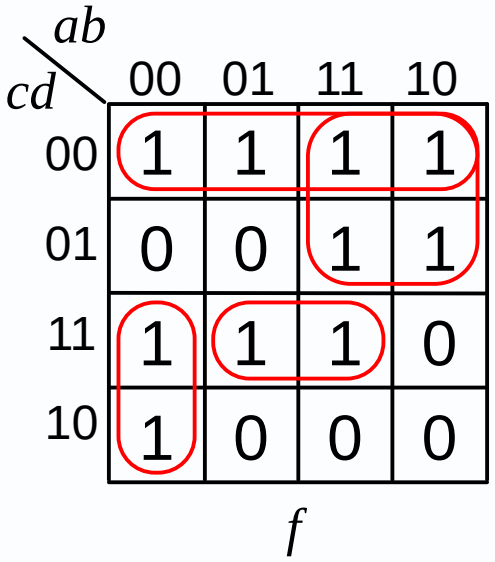
- Any 5-variable function can be realized using two 4-variable function generators and a 2-to-1 mux.

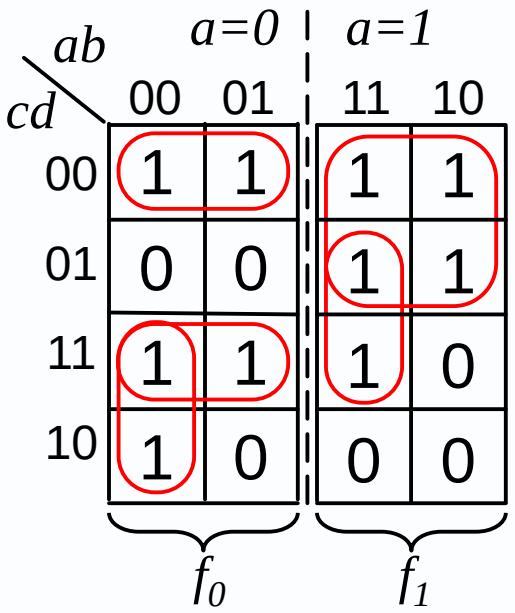
$$f(a,b,c,d,e) = a'f(0,b,c,d,e) + af(1,b,c,d,e)$$
$$= a'f_0 + af_1$$

Shannon's theorem

$$f(a,b,c,d) = c'd' + a'b'c + bcd + ac'$$
$$= a'f_0 + af_1 \ (our\ goal)$$

Here's how we do it:





$$f = a'(c'd' + b'c + cd) + a(c' + bd)$$

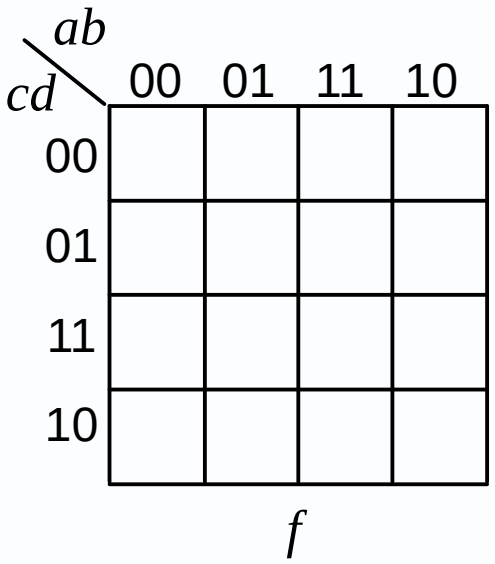
Example: Shannon's expansion (decomposition)

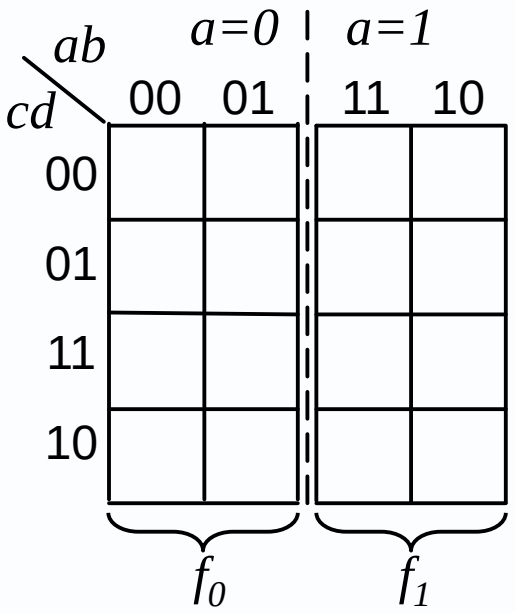
Decompose the following function into 2 functions, one for a' and the other for a.

$$f(a,b,c,d) = a'c'd' + abd + bcd + b'cd' + acd'$$
$$= a'f_0 + af_1$$

Method

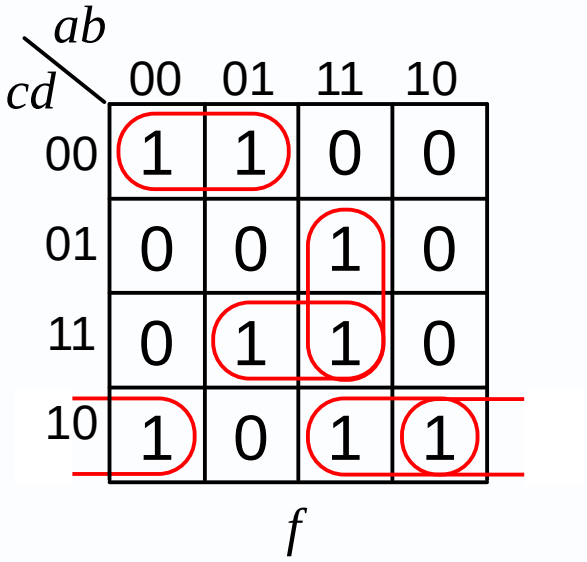
$$f(a,b,c,d) = a'c'd' + abd + bcd + b'cd' + acd'$$
$$= a'(\ldots) + a(\ldots)$$

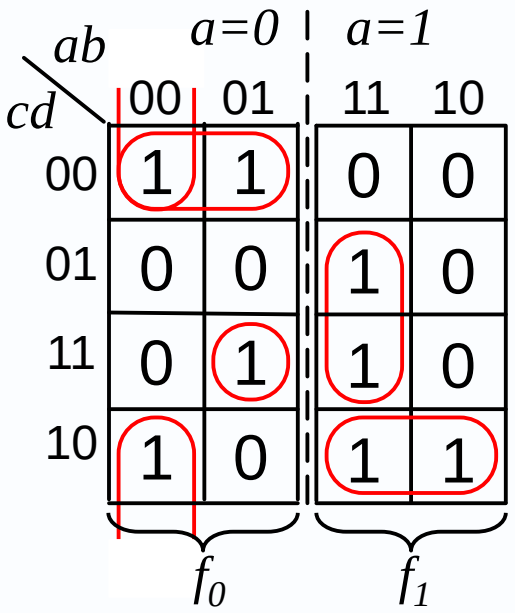




Answer

$$f(a,b,c,d) = a'c'd' + abd + bcd + b'cd' + acd'$$
$$= a'(c'd' + b'd' + bcd) + a(cd' + bd)$$





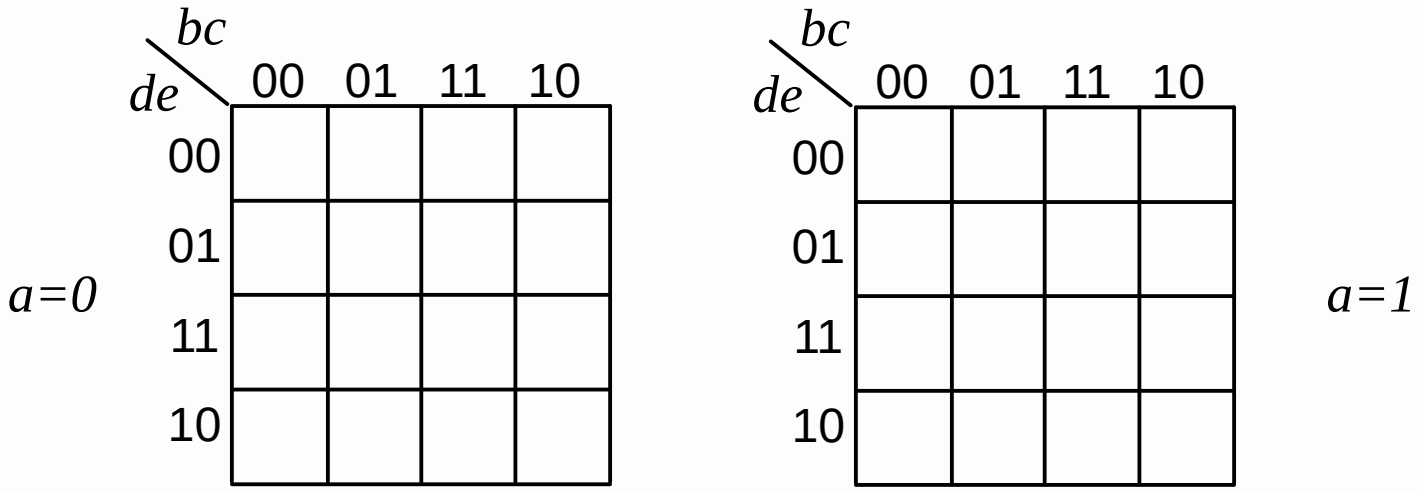
Example

Decompose the following function into 2 functions, one for a' and the other for a.

