Week 3 Review

Question #1

a) How do you write the number 78 as an 8-bit binary number?

128	64	32	16	8	4	2	1
0	1	0	0	1	1	1	0

b) What is the two's complement of `01101101`?

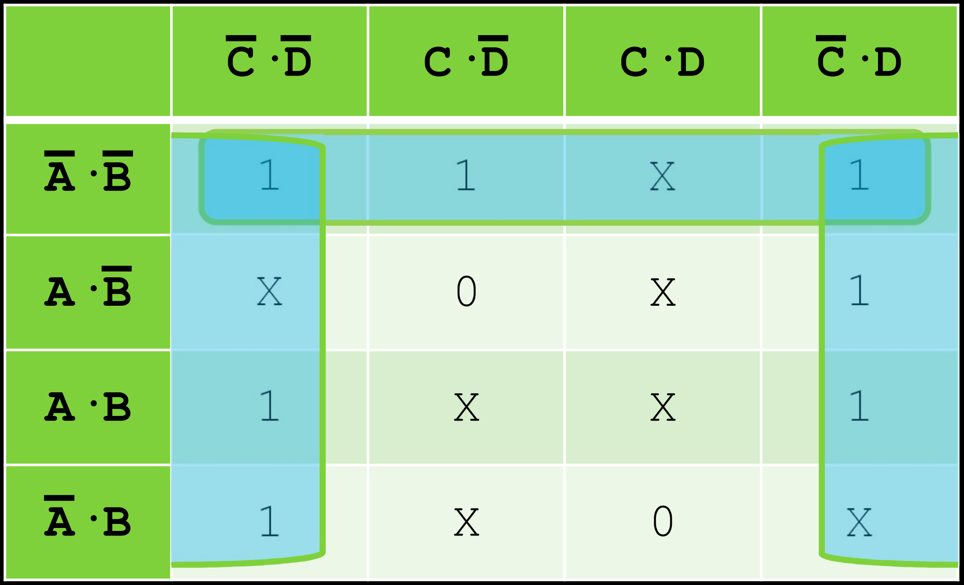
`10010011`

c) What is the sum of `01101101` and `01101101`?

`11011010` ← Note what's happening here!

Question #2

- What groupings are in the K-map on the right?

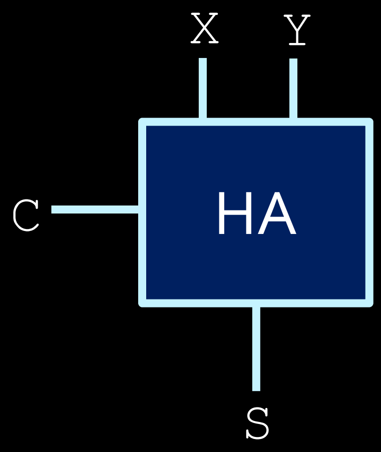


	$\overline{C}\cdot\overline{D}$	$C\cdot\overline{D}$	$C\cdot D$	$\overline{C}\cdot D$
$\overline{A}\cdot\overline{B}$	1	1	X	1
$A\cdot\overline{B}$	X	0	X	1
$A\cdot B$	1	X	X	1
$\overline{A}\cdot B$	1	X	0	X

- What logic equations do these groupings represent?

$$\boxed{\overline{A}\cdot\overline{B} + \overline{C}}$$

Question #3



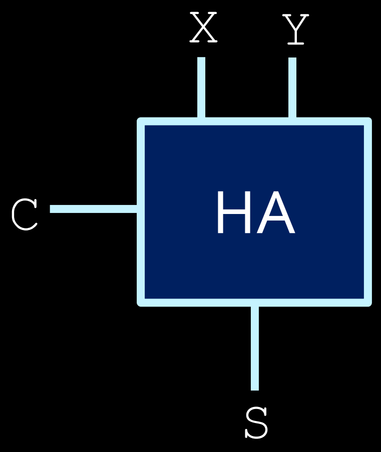
- Implement a half adder in Verilog.
- Step 1: What is the half adder logic equation?

$$C = X \cdot Y \qquad S = X \cdot \overline{Y} + \overline{X} \cdot Y = X \oplus Y$$

- Step 2: Equivalent Verilog components.

```
assign C = X & Y;
assign S = X & ~Y | ~X & Y;
```

Question #3 (cont'd)



- Step 3: What is the complete Verilog code for this device?

```
module half_adder(X, Y, C, S);
    input X, Y;
    output C, S;

    assign C = X & Y;
    assign S = X & ~Y | ~X & Y;
endmodule
```

Question #4

- How can you use a `case` statement to rewrite the 7-segment display in Verilog?

```
module seven_seg (seg,bin);

   input [3:0] bin;
   output [0:6] seg;
   reg [0:6] seg;
```

Question #4 (cont’d)

```
// start with always block for case statement

   always @(bin)
      begin
         case (bin) //case statement
            0 : seg = 7'b0000001;
            1 : seg = 7'b1001111;
            2 : seg = 7'b0010010;
            3 : seg = 7'b0000110;
            4 : seg = 7'b1001100;
            ...
            default : seg = 7’b1111111;
         endcase
      end
endmodule
```