

# IC Design HW3

B10901176 蔡弘祥

## 1. Circuit Overview

Some Checkers	$\lceil 0[5:4] \rceil$	$\lceil n0[1:0] \rceil$	flush checker	out	flush
	$\lceil 1[5:4] \rceil$	$\lceil n1[1:0] \rceil$		out'	flush'
	$\lceil 2[5:4] \rceil$	$\lceil n2[1:0] \rceil$			
	$\lceil 3[5:4] \rceil$	$\lceil n3[1:0] \rceil$			
	$\lceil 4[5:4] \rceil$	$\lceil n4[1:0] \rceil$			
	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$			
	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$			
Some Checkers	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$	straight checker	out	Straight
	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$		out'	Straight'
	$\lceil 4[3:0] \rceil$	$\lceil n4[3:0] \rceil$			
	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$			
	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$			
	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$			
	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$			
Some Checkers	$\lceil 4[3:0] \rceil$	$\lceil n4[3:0] \rceil$	4 of a kind checker	out	4ofakind
	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$		out'	4ofakind'
	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$			
	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$			
	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$			
	$\lceil 4[3:0] \rceil$	$\lceil n4[3:0] \rceil$			
	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$			
Some Checkers	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$	full house checker	out	full house
	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$		out'	full house'
	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$			
	$\lceil 4[3:0] \rceil$	$\lceil n4[3:0] \rceil$			
	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$			
	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$			
	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$			
Some Checkers	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$	3 of a kind possible checker	out	3ofakindpossible
	$\lceil 4[3:0] \rceil$	$\lceil n4[3:0] \rceil$		out'	3ofakindpossible'
	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$			
	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$			
	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$			
	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$			
	$\lceil 4[3:0] \rceil$	$\lceil n4[3:0] \rceil$			
Some Checkers	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$	2 pairs possible checker	out	2pairspossible
	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$		out'	2pairspossible'
	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$			
	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$			
	$\lceil 4[3:0] \rceil$	$\lceil n4[3:0] \rceil$			
	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$			
	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$			
Some Checkers	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$	1 pair possible checker	out	1pairpossible
	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$		out'	(pairpossible)
	$\lceil 4[3:0] \rceil$	$\lceil n4[3:0] \rceil$			
	$\lceil 0[3:0] \rceil$	$\lceil n0[3:0] \rceil$			
	$\lceil 1[3:0] \rceil$	$\lceil n1[3:0] \rceil$			
	$\lceil 2[3:0] \rceil$	$\lceil n2[3:0] \rceil$			
	$\lceil 3[3:0] \rceil$	$\lceil n3[3:0] \rceil$			

Type[3]	<p>flash straight</p> <p>Type[3] <math>\Rightarrow</math></p>
Type[2]	<p>'4ofakind' full house!</p> <p>temp[2]</p>
Type[1]	<p>temp[i]</p> <p>Type[3] <math>\Rightarrow</math></p> <p>'4ofakind' 3ofakindpossible &gt; pairspossible</p>
Type[0]	<p>temp[0]</p> <p>Type[3] <math>\Rightarrow</math></p> <p>&gt; pairspossible 1 pairpossible</p>

## 2. Temp[2:0]

Explanation:

For those combinations which are not straight, flush, or straight flush, we first determine the output named temp[2:0] using some checkers mentioned above.

Checker Name	Abbreviation
4 of a kind	4O
Full house	FH
3 of a kind possible	3O
2 pairs possible	2P
1 pair possible	1P

temp[2:0]	$\begin{array}{c} \text{2P, 1P} \\ \hline \begin{array}{c} 4O=0 & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 & 0 & x \\ FH,3O & x & 0 & x & x \\ 11 & x & x &   & x \\ 10 & x & x & x & x \end{array} \quad \begin{array}{c} 4O=1 & 00 & 01 & 11 & 10 \\ \hline 00 & 1 & 1 & 1 & 1 \\ FH,3O & 1 & 1 & 1 & 1 \\ 11 & x & x & x & x \\ 10 & x & x & x & x \end{array} \end{array}$

temp[2]	$\begin{array}{c} \text{2P, 1P} \\ \hline \begin{array}{c} 4O=0 & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 & 0 & x \\ FH,3O & x & 0 & x & x \\ 11 & x & x &   & x \\ 10 & x & x & x & x \end{array} \quad \begin{array}{c} 4O=1 & 00 & 01 & 11 & 10 \\ \hline 00 & 1 & 1 & 1 & 1 \\ FH,3O & 1 & 1 & 1 & 1 \\ 11 & x & x & x & x \\ 10 & x & x & x & x \end{array} \end{array}$
	$\text{temp}[2] = 4O + FH = (4O'FH)'$

temp[1]	$\begin{array}{c} \text{2P, 1P} \\ \hline \begin{array}{c} 4O=0 & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 &   & x \\ FH,3O & x & 1 & x & x \\ 11 & x & x &   & x \\ 10 & x & x & x & x \end{array} \quad \begin{array}{c} 4O=1 & 00 & 01 & 11 & 10 \\ \hline 00 & 1 & 1 & 1 & 1 \\ FH,3O & 1 & 1 & 1 & 1 \\ 11 & x & x & x & x \\ 10 & x & x & x & x \end{array} \end{array}$
	$\text{temp}[1] = 4O + 3O + 2P = (4O'3O'2P)'$

temp[0]	$\begin{array}{c} \text{2P, 1P} \\ \hline \begin{array}{c} 4O=0 & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 1 & 0 & x \\ FH,3O & x & 1 & x & x \\ 11 & x &   & 0 & x \\ 10 & x & x & x & x \end{array} \quad \begin{array}{c} 4O=1 & 00 & 01 & 11 & 10 \\ \hline 00 & 1 & 1 & 1 & 1 \\ FH,3O & 1 & 1 & 1 & 1 \\ 11 & x & x & x & x \\ 10 & x & x & x & x \end{array} \end{array}$
	$\text{temp}[0] = 4O + 2P'1P = [4O'(2P'1P)]'$

