Problem 1:

Your task is to design a 256-bit adder and you have three options: ripple carry adder, carry select adder, and carry look-ahead adder.

1. Assuming you picked ripple carry adder, what will be the estimated delay through this adder using the delay per gate as a unit: t_g.
2. Note that the ripple carry adder is the cheapest and most power efficient adder but it is very slow, name an application that can utilize this adder because it is restricted in power and area but not in speed.
3. Assuming you picked a carry select adder, determine the best partitioning of the bits to achieve the fastest response possible and find the estimated delay through this adder using the delay per gate as a unit: t_g.
4. How many extra Full adders and 2-to-1 multiplexers are required to be used in the carry select adder assuming your best partitioning of the bits in 3 as compared to the ripple carry adder.
5. Note that the carry select adder is more expensive and less power efficient than the ripple carry adder but it is for sure faster, name an application that can utilize this adder because it is moderately restricted on power, area, and speed.
6. How fast can a carry look-ahead adder compute this addition, give your answer in t_g.
7. Note that the carry look-ahead adder is the fastest, most expensive, and least power efficient adder among the three, name an application that can utilize this adder because it is only restricted on speed and can afford to use more area and power.

Problem 2:

1. Draw the hardware of a signed 6by6 array multiplier. Hint: you will be using inverters, AND gates, HA, FA, and cheap FA (FA*). Note that the last Full Adder of all the ripple-carry-adders used in this multiplier is cheap because it only needs to generate one output.
2. Draw the hardware inside FA*.
3. What is the total number of inverters, AND gates, HAs, FAs, and FA*s used?
4. What is the estimated delay in t_g through this array multiplier, please show your critical path using a different color on your hardware diagram in 1.

Problem 3:

1. Show the high-level structure of a 5-bit magnitude comparator using alternative design 3, that is, starting from the most significant bit and ending in the least significant bit and propagating an equal and AGB bits between the different slices.

2. Fill the table for the following inputs: (Hint you need to convert A and B to binary first).

A	B	Eq4	Eq3	Eq2	Eq1	Eq0	AGB4	AGB3	AGB2	AGB1	AGB0
26	25	1	1	1	0	1	1	1	0	1	0
24	25	1	1	1	1	0	1	1	0	0	1
25	25	1	1	1	1	1	1	1	0	0	0

26 = 11010
25 = 11001
24 = 11000

3. In your own words, explain how does this magnitude comparator works, specifically what is happening for each bit slice and how are the Eq and AGB signals propagated.
4. Also, explain how would you interpret your final outputs Eq0 and AGB0.

1.1: $tc = 2nt_g = 2(256)t_g = 512t_g$

1.2: An application that can utilize the Ripple Carry Adder would be a very small microcontroller that doesn't have much space to work with, such as a Raspberry Pi Pico.

1.3: The best partitioning of bits for a Carry Select Adder would be to have 16 16-bit RCAs.
The estimated delay for this CSA would be: $tc = 16 \cdot 2t_g + 15 \cdot 2t_g = 62t_g$

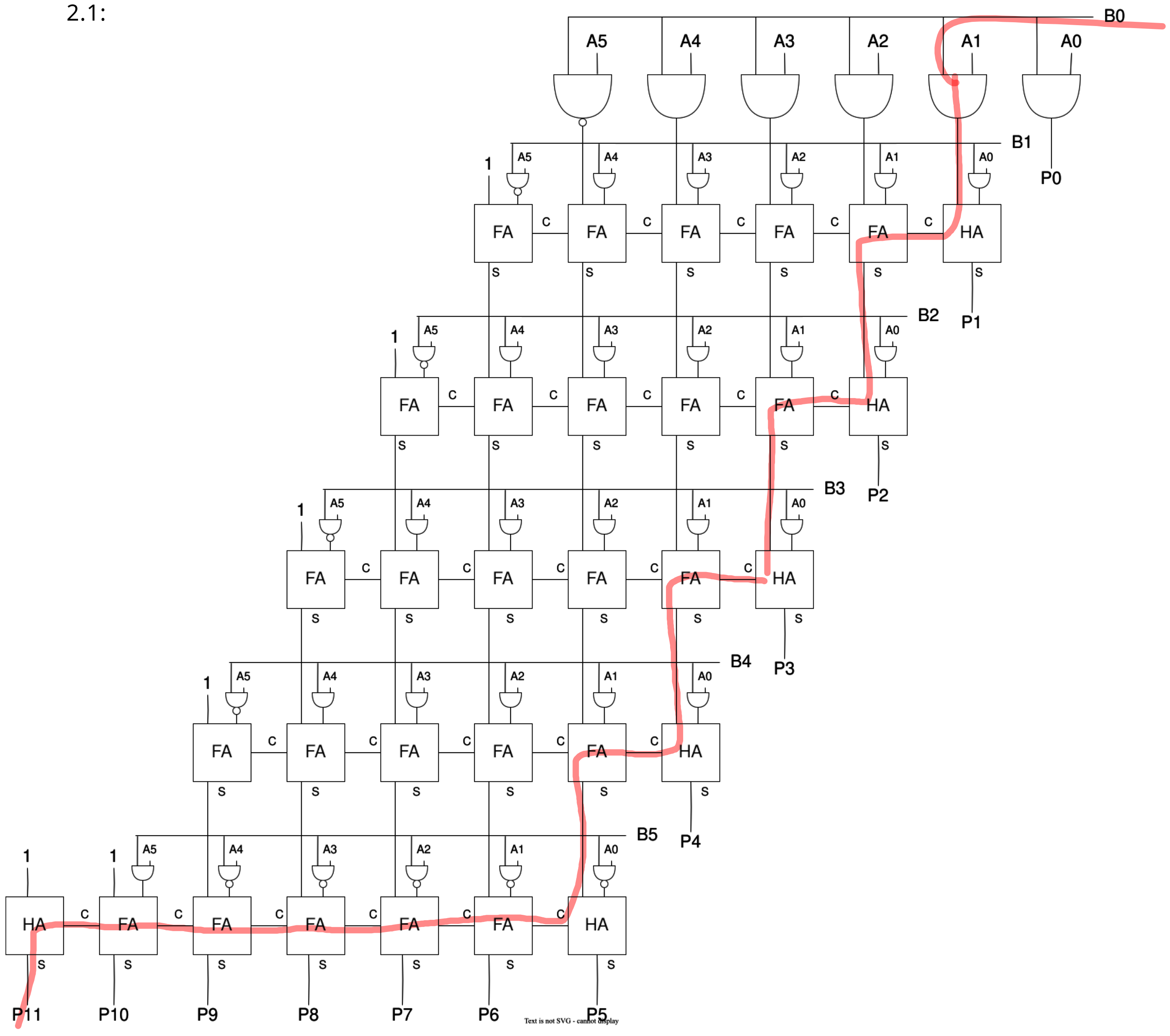
1.4: Since the RCA does not use any multiplexers, we are using 15 2-to-1 multiplexers for the CSA as opposed to 0 for the RCA. For Full Adders, the RCA uses 256 FAs while the CSA uses 16 16-bit RCAs, which also uses 256 FAs, so we are using the same amount of FAs.

