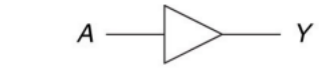
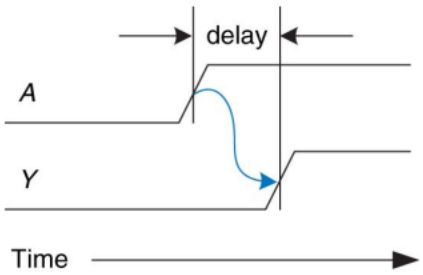
Timing and Advanced Adders

1 Timing

- A buffer doesn't change if an output is high or low, it just makes it either full high or full low

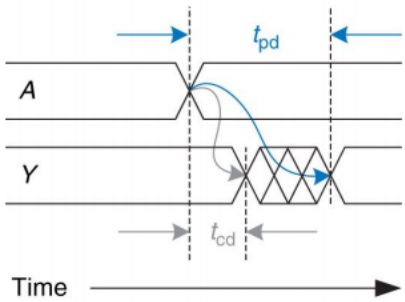
In any physical gate or circuit there is a delay between the input changing and the output adjusting appropriately.





- Note that the lines are not vertical, they rise between low and high

Propagation delay: t_{pd}: The max delay before the output is stable
Comtamination delay t_{cd}: The min delay before the output changes



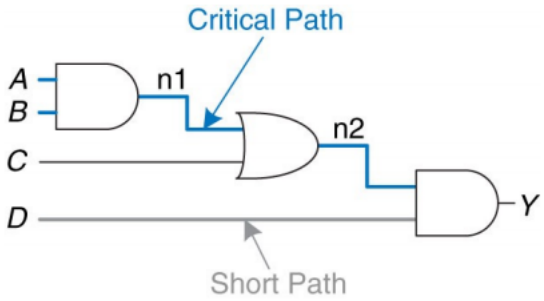
Delay is caused by:

- Capacitance and resistance in a circuit
- Speed of light limitation

Reasons why t_{pd} and t_{cd} may be different:

- Different rising and falling delays
- A circuit may have multiple inputs and outputs, some of which are faster than others
- Circuits slow down when hot and speed up when cold

2 Critical paths



In a circuit the critical path is the path determining the propagation delay of the circuit - i.e. the longest path in the circuit

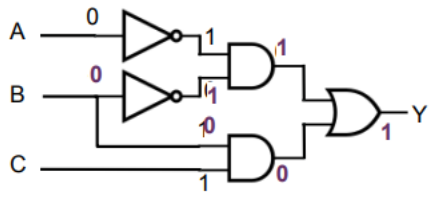
$$t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$$

The short path is the path determining the contamination delay of the circuit - i.e. the shortest path in the circuit

$$t_{cd} = t_{cd\_AND}$$

3 Glitches

Sometimes the output can temporarily move to an incorrect value before stabilising.
This is called a glitch.



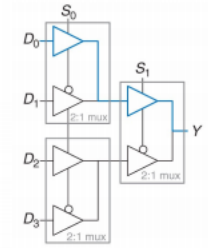
With a=0,b=1 and c=1. If b falls to 0, the output will change from 1 to 0 to 1.

4 Mux delays

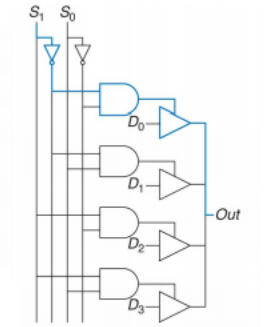
4-line multiplexor circuits.

