Dongyun Lee

ID: 112794190

Lab3:

ESE382-L01

Bench #4

1. For a Boolean function that has a smaller number of input combinations where its output is 0 than it does where its output is 1, why would it be better to write a CSOP expression for the input combinations where the output is 0 and then complement that expression or write a CPOS expres- sion for the function than write a a CSOP expression for the input combinations where the output is 1?

It would be better to write a CSOP expression when there are more number of output '0's than '1's because it makes the expression more simple and short. It depends on '0's or '1's on the output.

2. Compare the HDL Analyst hierarchical views of your four designs. Are the diagrams identical? If not, how do they differ? Are they functionally the same? If not, how do they differ functionally?

They are functionally the same. However the diagrams are not perfectly identical. This is because one is in CSOP form and the other one is in CPOS form. The implementations of AND gates and OR gates would be different. However, they functions the same.

3. Compare the pre-fit equations and post-fit equations from ispLEVER Classic for the four designs. Are the equations the same? If not, how do they differ?

Both pre-fit equations and post-fit equations are showing the same information. However, pre-fit equation focus more on the logic gates and post-fit equation shows more the actual gates and integrated circuit inside. Therefore, pre-fit equation represents logical design and post-fit equation represents how the design is mapped into an actual PLD.

Notes

a. In the pre-fit and post-fit equations the # symbol represents OR, the & symbol represents AND, and the ! symbol represents NOT.

b. The pre-fit and post-fit equations listings give equations for both the normal and complement of each function. This is done because it is sometimes easier (requires fewer product terms) to imple- ment the complement of a function and then configure the PLD to complement the output of the OR gate creating the desired function than it is to implement the normal form of the function.

4. From the ispLEVER Classic chip report how much of the isp22V10's resources (inputs, outputs, gates, flip-flops) were used for the decoder design?

The ispLEVER Classic chip report represents 4 inputs, 4 outputs, 12 product terms used, no flip-flops are used.

5. From the chip report determine exactly which equations were actually implemented in the PLD.

The equation shown in chip report is in CSOP form and it is post-fit equation. The program has optimized the equation from VHDL code to make it fit to the PLD.

6. Explain the differences in the output values when input combinations that are not valid XS3 codes are applied to your four designs. What determined these output values for each of your designs? If the system providing inputs to your design generates only valid XS3 codes, does each of your designs meet the design requirements? answer these question based on the lab instruction. conditional signal and selected signal are not done. so avoid those two when answering the questions

There are not big differences between CSOP form and CPOS form. However, using the selected signal or conditional signal, it would be different because the code can implement 'don't cares.' The logic expressions that I have written in VHDL determines the output. Also, the synthesis program might optimize the expressions. If I wanted the design to be only vaild on XS3 code then the design has met the requirements. From the hardware test, I have observe that it converts from XS3. To BCD.

Module : 'converter_xs3_bcd'

Input files:

ABEL PLA file : converter_xs3_to_bcd.tt3
Device library : P22V10GC.dev

Output files:

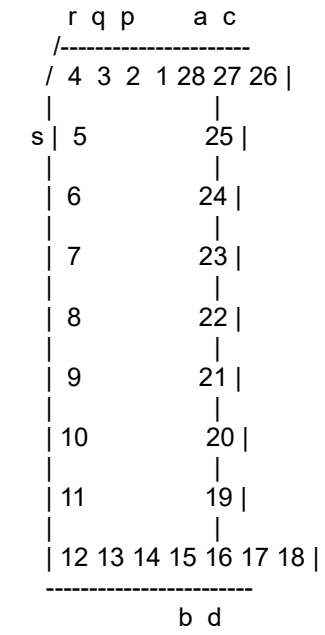
Report file : converter_xs3_to_bcd.rpt
Programmer load file : converter_xs3_to_bcd.jed

P22V10GC Programmed Logic:

```
d    = (  p & !q & r & s
       #   p & q & !r & !s );

c    = (  p & !q & !r
       #   !p & q & r & s
       #   p & !q & !s );

b    = (  !p & q & !r & s
       #   p & !q & !r & s
       #   !p & q & r & !s
       #   p & !q & r & !s );

a    = (  !p & q & !s
       #   p & !q & !s
       #   q & !r & !s );
```


P22V10GC Chip Diagram: (PLCC package)