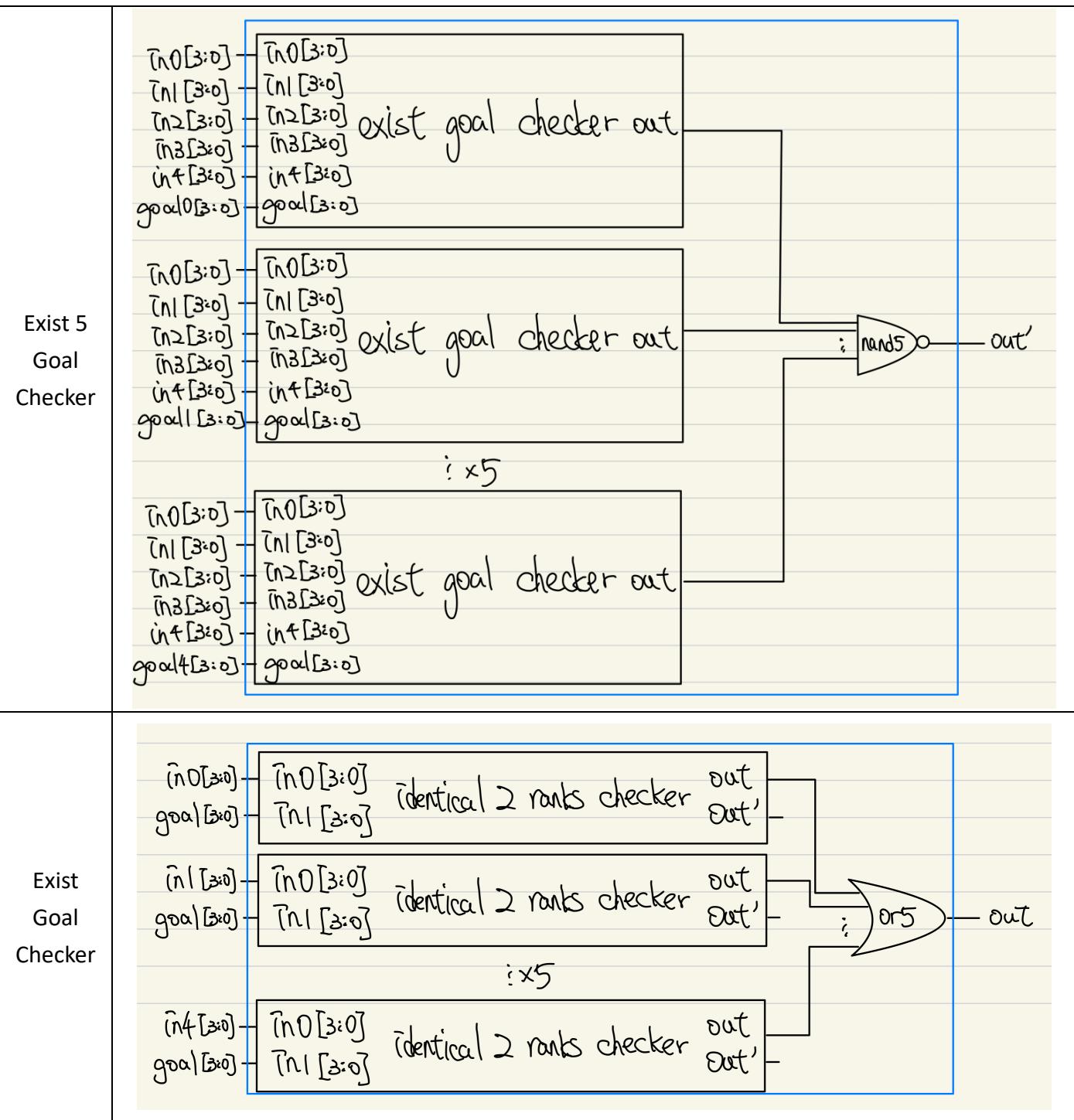
### 3. Self-defined Sub Module Overview

Sub Module Name	input	output	Explanation
Flush Checker	in0[1:0] in1[1:0] in2[1:0] in3[1:0] in4[1:0]	out out'	Input the first two bits of all cards out = true if they are flush or not
Straight Checker	in0[3:0] in1[3:0] in2[3:0] in3[3:0] in4[3:0]	out out'	Input the last four bits of all cards out = true if they are straight or not
Exist 5 Goal Checker	goal0[3:0] goal1[3:0] goal2[3:0] goal3[3:0] goal4[3:0] in0[3:0] in1[3:0] in2[3:0] in3[3:0] in4[3:0]	out'	Input the five numbers from the cards and the five goal numbers out = true if all goal numbers exist in the cards
Exist Goal Checker	goal[3:0] in0[3:0] in1[3:0] in2[3:0] in3[3:0] in4[3:0]	out	Input the five numbers from the cards and a goal number out = true if the goal number exists in the cards
4 of a Kind Checker	in0[3:0] in1[3:0] in2[3:0] in3[3:0] in4[3:0]	out out'	Input the last four bits of all cards out = true if they are 4 of a kind or not
Full House Checker	in0[3:0] in1[3:0] in2[3:0] in3[3:0] in4[3:0]	out out'	Input the last four bits of all cards out = true if they are full house or not
3 of a kind Possible Checker	in0[3:0] in1[3:0] in2[3:0]	out out'	Input the last four bits of all cards out = true if there are 3 cards with the same rank

