

King Fahd University of Petroleum and Minerals
College of Computer Science and Engineering
Computer Engineering Department

COE 202: Digital Logic Design (3-0-3)
Term 151 (Fall 2015-2016)
Major Exam 2
Saturday Nov. 21, 2015

Time: 120 minutes, Total Pages: 12

Name: KEY ID: _____ Section: _____

Notes:

- Do not open the exam book until instructed
- **No Calculators are allowed** (*basic, advanced, cell phones, etc.*)
- Answer all questions
- All steps must be shown
- Any assumptions made must be clearly stated

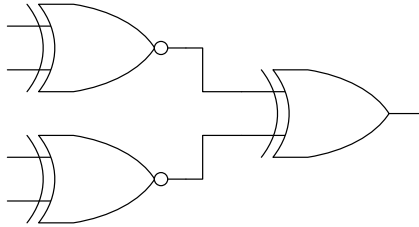
Question	Maximum Points	Your Points
1	7	
2	12	
3	7	
4	12	
5	16	
6	8	
7	11	
Total	73	

Question 1. Choose the correct answer (one answer only)

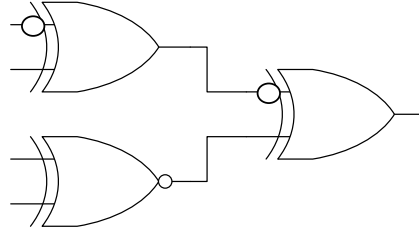
(7 Points)

1) Which of the following represents a 4-input XNOR function?

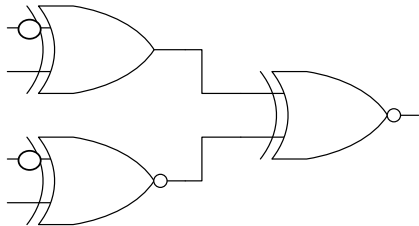
a)



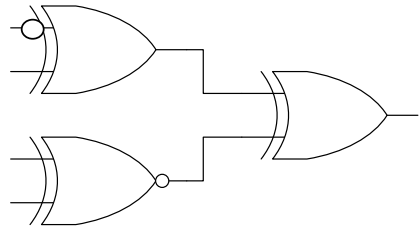
b)



c)



d)



2) NOR-OR (NOR first level and OR second level) function implementation is equivalent to:

- a) NAND-OR
- b) AND-NOR
- c) NOR-AND
- d) OR-NAND**

3) Minimizing the shown k-map results in:

- a) 2 terms, 2-variable each
- b) 2 terms, 1-variable each
- c) 1 term with 2 variables
- d) 1 term with 1 variable**

X			1
1			1

4) Considering $F(w, x, y, z)$, which of the following represents a single prime implicant having the largest area in a k-map (i.e., the largest group of 1's):

- a) $w + \bar{x} + y + z$
- b) $wx + \bar{y}\bar{z}$
- c) yz**
- d) $\bar{w}x\bar{y}z$

Question 2.**(12 Points)**

1) Represent $F(x, y, z) = (\bar{x} + z)(x + y + \bar{z})(x + \bar{y} + z)$ in the k-map shown below

		yz			
		00	01	11	10
x	0	1	0	1	0
	1	0	1	1	0

2) Given $F(A, B, C, D)$ shown in the k-map

a) List all essential prime implicants

$$AC, \bar{A}B\bar{C}, A\bar{B}D$$

b) Obtain minimized SOP expression of F

		CD			
		00	01	11	10
AB	00				
	01	1	1		
	11	1		1	1
	10		1	1	1

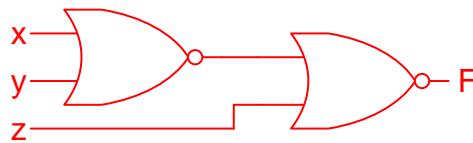
$$F = AC + \bar{A}B\bar{C} + A\bar{B}D + \begin{cases} B\bar{C}\bar{D} \\ A\bar{B}\bar{D} \end{cases}$$