Different characteristics for selector change and data change

Gate	t_{pd} (ps)
NOT	***30***
AND	60
3-AND	***80***
4-OR	90
Tristate (D)	***50***
Tristate (S)	35



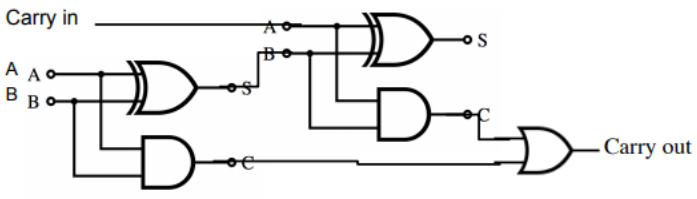
Prefer (b) when data changes last, but prefer (c) when control changes last



- Tristate D - Data
- Tristate S - Selector
- The different speeds for different changes need to be taken into account for whatever the use is

5 Adder

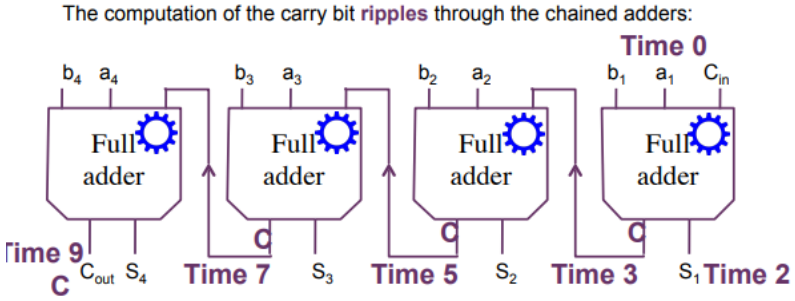
The full adder circuit we have looked at takes 3 gate delays for the carry out to be computed:



If we can pre-compute the first level, there are only 2 gate delays once the carry in arrives.

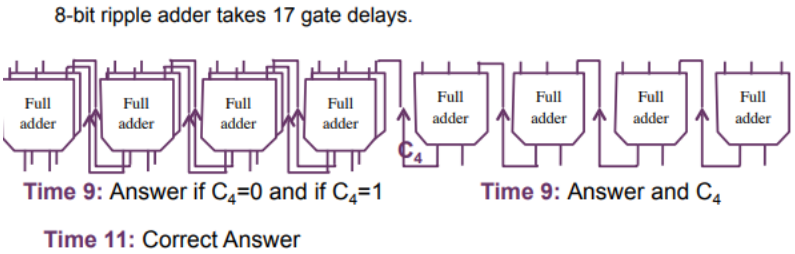
N.B. I am counting all gates as 1 time step. In reality some take longer than others.

6 Ripple adder



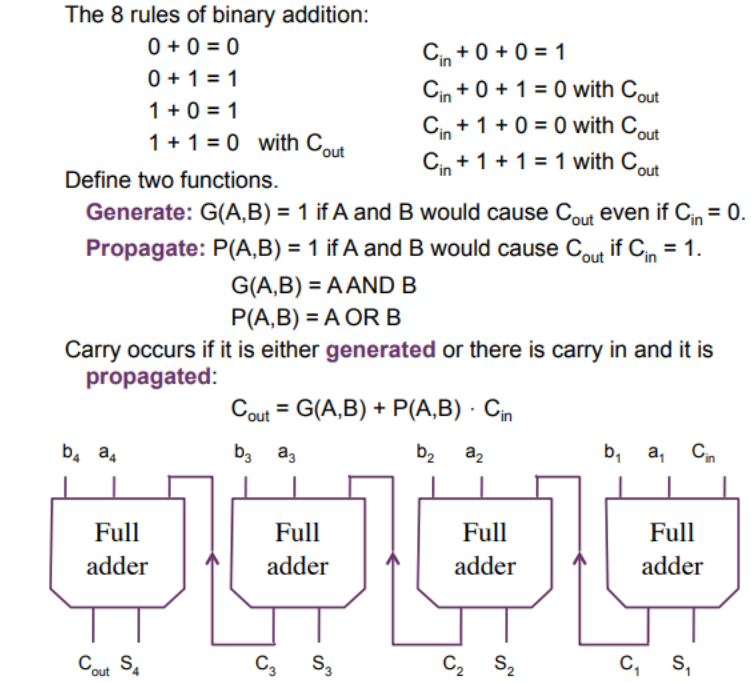
- 3 time steps for 1st carry, then 2 time steps for each after that as the first level is pre computed