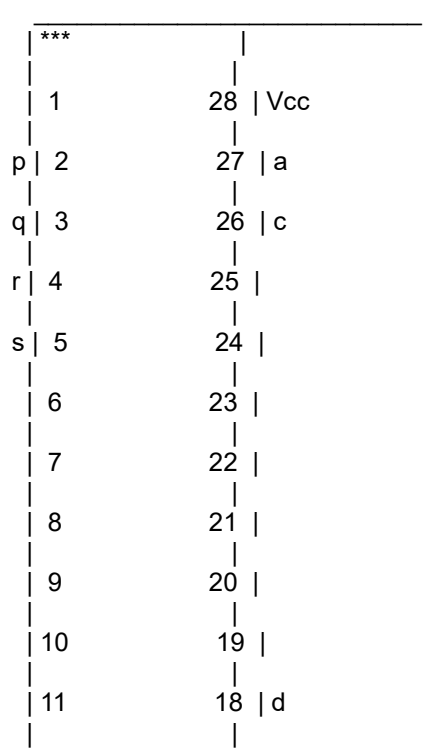
P22V10GC

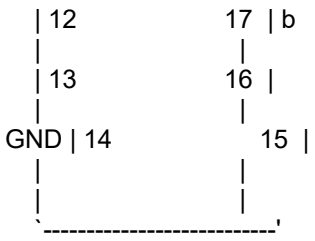


SIGNATURE: N/A

P22V10GC Chip Diagram: (SSOP package)

P22V10GC





SIGNATURE: N/A

P22V10GC Resource Allocations:

Device Resources	Resource Available	Design Requirement	Unused
Input Pins:			
Input:	12	4	8 (66 %)
Output Pins:			
In/Out:	10	4	6 (60 %)
Output:	-	-	-
Buried Nodes:			
Input Reg:	-	-	-
Pin Reg:	10	0	10 (100 %)
Buried Reg:	-	-	-

P22V10GC Product Terms Distribution:

Signal Name	Pin Assigned	Terms Used	Terms Max	Terms Unused
d	18	2	10	8
c	26	3	10	7
b	17	4	8	4
a	27	3	8	5