- 3) Given function $F(w, x, y, z) = \sum(2,4,10,12,14)$ with don't care conditions $d(w, x, y, z) = \sum(1,5,6,8)$
 a) Use k-maps to provide minimized POS expression for F

$$F = \bar{z}(x + y)$$

		yz			
	wx	00	01	11	10
00		0	x	0	1
01		1	x	0	x
11		1	0	0	1
10		x	0	0	1

- b) Implement F using minimum number of 2-input NOR gates

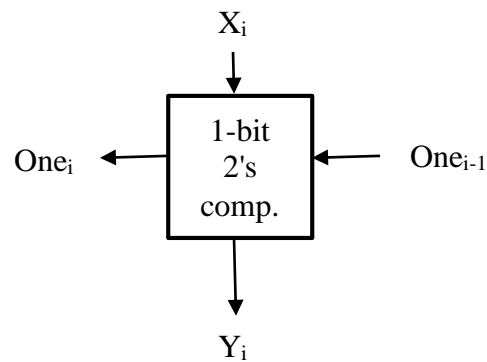


Question 3.**(7 Points)**

Given an n-bit signed 2's complement number, \mathbf{X} , it is required to design an iterative combinational circuit to compute the 2's complement of \mathbf{X} .

- (i) Show the inputs and outputs of the 1-bit 2's complement iterative cell to be used for designing the n-bit 2's complement circuit. (2 Points)
- (ii) Show the truth table of the 1-bit 2's complement cell. (2 Points)
- (iii) Obtain simplified equations for the outputs of the 1-bit 2's complement cell using only the following gate types: NOT, AND, OR, XOR. (2 Points)
- (iv) Using the 1-bit 2's complement cell, draw a block diagram for a circuit to compute the 2's complement of a 3-bit number \mathbf{X} . (1 Point)

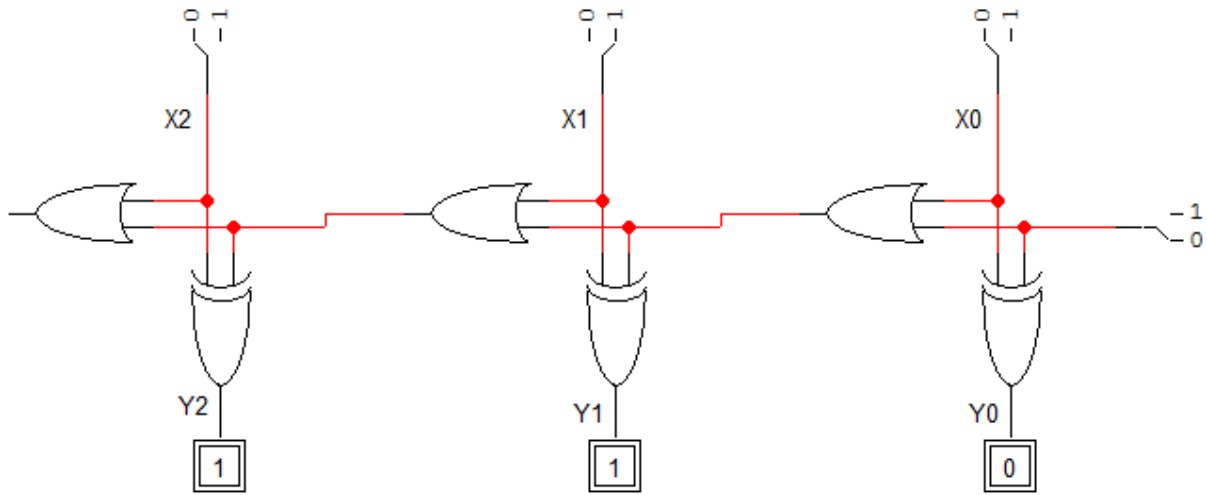
We will use a signal (One) that propagates between cells to indicate whether we have got one or not.



\mathbf{One}_{i-1}	\mathbf{X}_i	\mathbf{One}_i	\mathbf{Y}_i
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

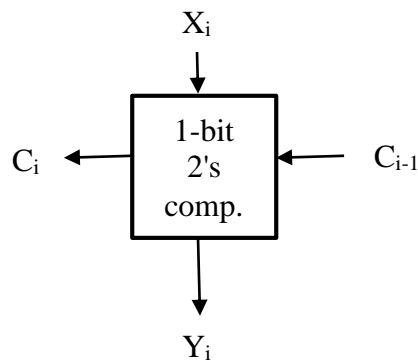
$$Y_i = X_i \oplus One_{i-1}$$

$$One_i = One_{i-1} + X_i$$



Alternative Solution:

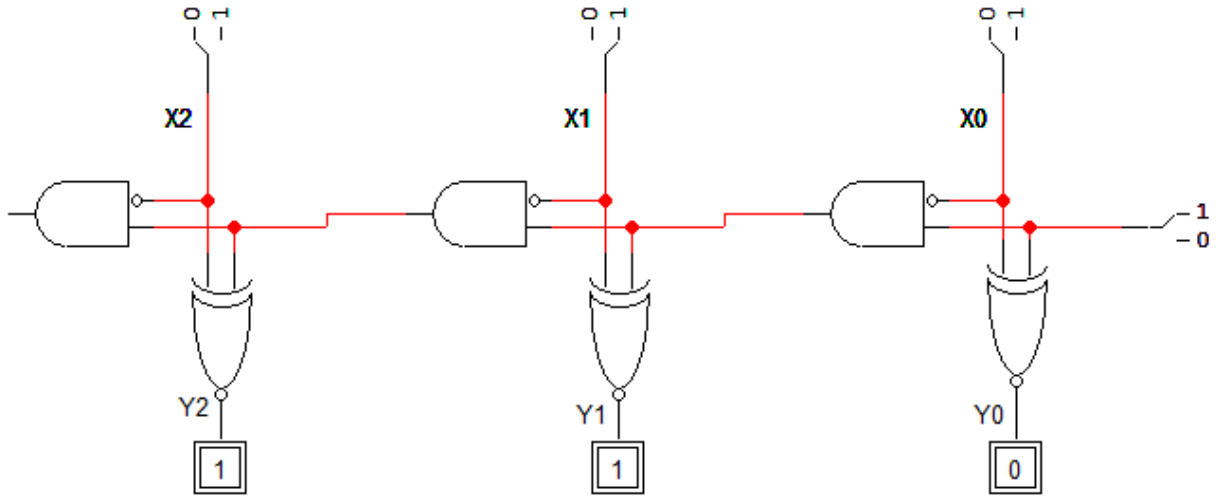
We will use a signal (C_{in}) that propagates between cells to indicate whether we have carry or not. The 2's complement will be computed as the 1's complement + 1.