7 Faster Adders - idea



- This uses c_4 for the multiplexor and makes large additions much faster, at the expense of more circuitry

8 Carry-Lookahead Adder

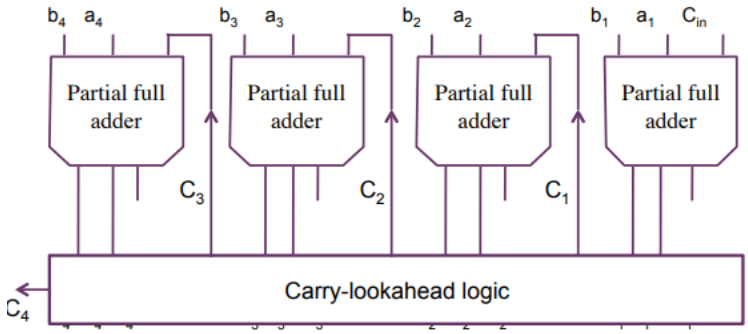


$C_{out} = G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1 \cdot C_{in}$

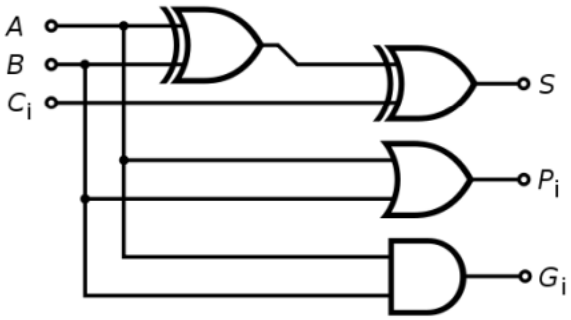
We can compute every P_i and G_i in one gate delay

So we can compute C_{out} (and all the intermediate carries) in 3 gate delays (1 for P_i,G_i, an AND layer and an OR layer).

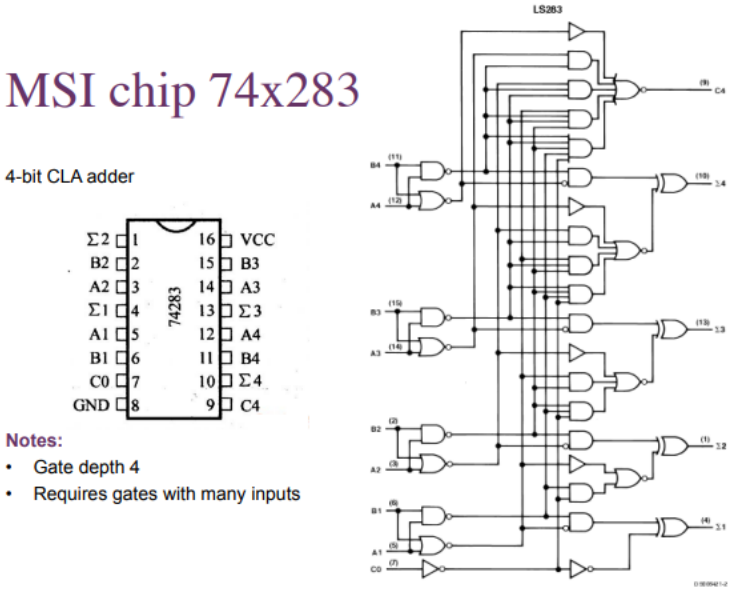
We could compute the full sum in 4 gate delays!



9 Partial Full Adder



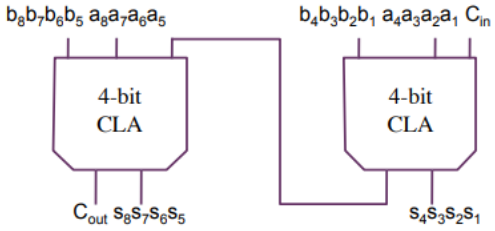
10 MSI Chip



11 More on the carry-lookahead adder

We could create a CLA which adds n-bit numbers in constant gate delay.
It would require order n^2 gates and gates taking order n inputs.
This is impractical for large n.