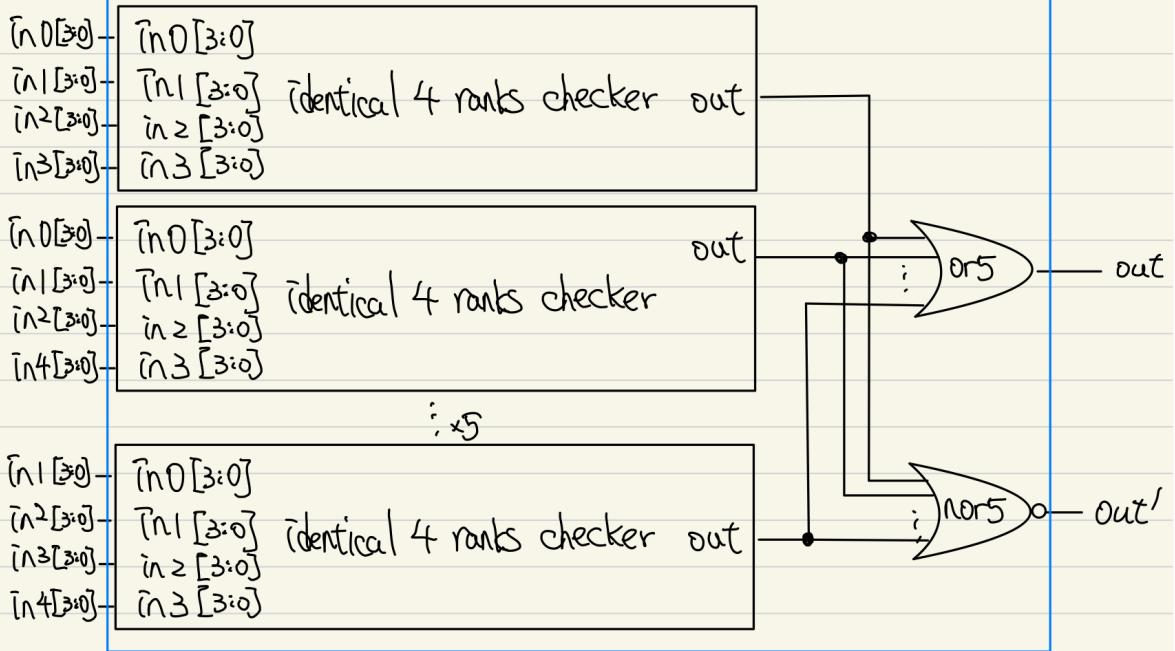
	in3[3:0] in4[3:0]		
2 Pairs Possible Checker	in0[3:0] in1[3:0] in2[3:0] in3[3:0] in4[3:0]	out out'	Input the last four bits of all cards out = true if there are 2 pairs or 4 of a kind
2 Pairs Possible Sub Checker	in0[3:0] in1[3:0] in2[3:0] in3[3:0]	out out'	Input the four numbers from the cards out = true if there are at least two pairs among the cards (all same rank). All 4 cards with the same rank will also return true.
1 Pair Possible Checker	in0[3:0] in1[3:0] in2[3:0] in3[3:0] in4[3:0]	out out'	Input the last four bits of all cards out = true if there are at least a pair among the cards
Identical 4 Ranks Checker	in0[3:0] in1[3:0] in2[3:0] in3[3:0]	out	Input four numbers out = true if they are the same
Identical 3 Ranks Checker	in0[3:0] in1[3:0] in2[3:0]	out	Input three numbers out = true if they are the same
Identical 2 Ranks Checker	in0[3:0] in1[3:0]	out out'	Input two numbers out = true if they are the same
Same 5 Bits Checker	in0 in1 in2 in3 in4	out	Input five bits out = true if they are the same
Same 4 Bits Checker	in0 in1 in2 in3	out	Input four bits out = true if they are the same
Same 3 Bits Checker	in0 in1 in2	out	Input three bits out = true if they are the same
MUX4C	A,B C,D CTRL1 CTRL2	out	Cascading Muxes, CTRL2 should be the slower signal

#### 4. Sub Module Circuit Diagram

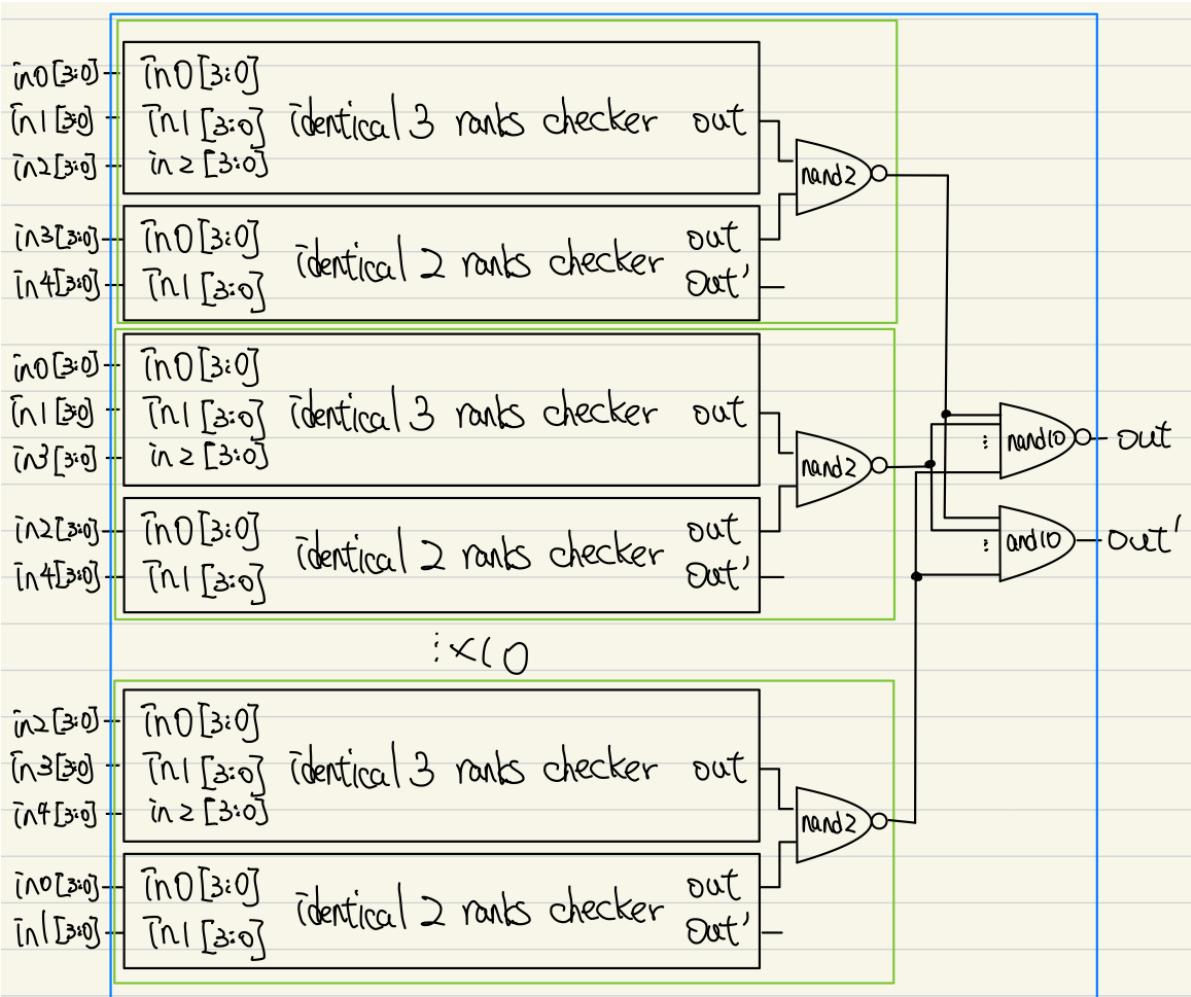
Sub Module Name	Diagram
Flush Checker	<p>Some 5 bits checker</p> <pre>     graph LR       subgraph Top [ ]         direction TB         T0["in0[0]"]         T1["in0[1]"]         T2["in0[2]"]         T3["in0[3]"]         T4["in0[4]"]         T0 --- T1         T1 --- T2         T2 --- T3         T3 --- T4       end       subgraph Bottom [ ]         direction TB         B0["in1[0]"]         B1["in1[1]"]         B2["in1[2]"]         B3["in1[3]"]         B4["in1[4]"]         B0 --- B1         B1 --- B2         B2 --- B3         B3 --- B4       end       T4 --&gt; Out1       B4 --&gt; Out2       Out1 --&gt; And2((and2))       Out2 --&gt; Nand2((nand2))       And2 --&gt; Out       Nand2 --&gt; OutP       Out --- OutP     </pre>