$$f(a,b,c,d,e) = b'c'd' + a'be + b'cde' + ab'd' + bcde$$
$$= a'f_0 + af_1$$

Method

$$f(a,b,c,d,e) = b'c'd' + a'be + b'cde' + ab'd' + bcde)$$
$$= a'(\ldots) + a(\ldots)$$

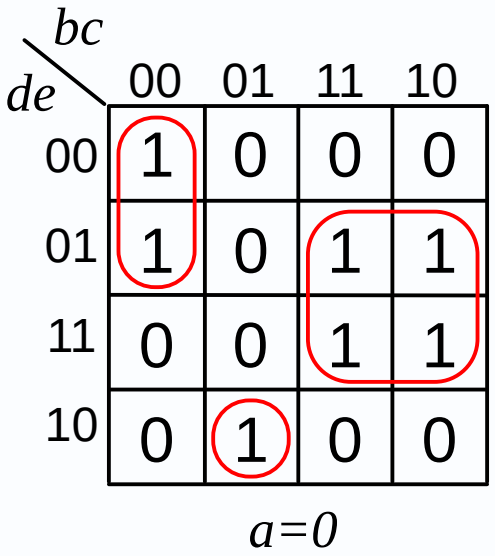


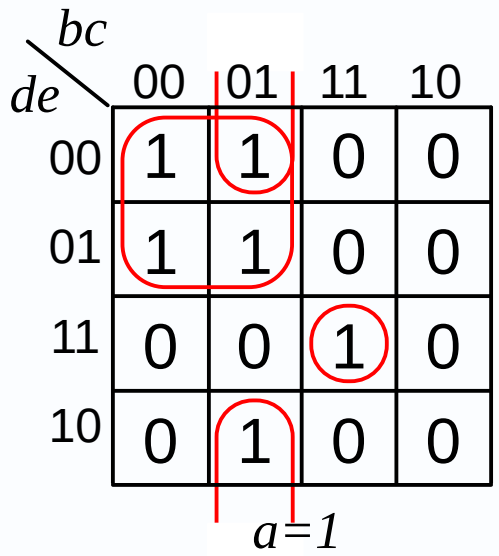
Answer

$$f = (a + a')b'c'd' + a'be' + (a + a')cde' + ab'd' + (a + a')bcde$$

$$f(a,b,c,d,e) = b'c'd' + a'be + b'cde' + ab'd' + bcde$$

$$= a'(b'c'd' + be + b'cde') + a(b'd' + b'ce' + bcde)$$



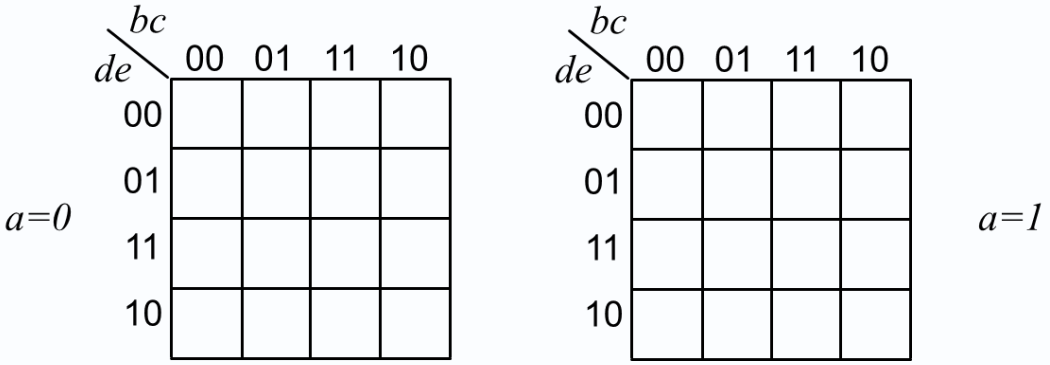


UNIVERSITY OF LIVERPOOL

Question

Decompose the following function into 2 functions, one for a' and the other for a.

$$f(a,b,c,d,e) = bc'd + abe + b'c'de + a'bd + bcd'e'$$
$$= a'f_0 + af_1$$



Complex Programmable Logic Devices (CPLD)

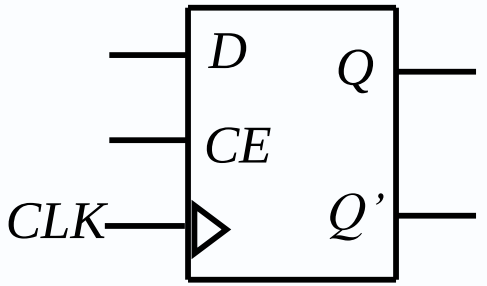
Xilinx XCR3064XL

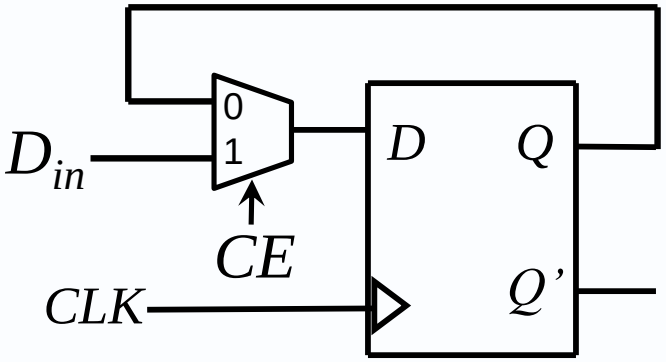
UNIVERSITY OF LIVERPOOL

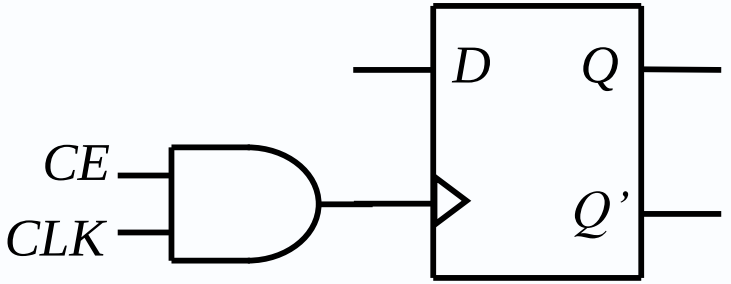
From OHO Elektronik GOP_XCR3064XL user manual v0.9 at:
http://www.trenz-electronic.de/fileadmin/docs/OHO-Elektronik/TE0251-GOP_XCR3064XL/documents/UM_XCR3064XL.pdf

Flip-Flops with Clock Enable

Flip-Flops with a clock enable are commonly used in CPLDs and FPGAs





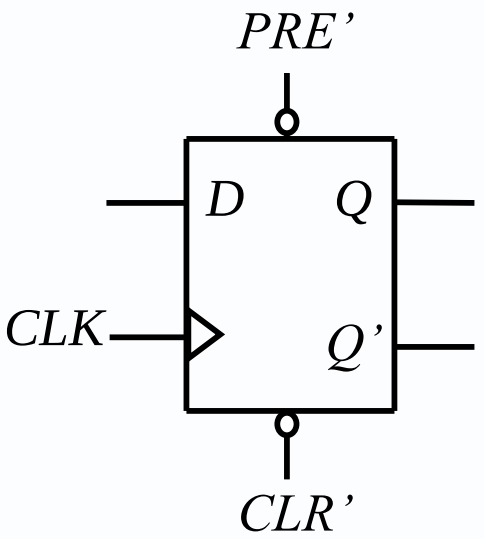


Two possible implementations.
ANDing the clock with clock enable (CE) results in loss of synchronisation but uses less electrical power.

Flip-Flops with Preset and Clear

Flip-Flops often have additional ‘asynchronous’ inputs which can be used to set the flip-flops to an initial state independent of the clock.

CLK	*D*	*PRE’*	*CLR’*	Q^+
X	X	0	0	Not allowed
X	X	0	1	1
X	X	1	0	0
↑	0	1	1	0
↑	1	1	1	1
0,1,↓	X	1	1	*Q*



Question

- Design a circuit for the following expression using a 4 to 1 mux with B and C connected to the select inputs.

$$f = B'A' + BA + CB'$$

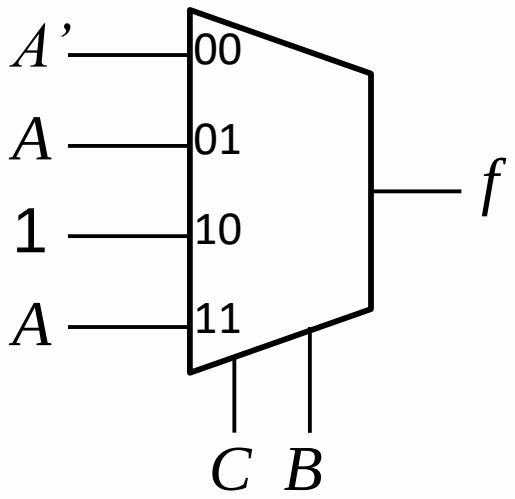
4-to-1 MUX has 4 data inputs

Given B and C as the 2 control / select inputs

(As there are 2^n data inputs, n = 2 makes sense)

Answer

C	B	A	f	
0	0	0	1	} A'
0	0	1	0	
0	1	0	0	} A
0	1	1	1	
1	0	0	1	} 1
1	0	1	1	
1	1	0	0	} A
1	1	1	1	



Question

- Design a circuit for the following expression using a 4 to 1 mux with C and B connected to the select inputs.

$$f = BA' + CBA + C'B$$

4-to-1 MUX has 4 data inputs

Given B and C as the 2 control / select inputs

(As there are 2^n data inputs, n = 2 makes sense)

Field Programmable Gate Arrays (FPGA)

This material will take you through a Altera® DE1 Development and Education board (Figure 1). Featuring an Altera Cyclone® II 2C20 FPGA, the DE1 board is designed for university and college laboratory use. It is suitable for a wide range of exercises in courses on digital logic and computer organization, from simple tasks that illustrate fundamental concepts to advanced designs.