C_{i-1}	X_i	C_i	Y_i
0	0	0	1
0	1	0	0
1	0	1	0
1	1	0	1

$$Y_i = (X_i \oplus C_{i-1})'$$

$$C_i = C_{i-1} X_i'$$

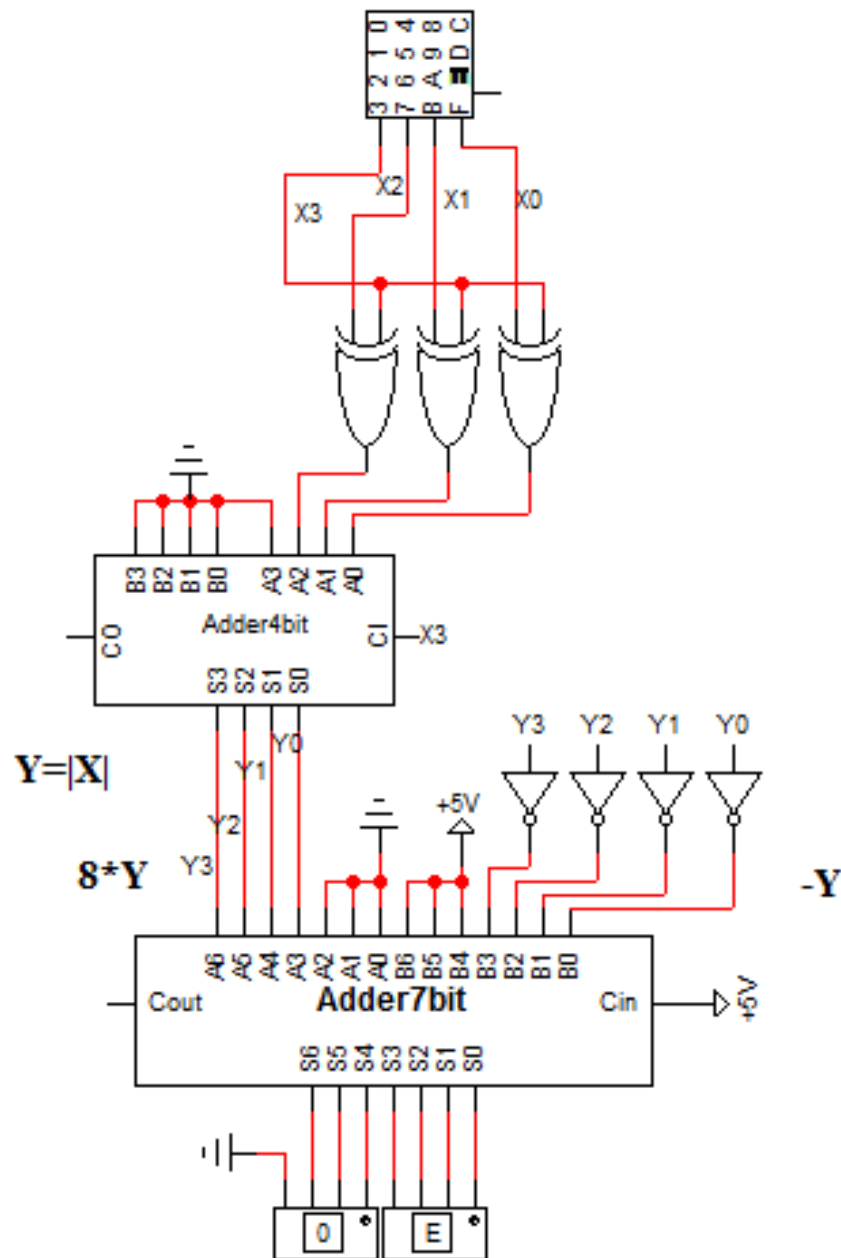


Question 4.

(12 Points)

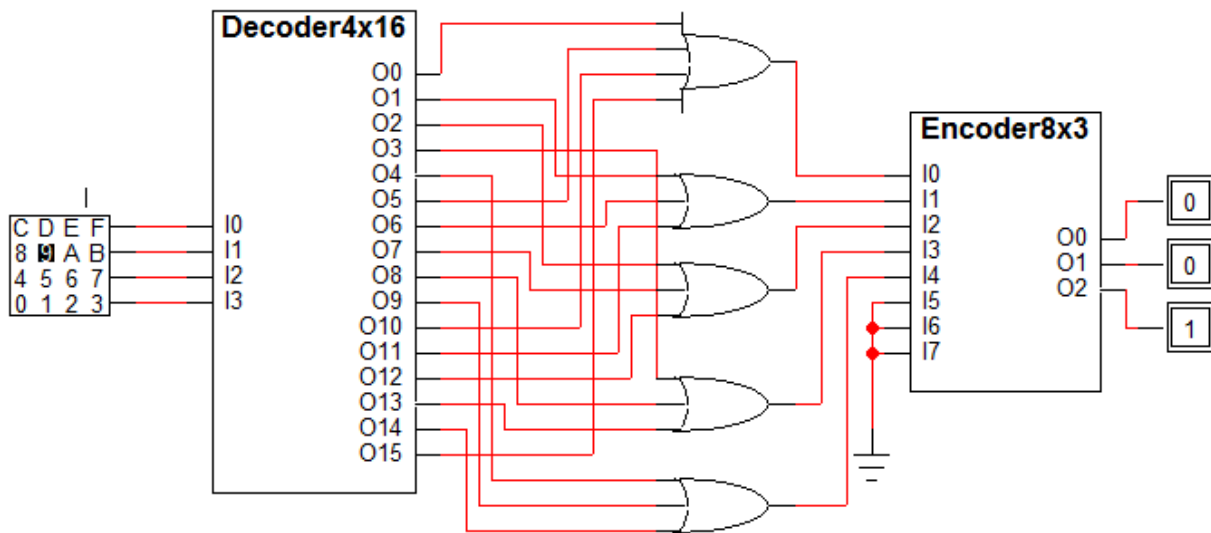
- (i) It is required to design a circuit to compute the equation $Y=|7*X|$, i.e., Y is equal to the absolute value of $7*X$, where X is a **4-bit signed number in 2's complement representation**. Your circuit should be designed using the minimum number and sizes of the following MSI components (Adders, Multiplexers) and additional logic gates if needed. Show clearly the size of all used components.