Straight Checker	<p>exist 5 goal checker</p> <pre>     graph LR       subgraph Top [ ]         direction TB         T0["in0[3:0]"]         T1["in0[3:0]"]         T2["in0[3:0]"]         T3["in0[3:0]"]         T4["in0[3:0]"]         T5["goal0[3:0]"]         T6["goal1[3:0]"]         T7["goal2[3:0]"]         T8["goal3[3:0]"]         T9["goal4[3:0]"]         T0 --- T1         T1 --- T2         T2 --- T3         T3 --- T4         T4 --- T5         T5 --- T6         T6 --- T7         T7 --- T8         T8 --- T9       end       subgraph Middle [ ]         direction TB         M0["in0[3:0]"]         M1["in0[3:0]"]         M2["in0[3:0]"]         M3["in0[3:0]"]         M4["in0[3:0]"]         M5["goal0[3:0]"]         M6["goal1[3:0]"]         M7["goal2[3:0]"]         M8["goal3[3:0]"]         M9["goal4[3:0]"]         M0 --- M1         M1 --- M2         M2 --- M3         M3 --- M4         M4 --- M5         M5 --- M6         M6 --- M7         M7 --- M8         M8 --- M9       end       subgraph Bottom [ ]         direction TB         B0["in0[3:0]"]         B1["in0[3:0]"]         B2["in0[3:0]"]         B3["in0[3:0]"]         B4["in0[3:0]"]         B5["goal0[3:0]"]         B6["goal1[3:0]"]         B7["goal2[3:0]"]         B8["goal3[3:0]"]         B9["goal4[3:0]"]         B0 --- B1         B1 --- B2         B2 --- B3         B3 --- B4         B4 --- B5         B5 --- B6         B6 --- B7         B7 --- B8         B8 --- B9       end       T9 --&gt; Out1       M9 --&gt; Out2       B9 --&gt; Out3       Out1 --&gt; And10((and10))       Out2 --&gt; Nand10((nand10))       And10 --&gt; Out       Nand10 --&gt; OutP       Out --- OutP       subgraph Multiplier [ ; x 10 ]         direction TB         M0         M1         M2         M3         M4         M5         M6         M7         M8         M9       end     </pre>