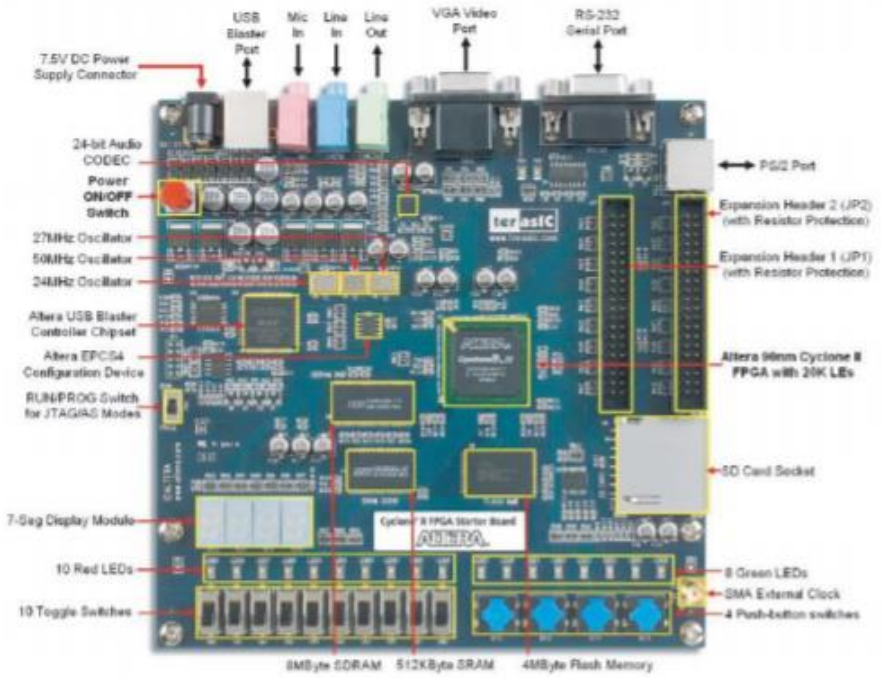


Figure 1 Altera DE1 Board

The following hardware is provided on the DE1 board:

- Altera Cyclone® II 2C20 FPGA device
- Altera Serial Configuration device – EPCS4
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 512-Kbyte SRAM
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory

Altera DE1 board

Cyclone II 2C20 FPGA

From supporting material for Experiment 28, 2018/19

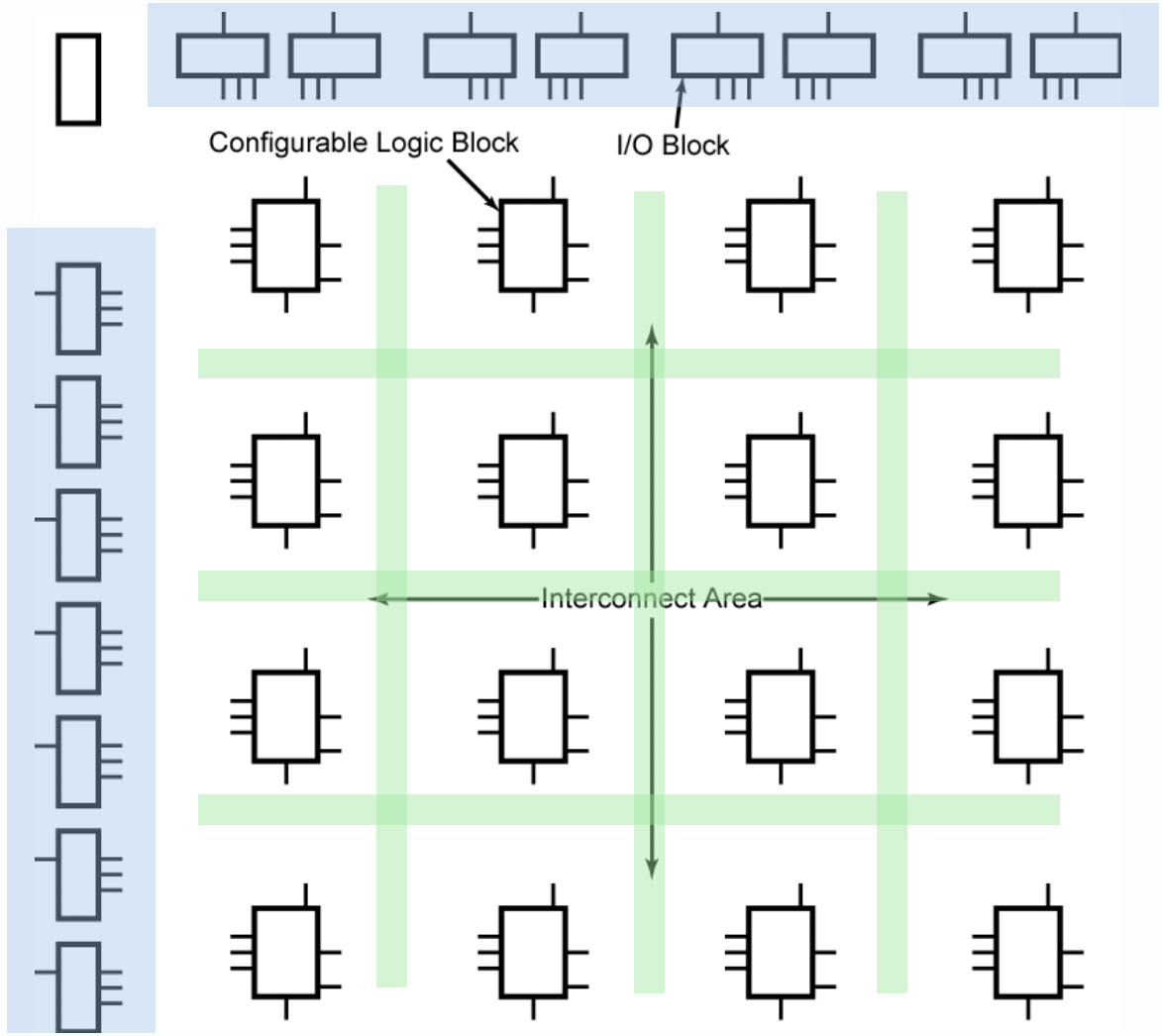
UNIVERSITY OF LIVERPOOL

Field Programmable Gate Arrays (FPGA)

An FPGA is an IC that contains an array of identical logic cells.

The I/O blocks connect the Configurable Logic Blocks (CLBs) to IC pins

The I/O blocks, the CLBs and the interconnection between CLBs are programmable



Simplified CLB (Configurable Logic Block)

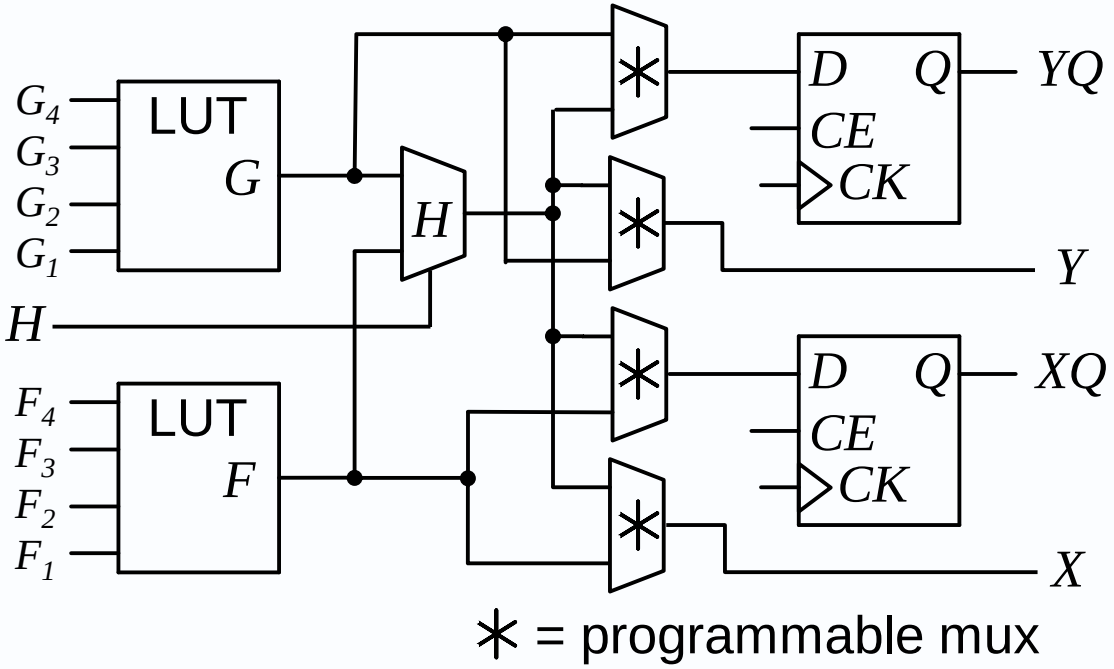
This (simplified) CLB contains 2 function generators, 2 flip-flops, and multiplexers for routing signals within the CLB

The function generators are implemented as lookup tables (LUT).