(7 Points)



- (ii) It is required to design a circuit to compute the equation $Y = X \bmod 5$, i.e. Y is the remainder of dividing X by 5, where X is a 4-bit unsigned number. For example, $9 \bmod 5 = 4$ and $10 \bmod 5 = 0$. Your circuit should be designed using the minimum number and sizes of the following MSI components (Decoder, Encoder) and additional logic gates if needed. Show clearly the size of all used components.

(5 Points)



Question 5.**(16 Points)**

(i) Fill in all blank cells in the table below. [4 points]

Binary (6-bits)	Equivalent decimal value with the binary interpreted as:			
	Unsigned integer	Signed-magnitude number	Signed-1's complement number	Signed-2's complement number
110110	54	-22	-9	-10

(ii) Fill in all blank cells in the table below. [6 points]

Decimal	Binary representation in 6 bits :		
	Signed-magnitude representation	Signed-1's complement representation	Signed-2's complement representation
+ 29	011101	011101	011101
- 29	111101	100010	100011
	Binary representation in 8 bits :		
	Signed-magnitude representation	Signed-1's complement representation	Signed-2's complement representation
- 29	10011101	11100010	11100011

(iii) Show how the following arithmetic operations are performed using 6-bit signed 2's-complement system. Check for overflow and mark clearly any overflow occurrences. [6 points]

$\begin{array}{r} 110100 \\ - 111110 \\ \hline \end{array}$	$\begin{array}{r} \mathbf{110100} \\ + \mathbf{000010} \\ \hline \mathbf{110110} \end{array}$	(1)	$\begin{array}{r} 110111 \\ + 111000 \\ \hline \mathbf{101111} \end{array}$	(2)
Overflow: Yes/ No			Overflow: Yes/ No	
$\begin{array}{r} 111110 \\ + 111111 \\ \hline \mathbf{111101} \end{array}$		(3)	$\begin{array}{r} 001101 \\ - 111101 \\ \hline \mathbf{001101} \\ + \mathbf{000011} \\ \hline \mathbf{010000} \end{array}$	(4)
Overflow: Yes/ No			Overflow: Yes/ No	

Question 6.

(8 Points)

(i) Show the implementation of the function $F(x,y,z) = \sum m(0,2,5,6,7)$

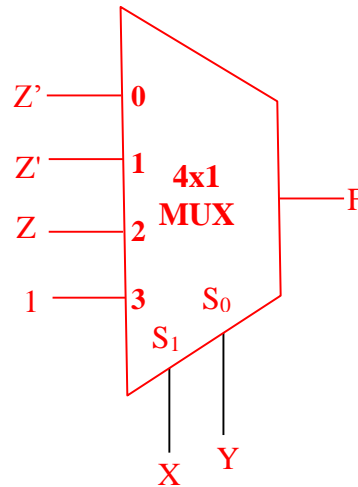
[6 points]

(a) Using a single MUX of minimum size

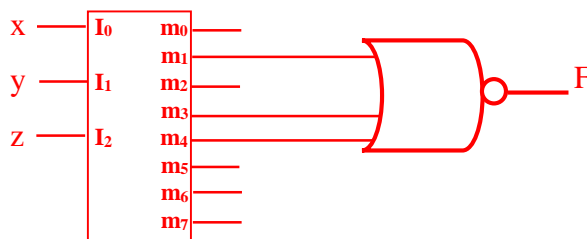
(b) Using a minimum size decoder and a single gate with minimum number of inputs

(a) F has 3 I/Ps \rightarrow 4-to-1 MUX:

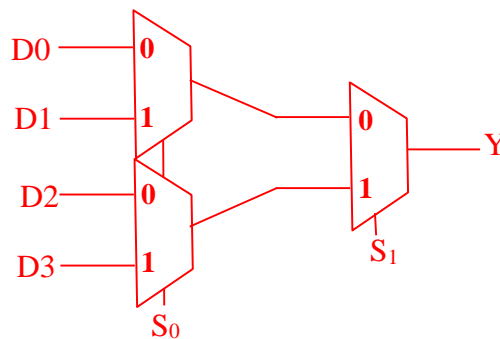
X	Y	Z	F	F
0	0	0	1	Z'
0	0	1	0	
0	1	0	1	Z'
0	1	1	0	
1	0	0	0	Z
1	0	1	1	
1	1	0	1	1
1	1	1	1	