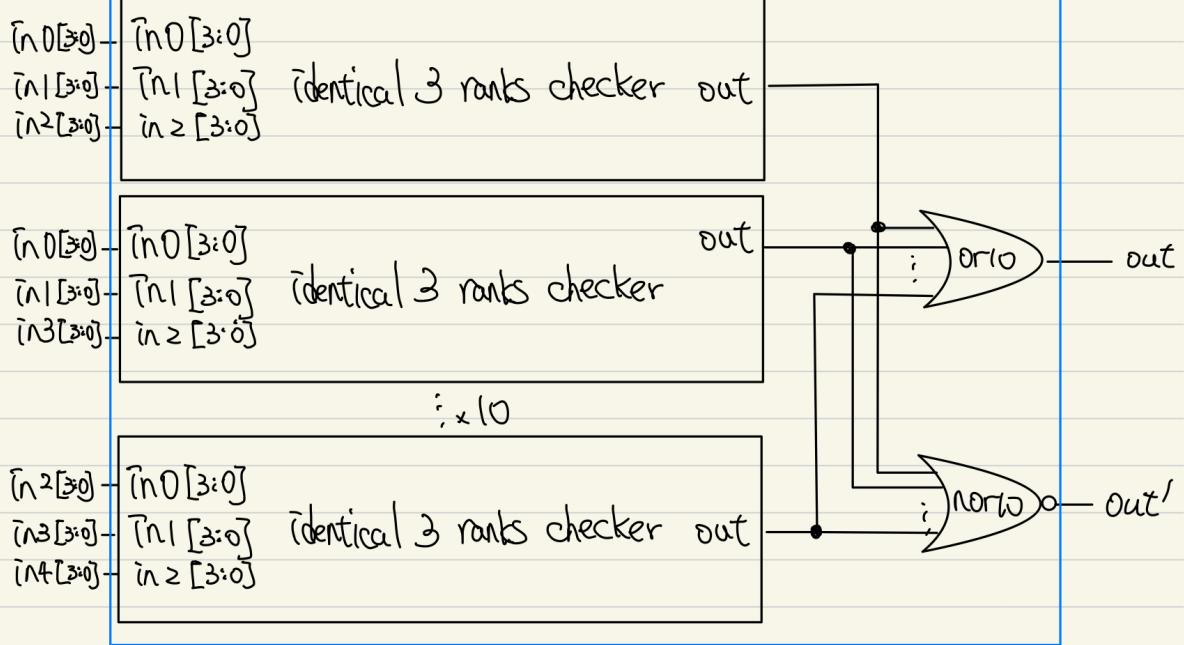
4 of a Kind Checker



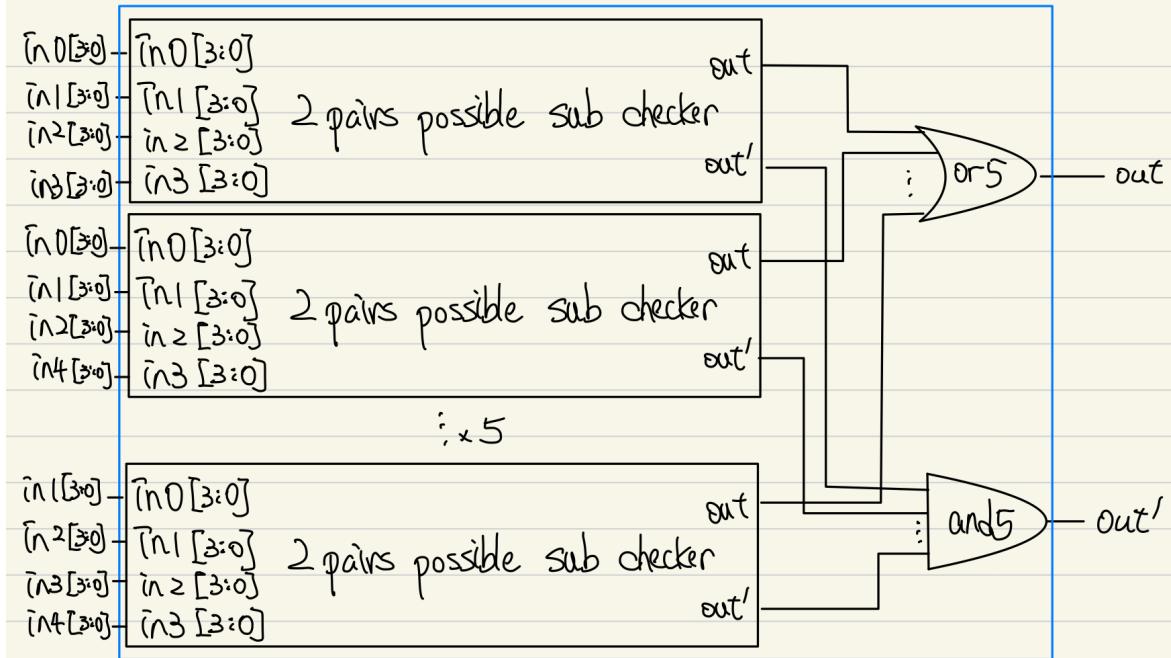
Full House Checker