A 4-input LUT is essentially a reprogrammable ROM with 16 1-bit words.

The CLB has two combinational outputs (X,Y) & two flip-flop outputs (XQ, YQ).

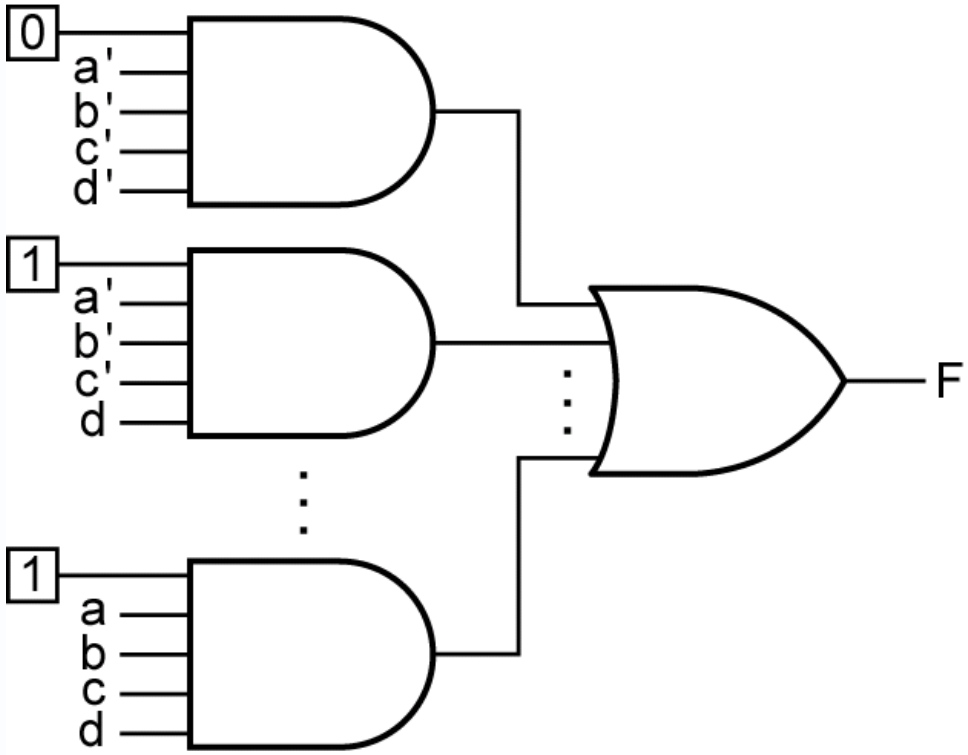
X and Y outputs & the flip-flop inputs are selected by the programmable Multiplexers (the control inputs are programmed when the FPGA is configured)



Implementing a (LUT) function generator

0 or 1 in the squares represents the bits stored in the LUT. These bits enable particular minterms.

A function with only one minterm or with as many as 15 minterms requires a single function generator



e.g.

$$F = abcd \quad \text{or}$$

$$F = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}b\bar{c}d + a\bar{b}\bar{c}d + \bar{a}bc\bar{d} + a\bar{b}c + abc + ab\bar{c} + \bar{a}\bar{b}cd$$

Question

- Design a circuit for the following expression using a BCD to decimal decoder and two OR gates.

$$f_1 = a'b'c' + a'c'd$$

$$f_2 = a'bd + ab'c'd'$$

Method

$$f_1 = a'b'c' + a'c'd$$

$$\overset{m_0}{} \qquad \overset{m_1}{} \qquad \overset{m_5}{}$$

$$= a'b'c'd' + a'b'c'd + a'b'c'd + a'bc'd$$

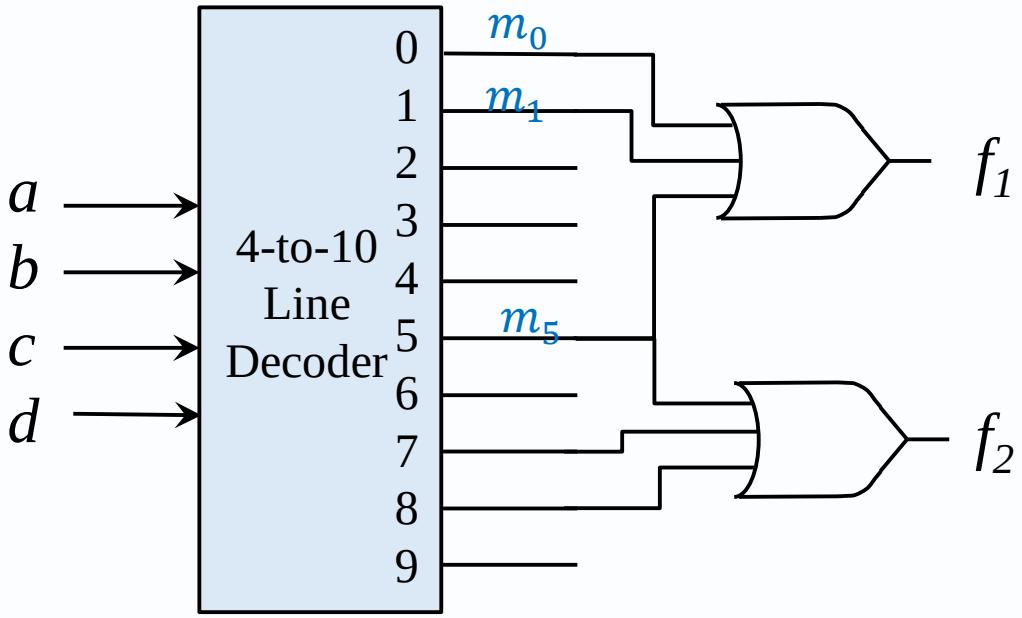
$$f_2 = a'bd + ab'c'd'$$

$$\overset{m_5}{} \qquad \overset{m_7}{} \qquad \overset{m_8}{}$$

$$= a'bc'd + a'bcd + ab'c'd'$$

a	b	c	d	f_1	f_2	
0	0	0	0	m_0 1	0	
0	0	0	1	m_1 1	0	
0	0	1	0	0	0	
0	0	1	1	0	0	
0	1	0	0	0	0	
0	1	0	1	m_5 1	1	m_5
0	1	1	0	0	0	
0	1	1	1	0	1	m_7
1	0	0	0	0	1	m_8
1	0	0	1	0	0	
1	0	1	0	0	0	
1	0	1	1	0	0	
1	1	0	0	0	0	
1	1	0	1	0	0	
1	1	1	0	0	0	
1	1	1	1	0	0	

