a)

As long as any input is active, TS_0 will be active.

$$TS_0 = TA_1 + TA_0 + TB_1 + TB_0$$

As long as at least two inputs are active, TS_1 will be active. Alternatively, if three or more inputs are *in*active, TS_1 will be inactive (presented here in two equivalent forms).

$$TS_1 = \overline{(\overline{TA_1} \cdot \overline{TA_0} \cdot \overline{TB_1}) + (\overline{TA_1} \cdot \overline{TA_0} \cdot \overline{TB_0}) + (\overline{TA_1} \cdot \overline{TB_1} \cdot \overline{TB_0}) + (\overline{TA_0} \cdot \overline{TB_1} \cdot \overline{TB_0})}$$

$$TS_1 = (TA_1 + TA_0 + TB_1) \cdot (TA_1 + TA_0 + TB_0) \cdot (TA_1 + TB_1 + TB_0) \cdot (TA_0 + TB_1 + TB_0)$$

At least three inputs must be active for TS_2 to be active:

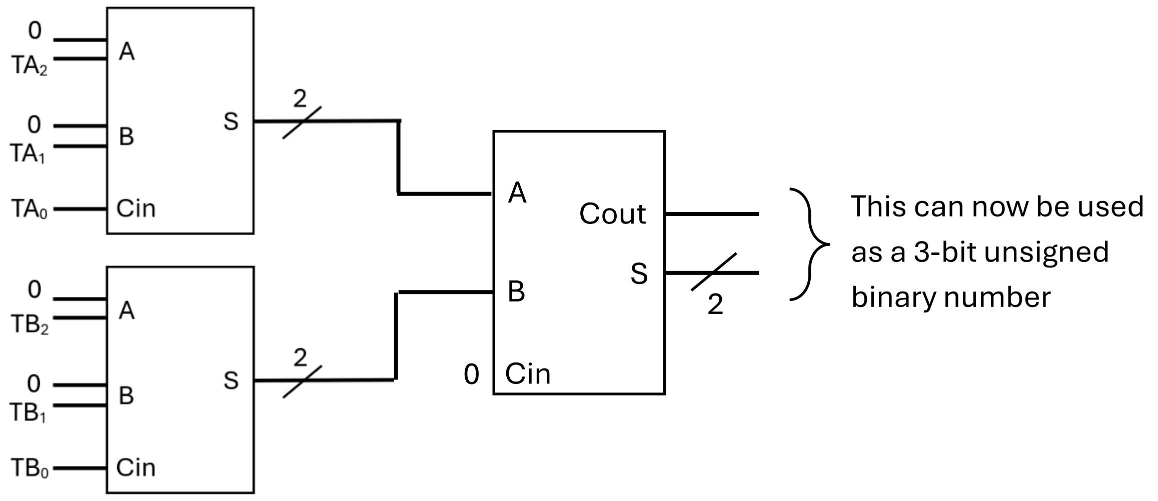
$$TS_2 = TA_1 \cdot TA_0 \cdot TB_1 + TA_1 \cdot TA_0 \cdot TB_0 + TA_1 \cdot TB_1 \cdot TB_0 + TA_0 \cdot TB_1 \cdot TB_0$$

All inputs must be active for TS_3 to be active:

$$TS_3 = TA_1 \cdot TA_0 \cdot TB_1 \cdot TB_0$$

b)

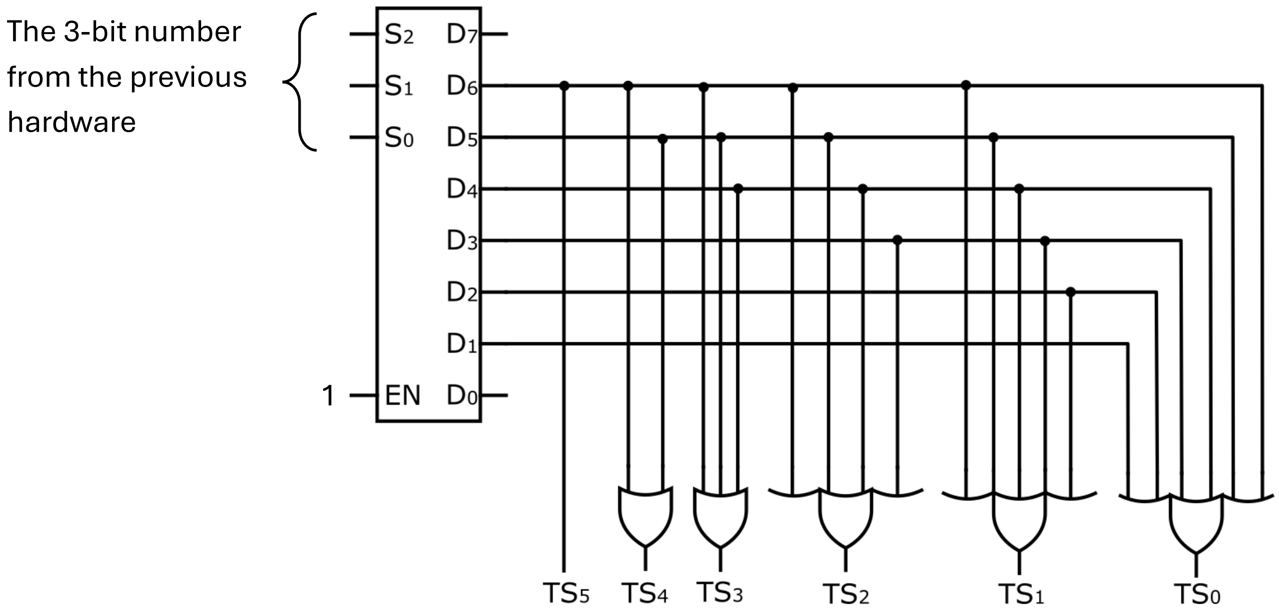
Each bit in the tally number system represents numerical “1” so each bit can be added as the least-significant bit of six *separate* unsigned numbers, using normal binary adders. An efficient way to do that with six inputs would be to make use of the carry in on two 2-bit adders, since the carry in also gets added to the least-significant place, and then add those results with another 2-bit adder and including the carry out as part of the numerical result. Since the maximum expected result will be 6, these adders will always be enough.



Thinking about converting an unsigned number to a right-justified “tally” system,

Number	TS_5	TS_4	TS_3	TS_2	TS_1	TS_0
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	1
3	0	0	0	1	1	1
4	0	0	1	1	1	1
5	0	1	1	1	1	1
6	1	1	1	1	1	1

it looks most similar to a line decoder (highlighted in red), but it will need some massaging, such as:


The 3-bit number from the previous hardware
S_2
S_1
S_0
1
EN
D_7
D_6
D_5
D_4
D_3
D_2
D_1
D_0
TS_5
TS_4
TS_3
TS_2
TS_1
TS_0