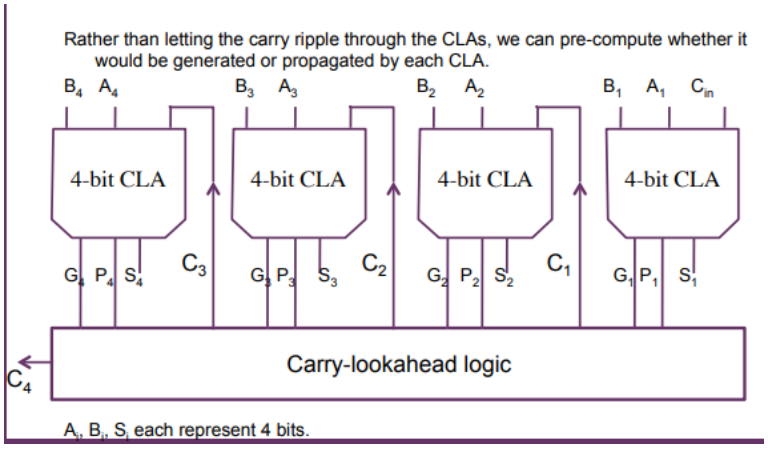
First solution: chain 4-bit CLAs



The carry now ripples along the chain 4-times as fast as originally

12 2 Level Carry-Lookahead Adder

Rather than letting the carry ripple through the CLAs, we can pre-compute whether it would be generated or propagated by each CLA.



We now have a 16-bit 2-level CLA.

Gate delays:

- 1 gate delay to compute first level Gs, Ps
- 2 gate delays to compute second level Gs, Ps
- 2 gate delays to compute carries
- 1 gate delay to compute sums

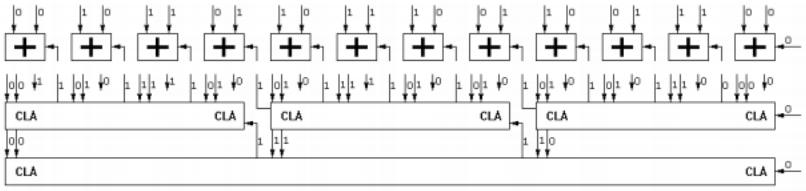
Total **6 gate delays**.
A standard 16-bit ripple carry adder would take 33 gate delays.

These 16-bit adders could be chained to give an n-bit adder with gate delay 2(n/16)+4.
Alternatively, a 3 (or more) level CLA could be used to create an n-bit adder with delay O(log n) and not too much circuitry.

13 Example

```
A       0 1 1 0 1 1 1 0 1 0 1 0        : 1770
B     + 0 0 1 1 0 1 0 1 0 1 1 0        : 854
Sum     1 0 1 0 0 1 0 0 0 0 0 0        : 2624
```

Input bits (A,B) Output bits (G,P,S)



14 Summary

- t_{pd} - propagation delay (time until last answer comes in)
- t_{cd} - contamination delay (time before any change)
- t_{pcq} - max time until recorded after tick
- t_{ccq} - How long before the value is recorded after the tick
- t_{setup} - time stable at least this long before tick
- t_{hold} - time stable at least this long after tick

15 Improving clock speed

- Putting registers in allows you to reduce the long delays, allowing a faster clock speed
- Work like a production line by separating the jobs
- Allows 1st half to work on next calculation while 2nd half is working on the end of the calculation
- First half takes less time than all of previous one, allowing for a faster clock speed
- So overall a calculation takes 2 clock ticks, so the latency will be larger, but the time for a large number of calculations will reduce drastically
- If you keep adding registers it keeps getting better, but the resources required mean that you get diminishing returns