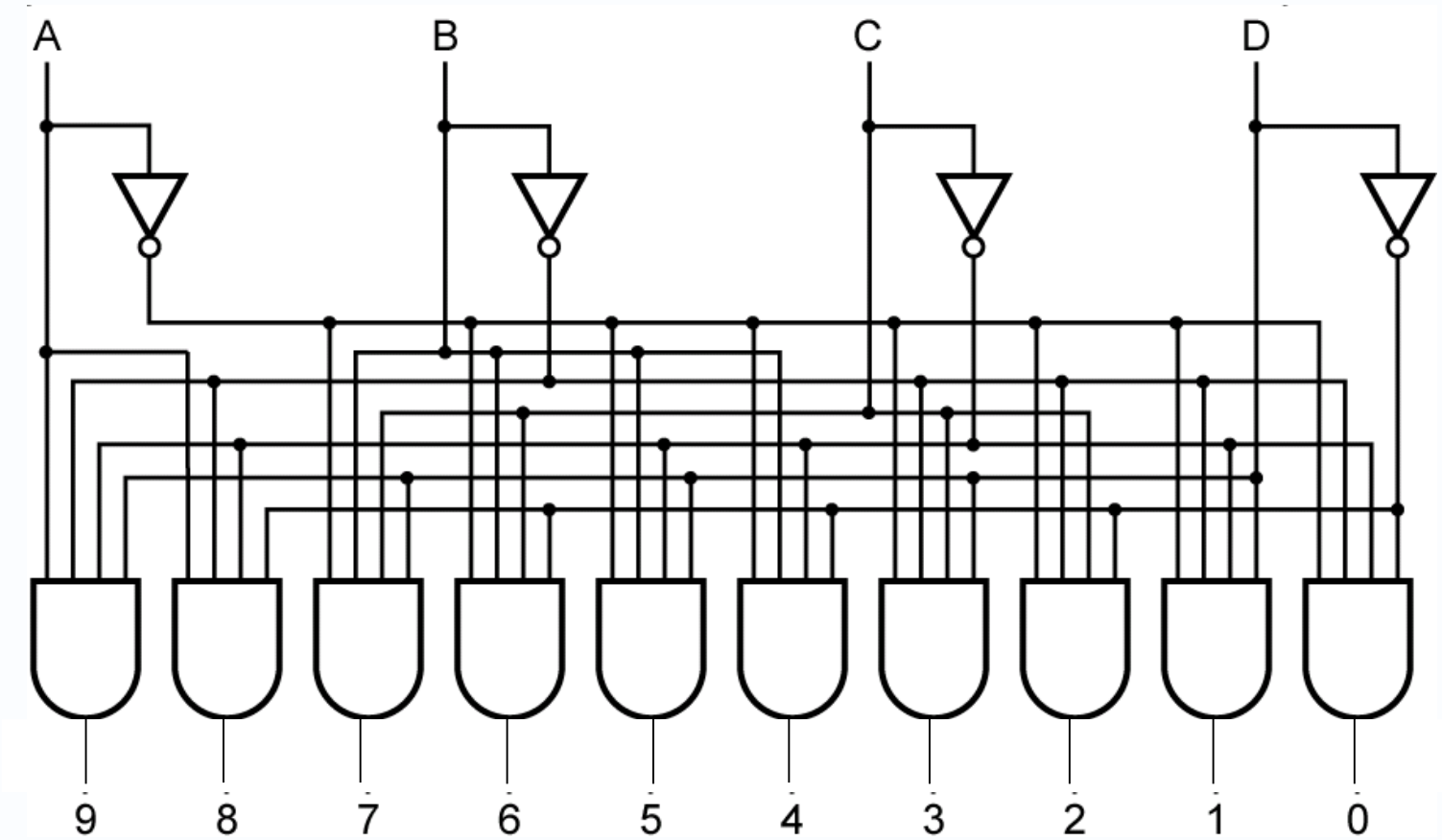
Answer



$$f_1 = m_0 + m_1 + m_5$$

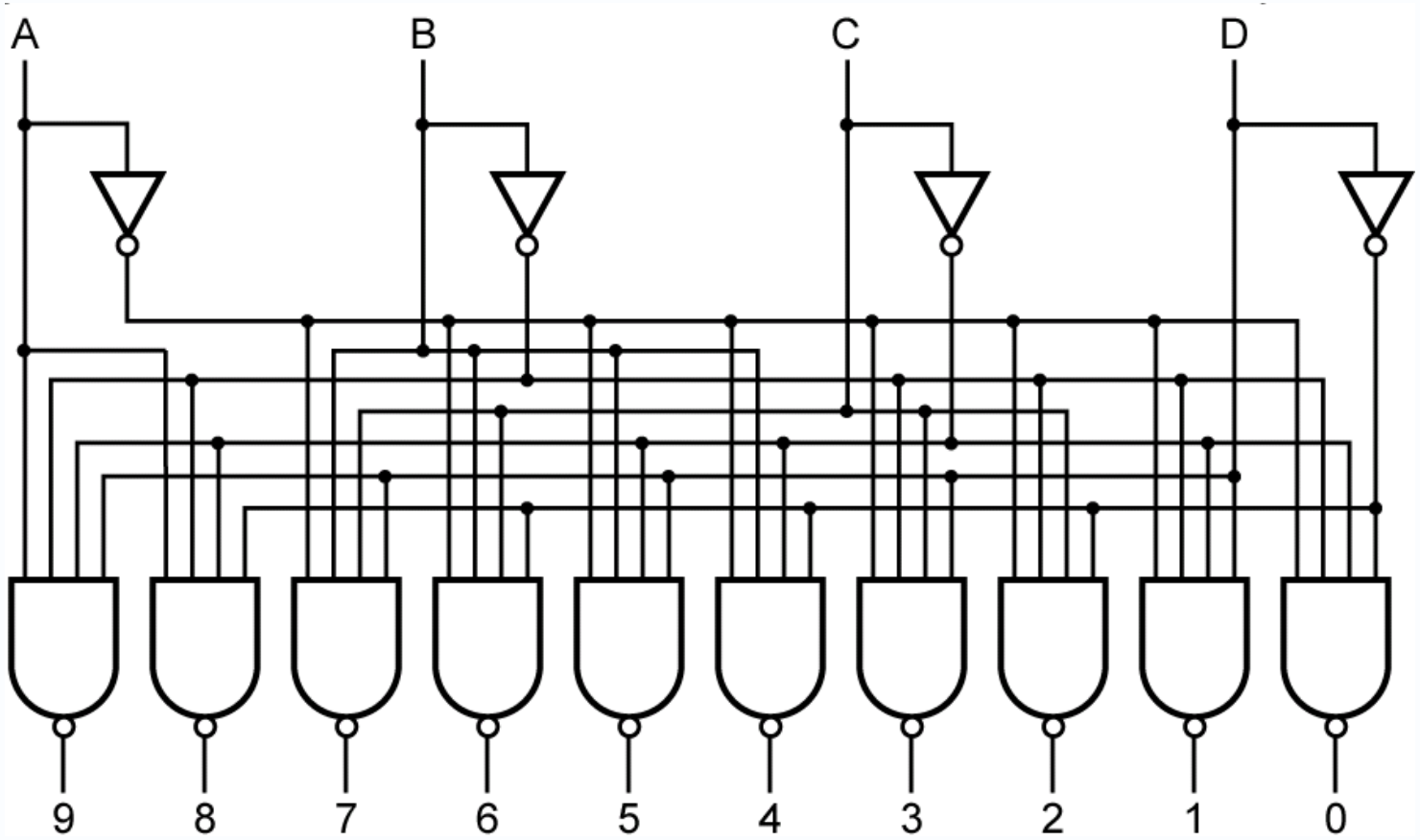
$$f_2 = m_5 + m_7 + m_8$$

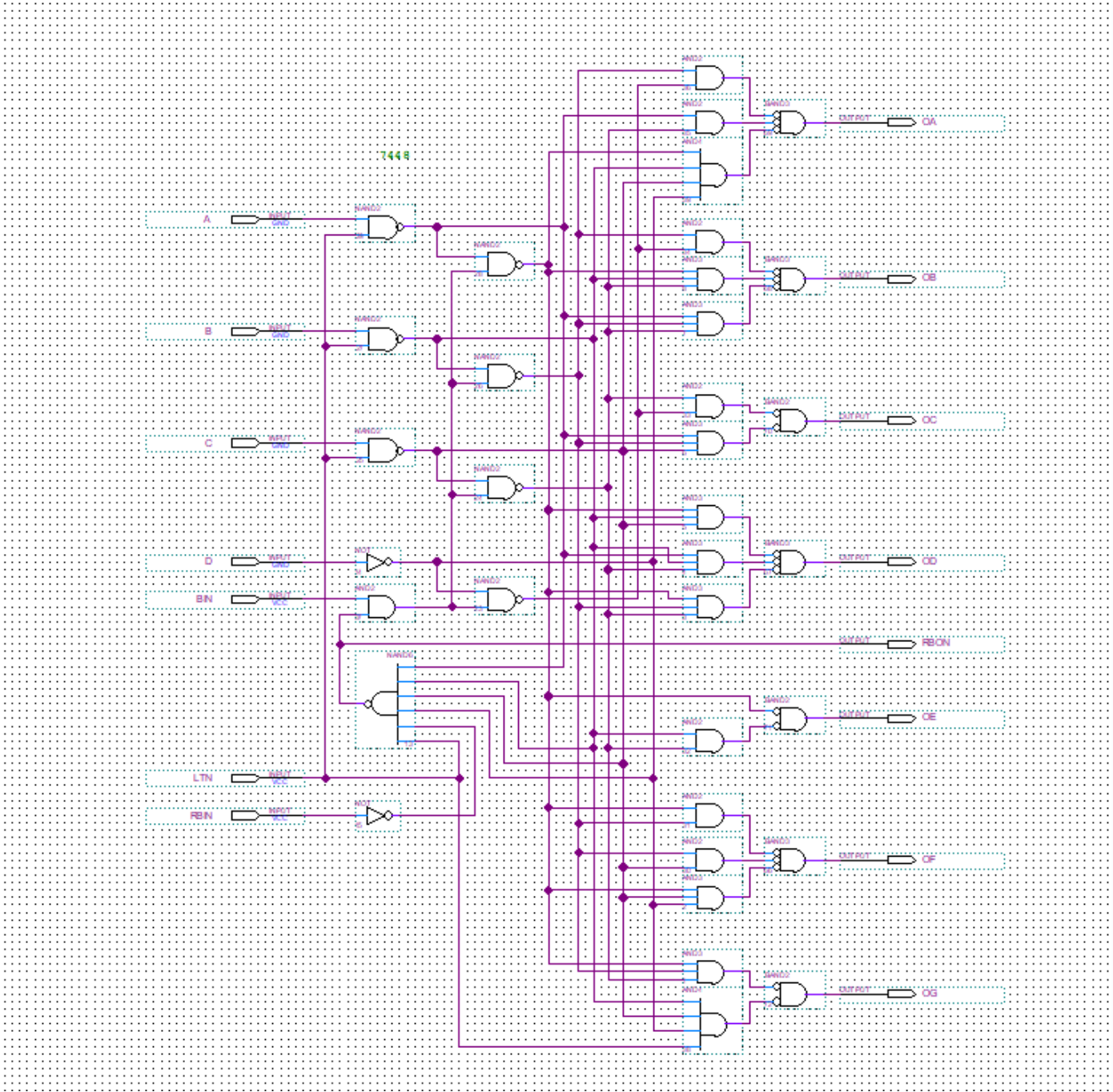
Circuit diagram for BCD to decimal decoder