1.5: An application of the Carry Select Adder would likely be in every day computers and systems that normal people use. The speed and efficiency required by these systems is moreso than an RCA, while these systems have access to more power and area than RCAs as well.

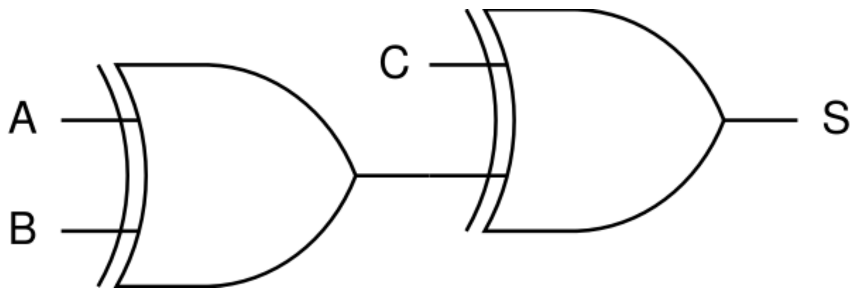
1.6: $$tc = (log_2 n)t_g + 2t_g = (log_2 256)t_g + 2t_g = 8t_g + 2t_g = 10t_g$$

1.7: An application for the Carry Look-Ahead Adder would be in large server databases and networks that require computations to be as fast and efficient as possible. Servers for companies like Microsoft and Google come to mind.

2.1:



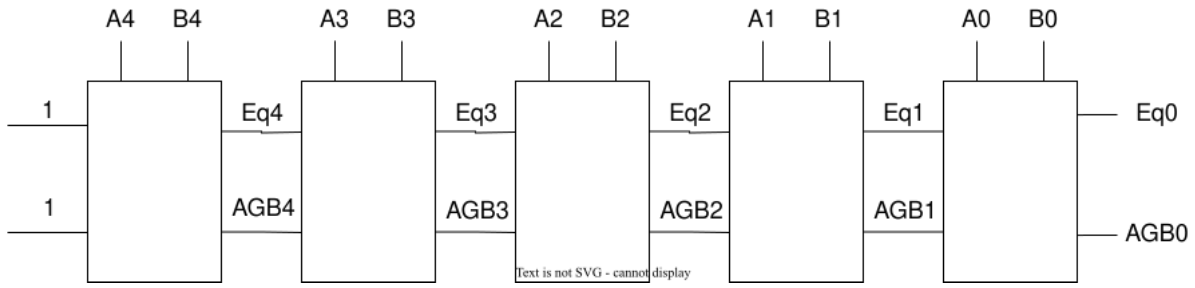
2.2:



2.3: 10 inverters, 36 AND gates, 6 HAs, 21 FAs, and 4 FA*s

2.4: $tc = 6 \cdot 1t_g + 9 * 2t_g + 1t_g = 25t_g$

3.1:



3.3: Starting with the most significant bit, for each bit slice, the bits are first passed into an XNOR gate. Then, the output of that XNOR is passed into an AND gate with the Eq signal from the previous comparator (for the most significant bit, 1). This AND gate gives us our next Eq signal. Then, the AGB bit and A bit are passed into a 2-to-1 multiplexer, with the Eq bit used for the input signal bit of the multiplexer. The output of this multiplexer is outputted as the next AGB bit.

3.4: If Eq0 is 1, A and B are equal and we don't care about AGB0. If Eq0 is 0 and AGB0 is 0, then A is less than B. If Eq0 is 0 and AGB0 is 1, then A is greater than B.