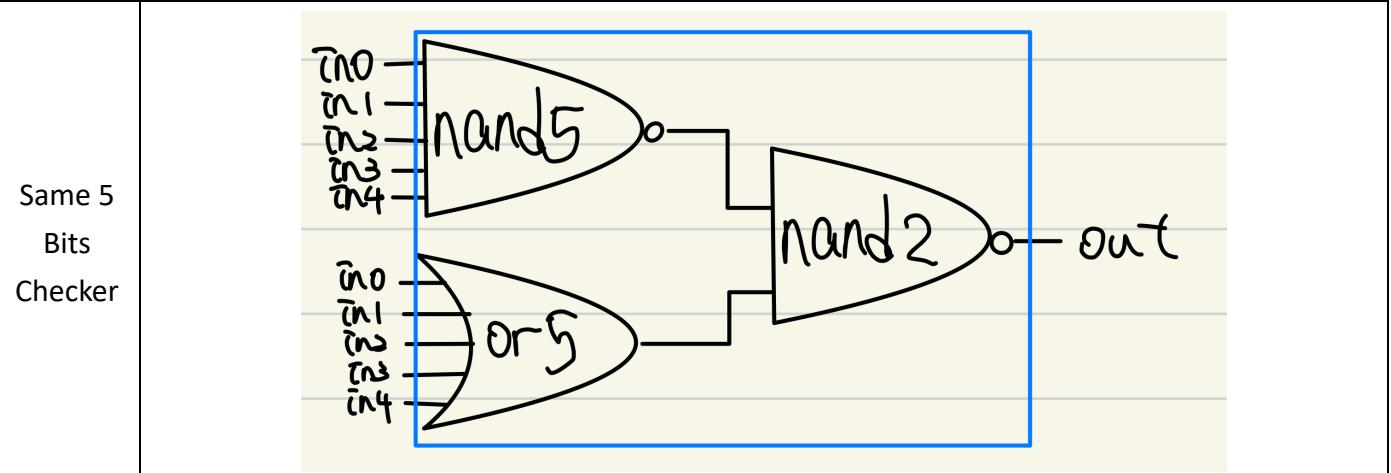
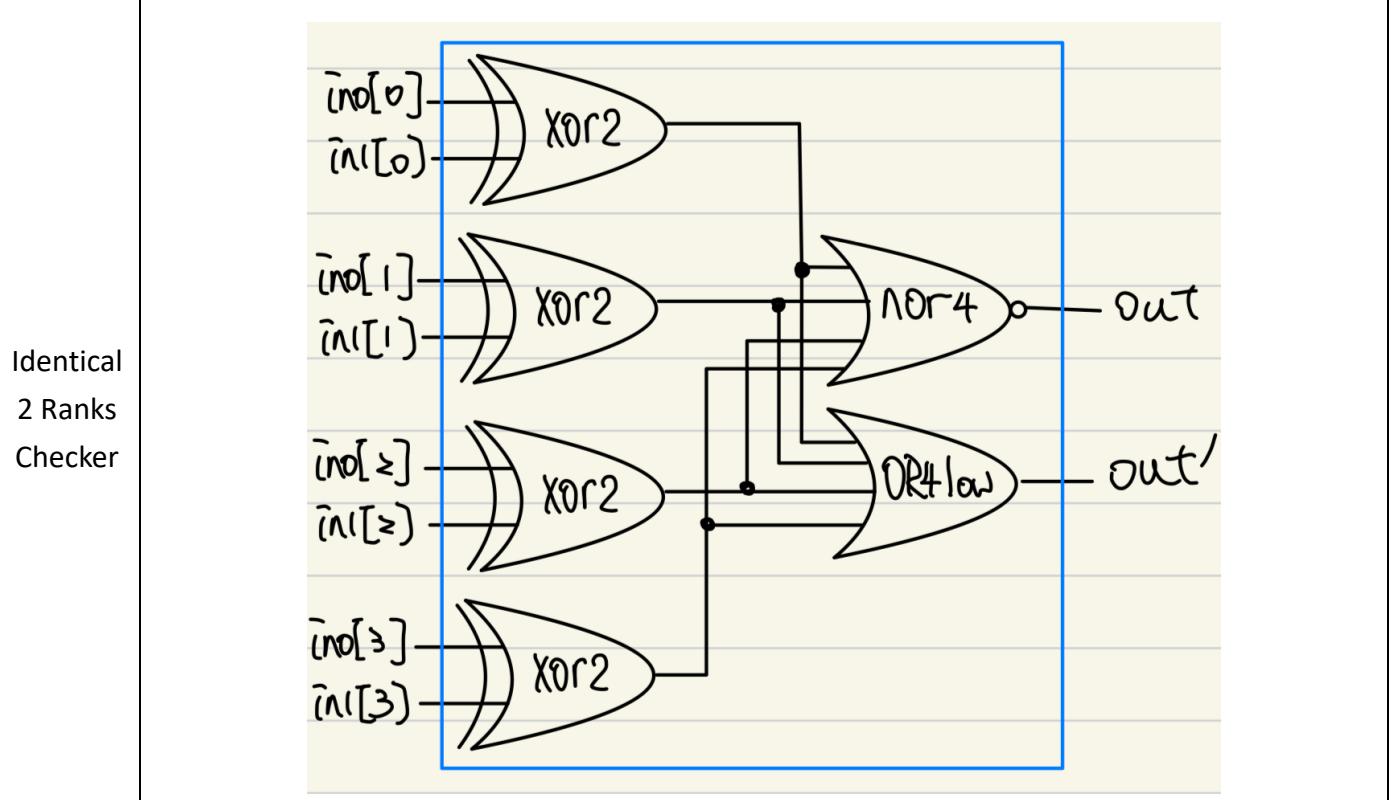
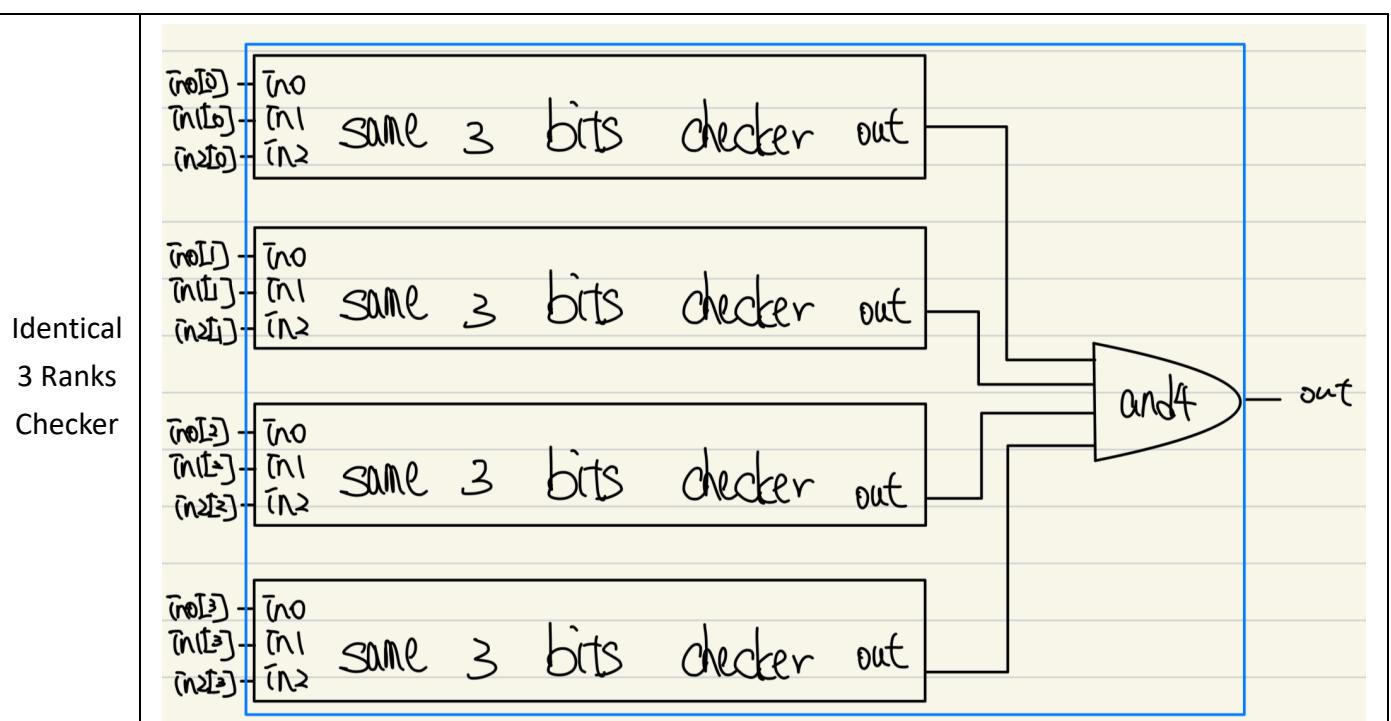
3 of a kind Possible Checker