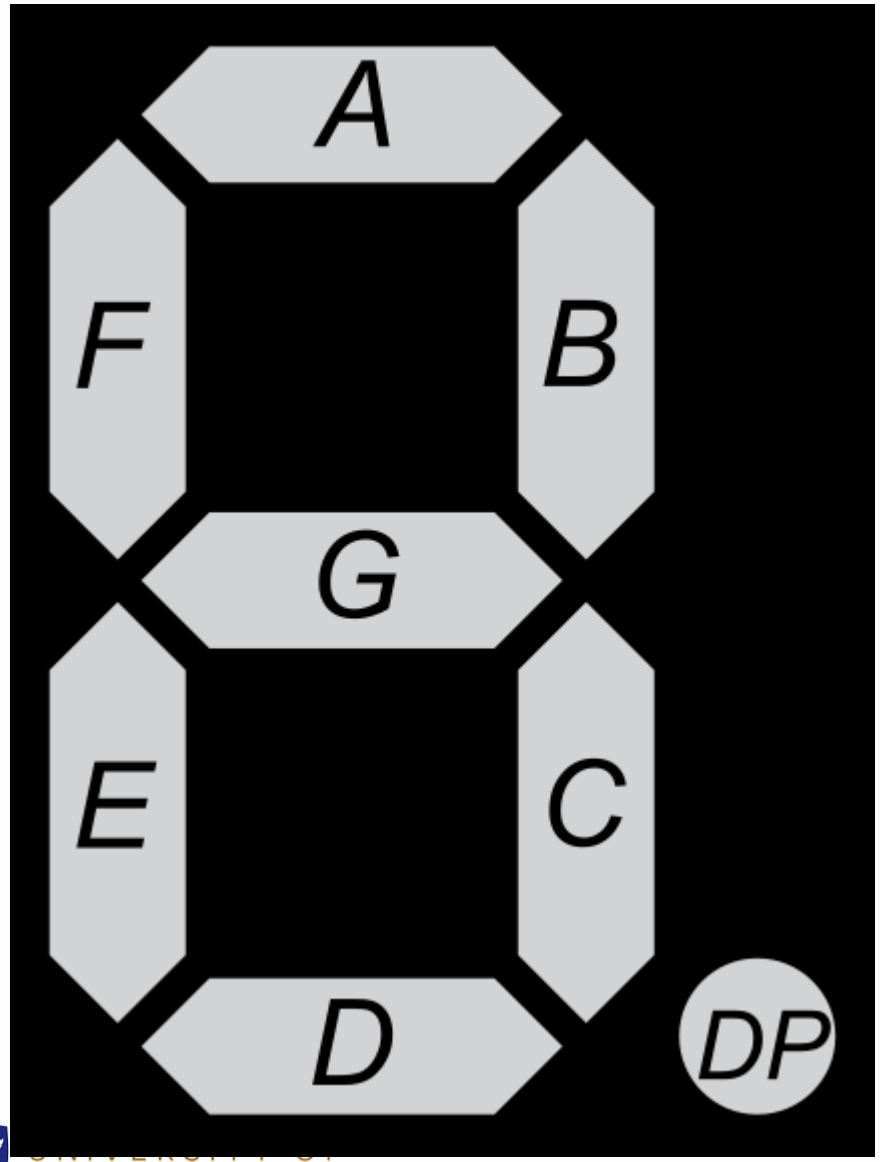
*Lots of AND gates here **...but...** NAND gates are cheaper to manufacture than AND gates (fewer transistors)*

Circuit diagram for BCD to decimal decoder *(inverted outputs)*





UNIVERSITY OF LIVERPOOL



BCD Inputs				Outputs						
a	b	c	d	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	1	1	0	0	1
0	0	1	1	0	1	1	0	0	1	1
0	1	0	0	1	0	1	1	0	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Synchronous Binary Counters using D-type Flip-Flops

Outputs of flip-flops determine the state of the counter.

Binary counters restart at zero after it reaches all one state.

A 3-bit counter can be designed as follows:

Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Binary Counter using D-type FF

Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	0	0	0	0

C^+

BA \ C	0	1
00	0	1
01	0	1
11	1	0
10	0	1

B^+

BA \ C	0	1
00	0	0
01	1	1
11	0	0
10	1	1

A^+

BA \ C	0	1
00	1	1
01	0	0
11	0	0
10	1	1

$$D_C = C^+ = \overline{C}.B.A + C.\overline{B} + C.\overline{A}$$
$$= C \oplus (A.B)$$
$$D_B = B^+ = \overline{B}.A + B.\overline{A} = A \oplus B$$
$$D_A = A^+ = \overline{A}$$

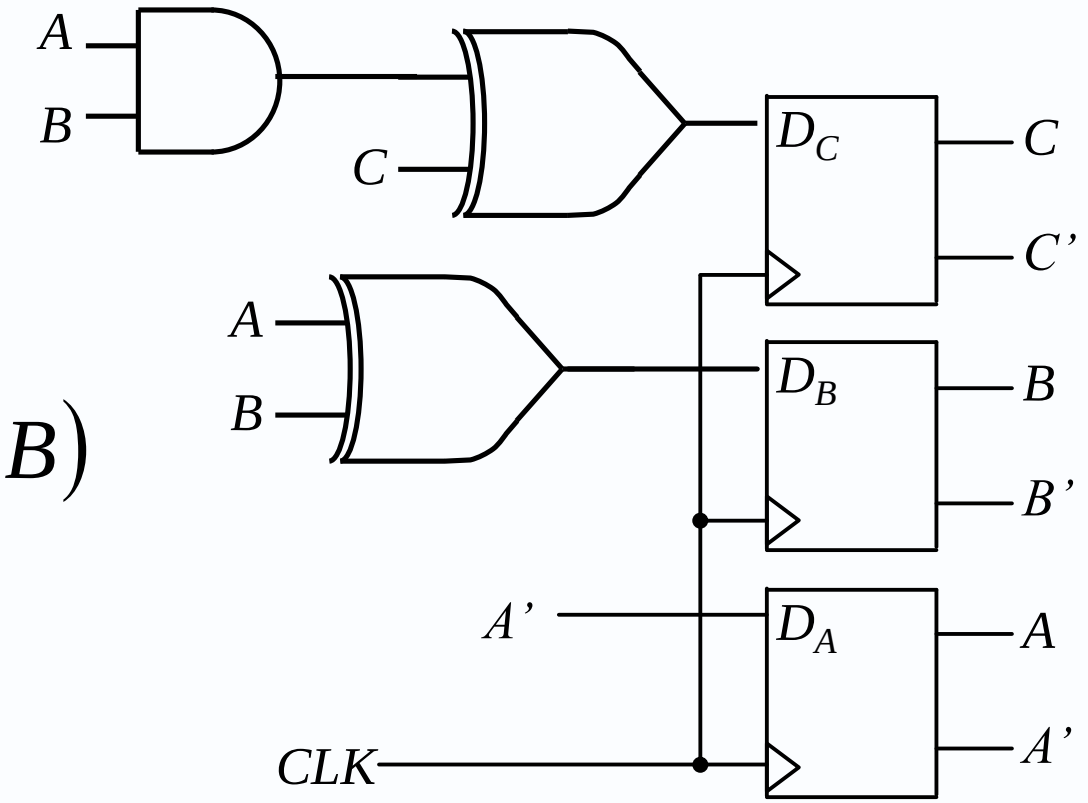
UNIVERSITY OF LIVERPOOL

Binary Counter using D-type FF

$$D_C = C \oplus (A.B)$$

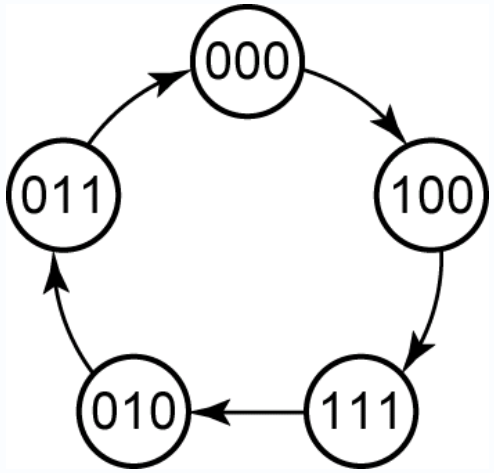
$$D_B = A \oplus B$$

$$D_A = \overline{A}$$



Counters for other sequences using D-type Flip-Flops

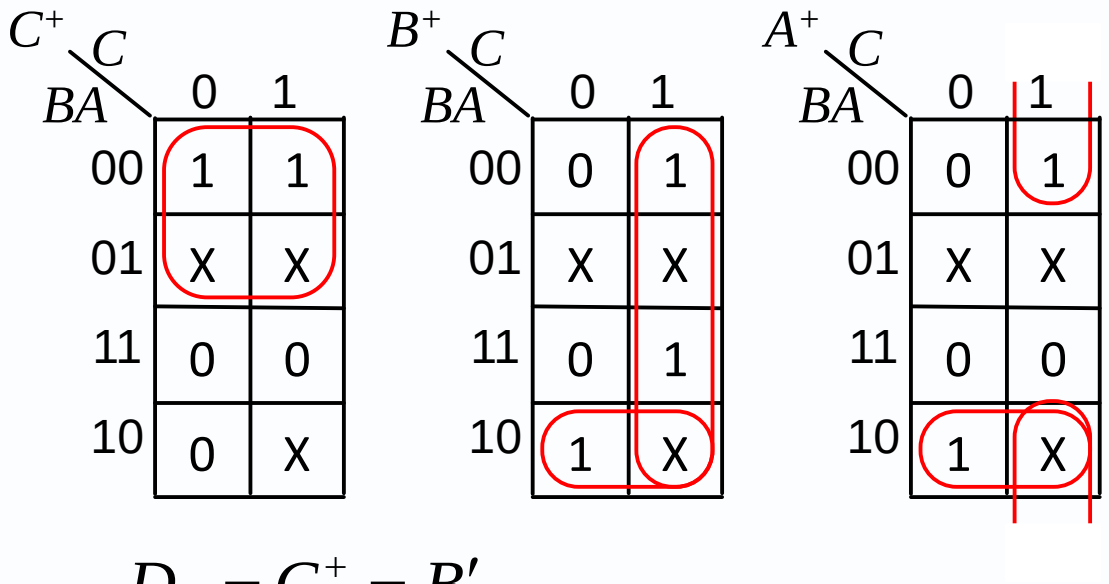
In some applications the sequence of states of a counter is not in strict binary order.



Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0

Counters for other sequences using D-type Flip-Flops

Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0



$$D_C = C^+ = B'$$

$$D_B = B^+ = C + BA'$$

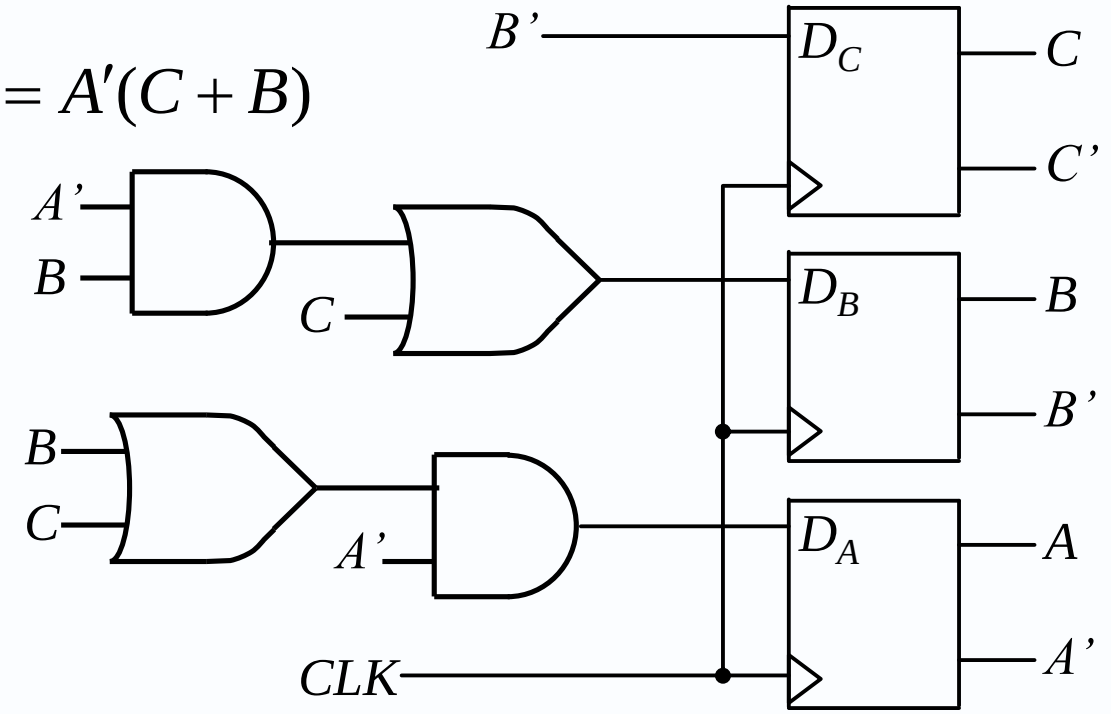
$$D_A = A^+ = CA' + BA' = A'(C + B)$$

Counters for other sequences using D-type Flip-Flops

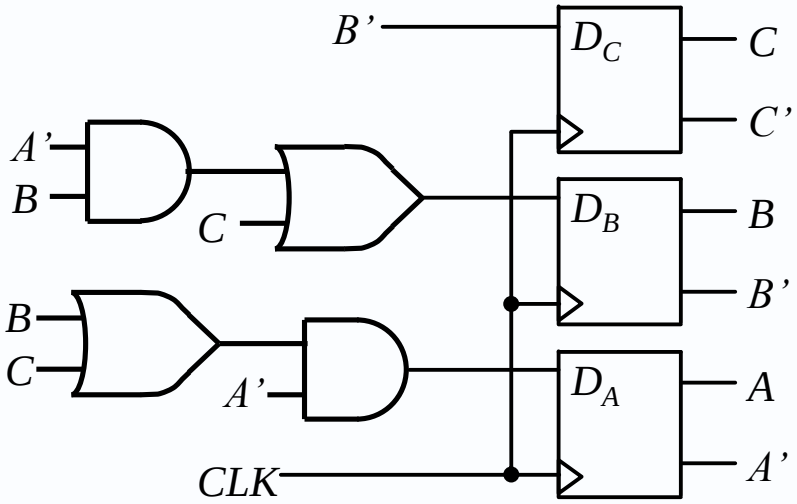
$$D_C = B'$$

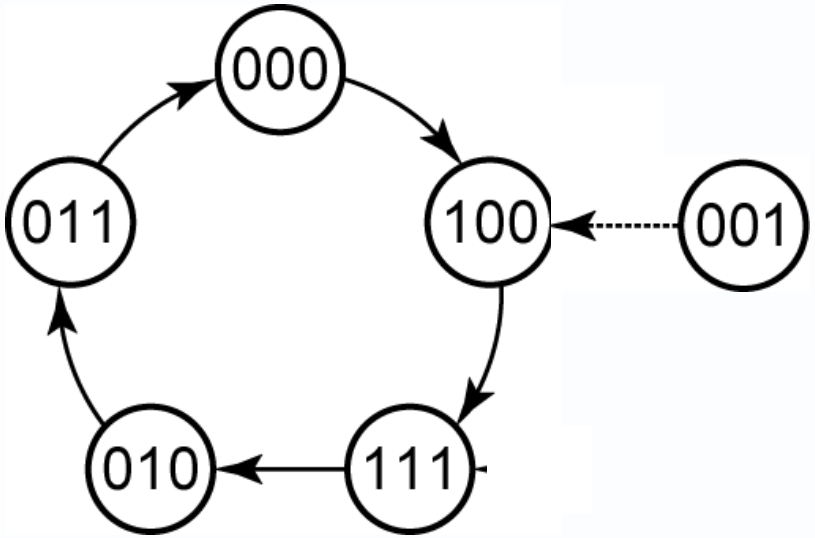
$$D_B = C + BA'$$

$$D_A = CA' + BA' = A'(C + B)$$



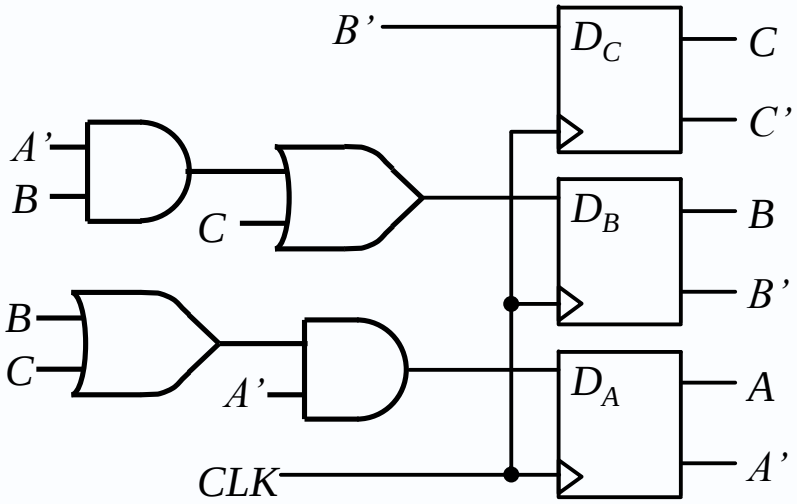
Counters: unused states

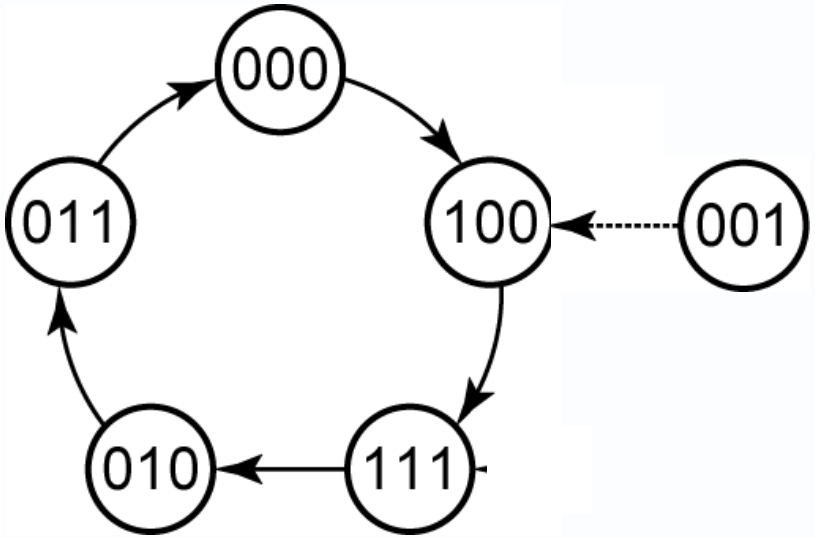




After power-on, the initial state of the flip-flops is unpredictable. If the initial state is CBA =101 then according to the circuit the next state value will be 110 which is not one of the valid states. Fortunately after the next clock pulse the output will become 011 which is a valid state.