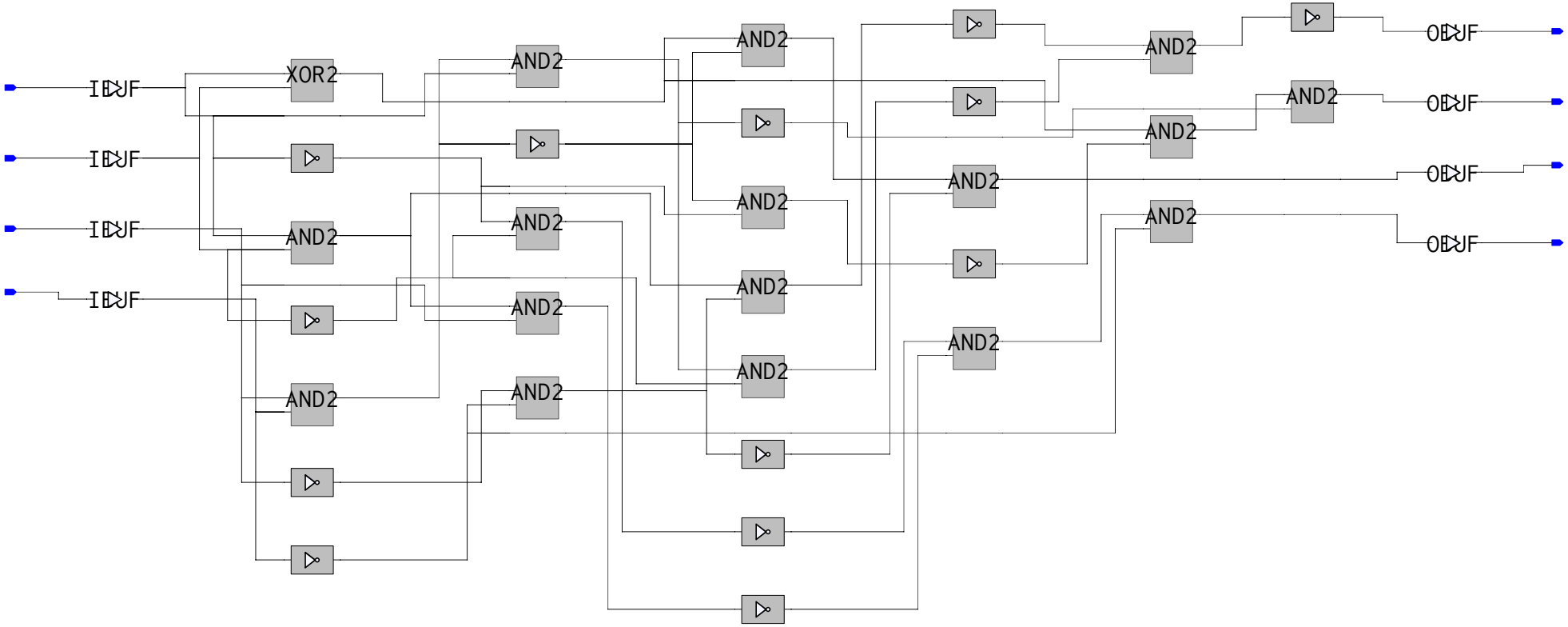
==== List of Inputs/Feedbacks ====

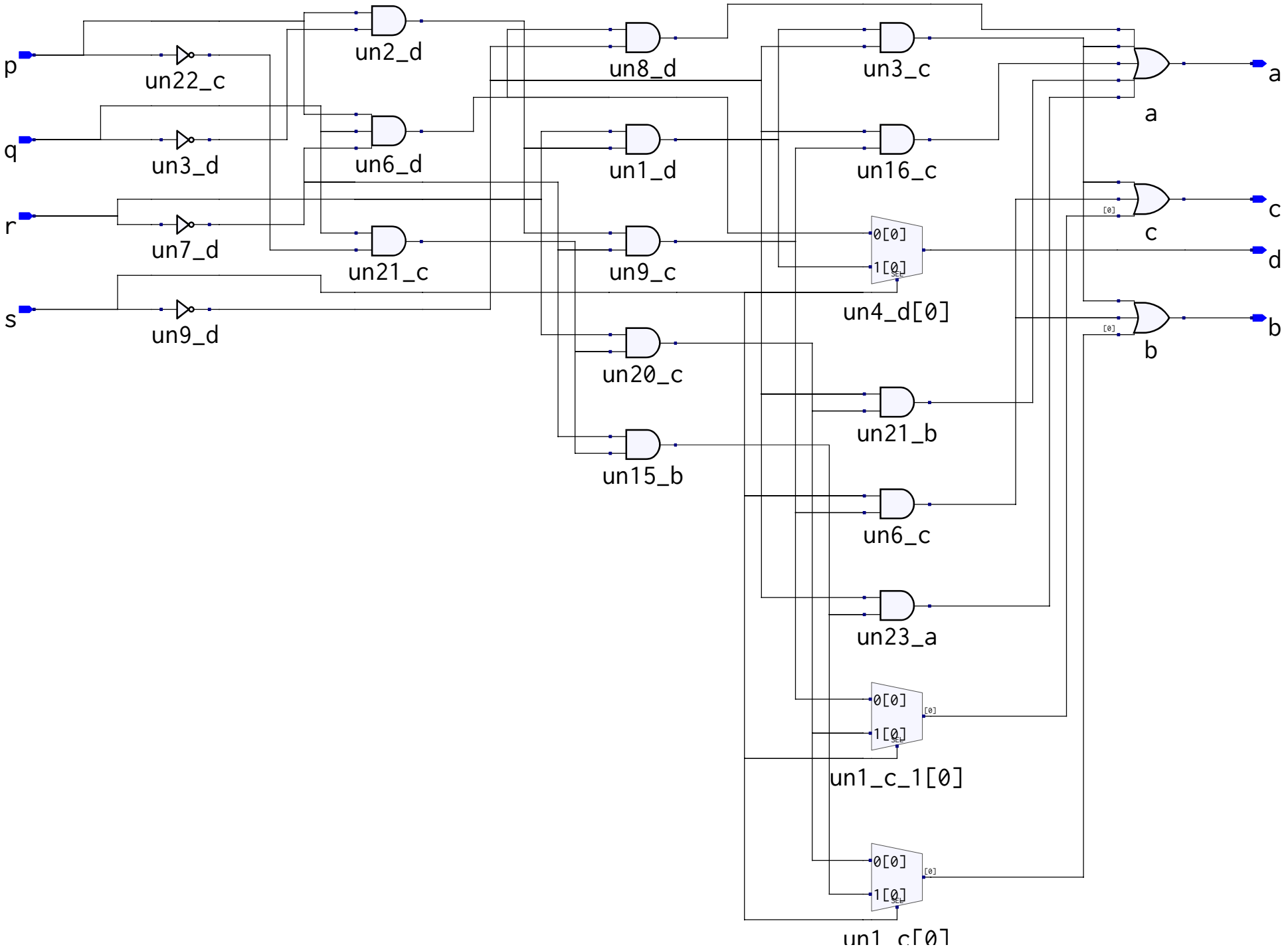
Signal Name	Pin	Pin Type
p	2	CLK/IN

q | 3 | INPUT
r | 4 | INPUT
s | 5 | INPUT

P22V10GC Unused Resources:

Pin Number	Pin Type	Product Terms	Flip-flop Type
6	INPUT	-	-
7	INPUT	-	-
9	INPUT	-	-
10	INPUT	-	-
11	INPUT	-	-
12	INPUT	-	-
13	INPUT	-	-
16	INPUT	-	-
19	BIDIR	NORMAL 12	D
20	BIDIR	NORMAL 14	D
21	BIDIR	NORMAL 16	D
23	BIDIR	NORMAL 16	D
24	BIDIR	NORMAL 14	D
25	BIDIR	NORMAL 12	D


XOR2
AND2
AND2
AND2
AND2
AND2
AND2
AND2
AND2
AND2
AND2
AND2
AND2
AND2


p
q
r
s
un22_c
un3_d
un7_d
un9_d
un2_d
un6_d
un21_c
un8_d
un1_d
un9_c
un20_c
un15_b
un3_c
un16_c
0[0]
1[0]
SEL
un4_d[0]
un21_b
un6_c
un23_a
0[0]
1[0]
SEL
[0]
un1_c_1[0]
0[0]
1[0]
SEL
[0]
un1_c[0]
a
a
[0]
c
c
[0]
b
b
d