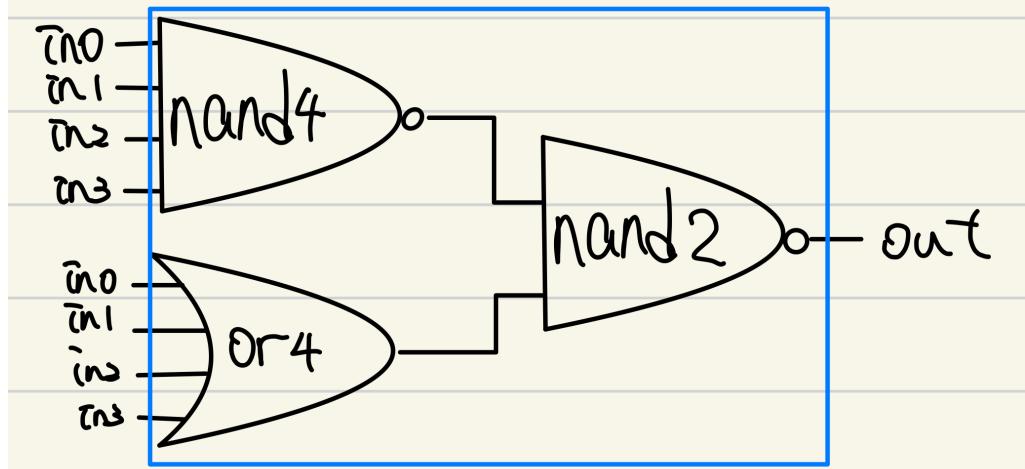
2 Pairs Possible Checker



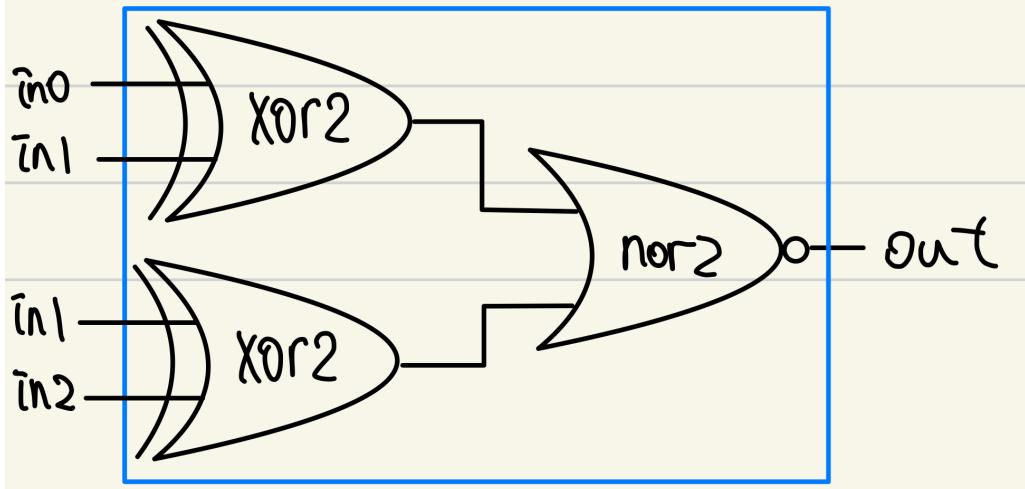
<p>2 Pairs Possible Sub Checker</p>	<p><math>\bar{in}_0[3:0]</math> <math>\bar{in}_0[3:0]</math> identical 2 ranks checker  <math>\bar{in}_1[3:0]</math> <math>\bar{in}_1[3:0]</math></p> <p><math>\bar{in}_2[3:0]</math> <math>\bar{in}_2[3:0]</math> identical 2 ranks checker  <math>\bar{in}_3[3:0]</math> <math>\bar{in}_3[3:0]</math></p> <p><math>\bar{in}_0[3:0]</math> <math>\bar{in}_0[3:0]</math> identical 2 ranks checker  <math>\bar{in}_1[3:0]</math> <math>\bar{in}_1[3:0]</math></p> <p><math>\bar{in}_2[3:0]</math> <math>\bar{in}_2[3:0]</math> identical 2 ranks checker  <math>\bar{in}_3[3:0]</math> <math>\bar{in}_3[3:0]</math></p> <p><math>\bar{in}_0[3:0]</math> <math>\bar{in}_0[3:0]</math> identical 2 ranks checker  <math>\bar{in}_1[3:0]</math> <math>\bar{in}_1[3:0]</math></p>
<p>1 Pair Possible Checker</p>	<p><math>\bar{in}_0[3:0]</math> <math>\bar{in}_0[3:0]</math> identical 2 ranks checker  <math>\bar{in}_1[3:0]</math> <math>\bar{in}_1[3:0]</math></p> <p><math>\bar{in}_0[3:0]</math> <math>\bar{in}_0[3:0]</math> identical 2 ranks checker  <math>\bar{in}_2[3:0]</math> <math>\bar{in}_2[3:0]</math></p> <p><math>\vdots \times 10</math></p> <p><math>\bar{in}_3[3:0]</math> <math>\bar{in}_3[3:0]</math> identical 2 ranks checker  <math>\bar{in}_4[3:0]</math> <math>\bar{in}_4[3:0]</math></p>
<p>Identical 4 Ranks Checker</p>	<p><math>\bar{in}_0[0]</math> <math>\bar{in}_0[0]</math> same 4 bits checker out  <math>\bar{in}_1[0]</math> <math>\bar{in}_1[0]</math>  <math>\bar{in}_2[0]</math> <math>\bar{in}_2[0]</math>  <math>\bar{in}_3[0]</math> <math>\bar{in}_3[0]</math></p> <p><math>\bar{in}_0[1]</math> <math>\bar{in}_0[1]</math> same 4 bits checker out  <math>\bar{in}_1[1]</math> <math>\bar{in}_1[1]</math>  <math>\bar{in}_2[1]</math> <math>\bar{in}_2[1]</math>  <math>\bar{in}_3[1]</math> <math>\bar{in}_3[1]</math></p> <p><math>\bar{in}_0[2]</math> <math>\bar{in}_0[2]</math> same 4 bits checker out  <math>\bar{in}_1[2]</math> <math>\bar{in}_1[2]</math>  <math>\bar{in}_2[2]</math> <math>\bar{in}_2[2]</math>  <math>\bar{in}_3[2]</math> <math>\bar{in}_3[2]</math></p> <p><math>\bar{in}_0[3]</math> <math>\bar{in}_0[3]</math> same 4 bits checker out  <math>\bar{in}_1[3]</math> <math>\bar{in}_1[3]</math>  <math>\bar{in}_2[3]</math> <math>\bar{in}_2[3]</math>  <math>\bar{in}_3[3]</math> <math>\bar{in}_3[3]</math></p>