Counters: unused states

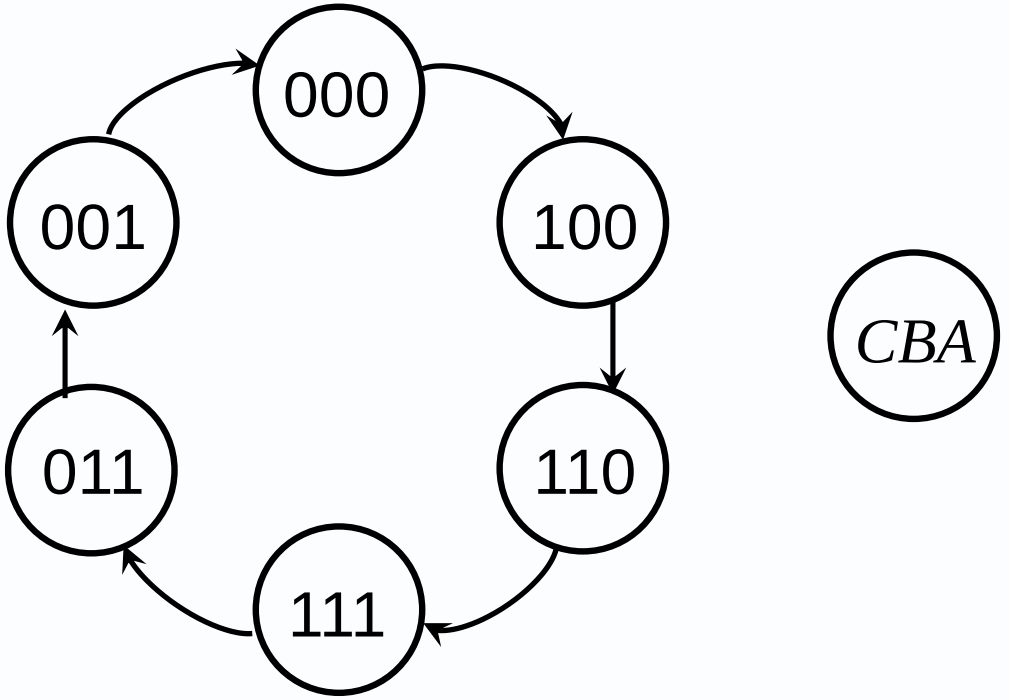




All don't care states should be checked to make sure that they eventually lead into the main counting sequence.

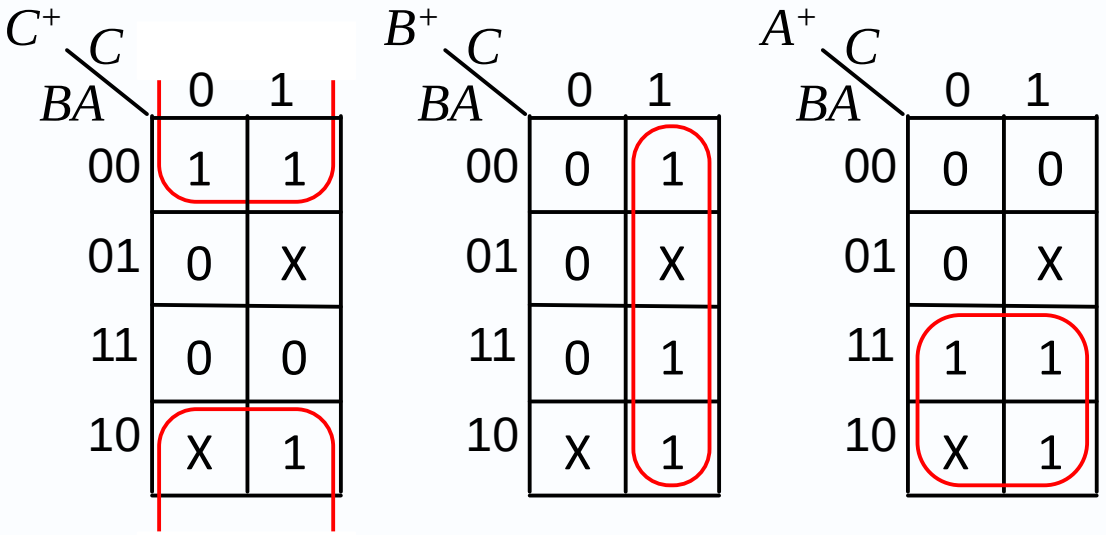
Question

Design a synchronous counter using D-type flip-flops for the following sequence (grey code).



Answer

Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	--	--	--
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	--	--	--
1	1	0	1	1	1
1	1	1	0	1	1



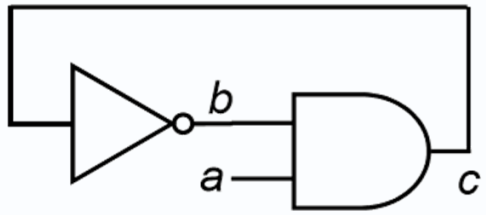
$$D_C = C^+ = A'$$

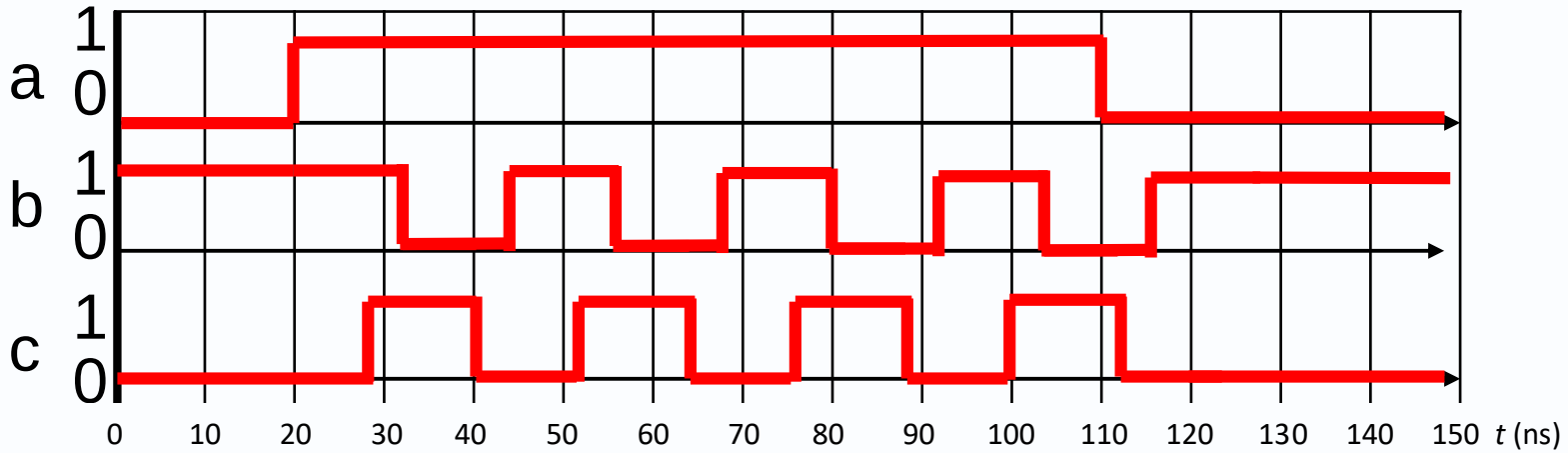
$$D_B = B^+ = C$$

$$D_A = A^+ = B$$

Remember this?

The inverter in the figure has a propagation delay of 4 ns and the AND gate of 8 ns. Draw a timing diagram for the circuit showing a, b, c. a and c are initially equal to 0, b is initially one. After 20 ns a becomes 1 for 90 ns and then 0 again.





UNIVERSITY OF LIVERPOOL

Full 1 bit Adder

X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = Carry input
C_{out} = Carry output

$$C_{out} = X \cdot C_{in} + Y \cdot C_{in} + X \cdot Y$$

$$Sum = X' \cdot Y' \cdot C_{in} + X' \cdot Y \cdot C'_{in} + X \cdot Y' \cdot C'_{in} + X \cdot Y \cdot C_{in}$$

Full 1 bit Adder

X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = Carry input
C_{out} = Carry output

$$C_{out} = X \cdot C_{in} + Y \cdot C_{in} + X \cdot Y$$

Note: $X \cdot C_{in} + Y \cdot C_{in} + X \cdot Y \cdot C_{in}$ simplifies to $X \cdot C_{in} + Y \cdot C_{in}$ (Try it!)

$$Sum = X' \cdot Y' \cdot C_{in} + X' \cdot Y \cdot C'_{in} + X \cdot Y' \cdot C'_{in} + X \cdot Y \cdot C_{in}$$

Summary and suggested reading

- FPGAs (Section 9.8)
- Decoders (see earlier lecture)
- Adders and counters (Sections 4.7 & 12.3)
- Shannon's expansion

Roth and Kinney *Fundamentals of Logic Design*