(b) Since # of Maxterms is smaller than the # of Minterms, we implement F as POM. Note that $M_i = m'_i$, so we need to invert-AND F's Maxterms, i.e. NOR them:



(ii) Implement a 4-to-1 MUX using a minimum number of 2-to-1 MUXs. Clearly mark and label all inputs. [2 points]



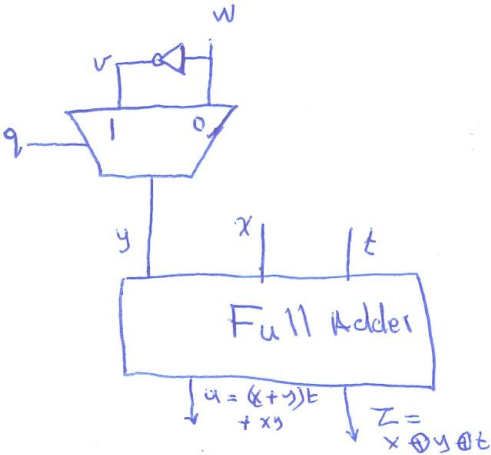
Question 7.

(11 Points)

(iii) A piece of hardware is described as a Verilog module. One of the given below Verilog codes is the correct description of this piece:

a. Indicate which of these codes is valid and which is invalid fully justifying your answer? (3 Points)

b. Give the logic diagram of this piece of hardware? (3 Points)

<p>VALID</p> 	<pre> module V_Q7A (output reg z, u, input x, w, t, q); wire v, y; assign y = (q==1'b1) ? v : w; //if q=1 y=v else y=w always @(x,y,t) begin z = (x^y)^t; u = (x&y) (x&t) (y&t); end not (v, w); //inverter gate instance endmodule </pre>
<p>INVALID (Continuous Assignment cannot be included within an always block)</p>	<pre> module V_Q7B (output reg z, u, input x, w, t, q); wire v; reg y; always @(x,q,t) begin assign y=(q==1'b1)? v:w; //if q=1 y=v else y=w z = (x^y)^t; u = (x&y) (x&t) (y&t); end not (v, w); //inverter gate instance endmodule </pre>
<p>INVALID (INSTANTATION statement cannot be included within an always block)</p>	<pre> module V_Q7C (output reg z, u, input x, w, t, q); reg v, y; always @(x,q,t, v) begin if (q ==1'b1) y = v ; else y = w; z = (x^y)^t; u = (x&y) (x&t) (y&t); not (v, w); //inverter gate instance end endmodule </pre>

- (iv) You are to write a test bench for the 4-bit adder module which has the following declaration: (5 Points)

```
module adder4 (output reg [3 : 0] sum , output reg cout , input [3 : 0] A , B) ;
```

Use the shown test patterns

Time Unit	A	B
0	5	6
10	15	9
20	9	3
30	13	14

```
module tb_add4 () ;
```

```
wire [3:0] sum ;
```

```
wire cout ;
```

```
reg [3:0] A, B ;
```

```
adder4 UUT1 (sum, cout, A, B) ;
```

```
initial
```

```
  begin
```

```
    A=5 ; B = 6;
```

```
    #10 A=15 ; B = 9;
```

```
    #10 A=9 ; B = 3;
```

```
    #10 A=13 ; B = 14;
```

```
  end
```

```
endmodule
```

Verilog Primitives

❖ Basic logic gates only

- ❖ **and**
- ❖ **or**
- ❖ **not**
- ❖ **buf**
- ❖ **xor**
- ❖ **nand**
- ❖ **nor**
- ❖ **xnor**

These gates are expandable: 1st node is O/P node, followed by 1, 2, 3 ... number of input nodes

Verilog Operators

{ }	concatenation	~	bit-wise NOT
+ - * / **	arithmetic	&	bit-wise AND
%	modulus		bit-wise OR
> >= < <=	relational	^	bit-wise XOR
!	logical NOT	^~ ~^	bit-wise XNOR
&&	logical AND	&	reduction AND
	logical OR		reduction OR
==	logical equality	~&	reduction NAND
!=	logical inequality	~	reduction NOR
===	case equality	^	reduction XOR
!==	case inequality	~^ ^~	reduction XNOR
? :	conditional	<<	shift left
		>>	shift right