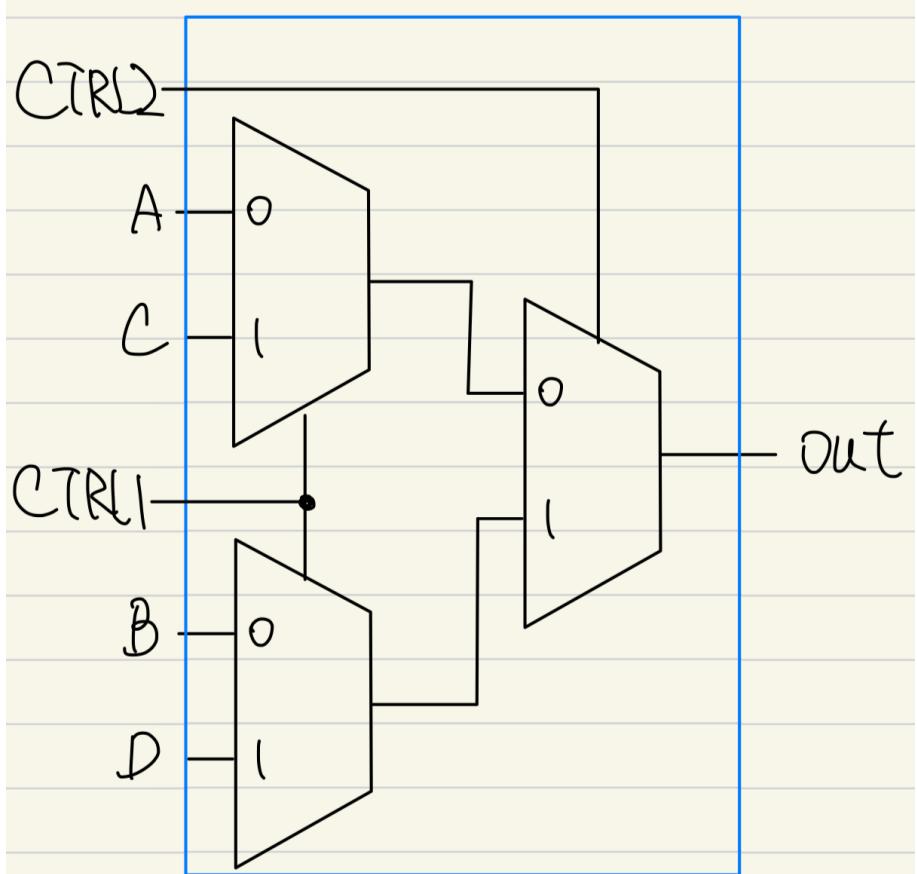
Same 4 Bits Checker



Same 3 Bits Checker



MUX4C



## 5. Self-defined Basic Gates

Name	Boolean Expression with the minimal delay	Delay (ns)
XNOR2	$z = (a \wedge b)'$	$EO + IV = 0.470$
OR3low	$z = [(a+b)'c]'$	$NR2 + ND2 = 0.403$
OR4low	$z = [(a+b)'(c+c)']'$	$NR2 + ND2 = 0.403$
OR5	$z = [(a+b)'(c+d)'e]'$	$NR2 + ND3 = 0.453$
NR5	$z = (a+b)'(c+d)'e'$	$NR2 + ND3 = 0.502$
AN5	$z = [(ab)' + (cd)' + e]'$	$ND2 + NR3 = 0.521$
ND5	$z = [(abc)(de)]'$	$AD3 + ND2 = 0.451$
OR6	$z = [(a+b)'(c+d)'(e+f)]'$	$NR2 + ND3 = 0.453$
NR6	$z = (a+b)'(c+d)'(e+f)'$	$NR2 + ND3 = 0.502$
AN6	$z = [(ab)' + (cd)' + (ef)]'$	$ND2 + NR3 = 0.521$
ND6	$z = [(abc)(def)]'$	$AD3 + ND2 = 0.451$
OR8	$z = [(a+b+c+d)'(e+f+g+h)]'$	$NR4 + ND2 = 0.521$
NR8	$z = (a+b+c+d)'(e+f+g+h)'$	$NR4 + AN2 = 0.570$
AN8	$z = [(abcd)' + (efgh)]'$	$AN4 + NR2 = 0.523$
ND8	$z = [(abc)(def)(gh)]'$	$AN3 + ND3 = 0.501$
AN9	$z = (abc)(def)(ghi)$	$AN3 + AN3 = 0.550$
ND9	$z = [(abc)(def)(ghi)]'$	$AN3 + ND3 = 0.501$
OR10	$z = [(a+b+c+0)'(d+e+f+0)'(g+h+i+j)]'$	$NR4 + ND3 = 0.571$
NR10	$z = (a+b+c+0)'(d+e+f+0)'(g+h+i+j)'$	$NR4 + AN3 = 0.620$
AN10	$z = [(abc)' + (def)' + (gh)' + (ij)]'$	$ND3 + NR4 = 0.571$
ND10	$z = [(abc)(def)(gh)(ij)]'$	$AN3 + ND4 = 0.571$

## 6. Discussion

1. How to classify ?	
Straight flush	Both straight and flush
Straight	Brute force is the fastest solution since there are only 10 cases
Flush	Just check if the first two bits of all 5 cards are the same or not
4 of a kind	Just check if there are 4 cards that are the same rank
Full house	Just check if there are 3 cards with the same rank and 2 cards with the other rank
3 of a kind	Check if there are 3 cards with the same rank while <b>not forming any of the cases above</b>
2 pairs	Check if there are at least 2 pairs while <b>not forming any of the cases above</b>
1 pair	Check if there are at least a pair while <b>not forming any of the cases above</b>
High card	<b>None of the cases above</b>
2. Why not design 3 of a kind checker, 2 pairs checker, and 1 pair checker ?	
Since " <b>not forming any of the cases above</b> " is not easy to design, just design " <b>possible checker</b> " instead and handle the next level classification when dealing with output	
3. How to minimize the delay ?	
There are many aspects and manners that allow us to improve the delay	
Aspects	Explanation
Parallel Computing	Paralleling logic gates while not cascading them
Gates Substitution	Use nand/and instead of nor/or since nand/and are faster
Inverted Outputs	<p>For some modules, I've designed two outputs which are <b>out</b> and <b>out'</b>. We directly handle the inverted output in modules, which can let us avoid adding an inverter after the output and substantially decrease the delay.</p> <p>For example, we only need <b>FH'</b> (full house checker inverted output) while not <b>FH</b>, the only difference of the delay between them is nand10 and and10. From the table of section 5, we found out that their delay are basically the same, which means we can get the <b>out'</b> while not using an additional inverter.</p>
Cascading Muxes	<p>When designing a 4-1 mux, I did consider directly using parallel logic gates to create it. Theoretically, it should perform better than cascading 2-1 muxes, but the result was worse than the cascading case. After analyzing the delay of every signal, I've found out that some signals are slower, causing the parallel logic to wait for them. While by cascading them, we can first cope with the early signals and handle the slower signal later.</p> <p>For example, I've used 4-1 mux on my output handling. Deal with the <b>flush</b> signal on the first stage in the mux and then deal with the <b>straight</b> signal on the second stage because <b>flush</b> is way much faster than <b>straight</b> which is the slowest signal among all of the checkers.</p>
Self-defined basic gates	<p>It is necessary to redesign some basic gates since the provided version are too slow. For instance, the delay of <b>OR4</b> defined in "<b>lib.v</b>" is 0.544, but the improved delay is only 0.403 (<b>OR4low</b>). In practice, we can combine <b>NR2</b> and <b>ND2</b>. Check out more combinations in section 5.</p>

