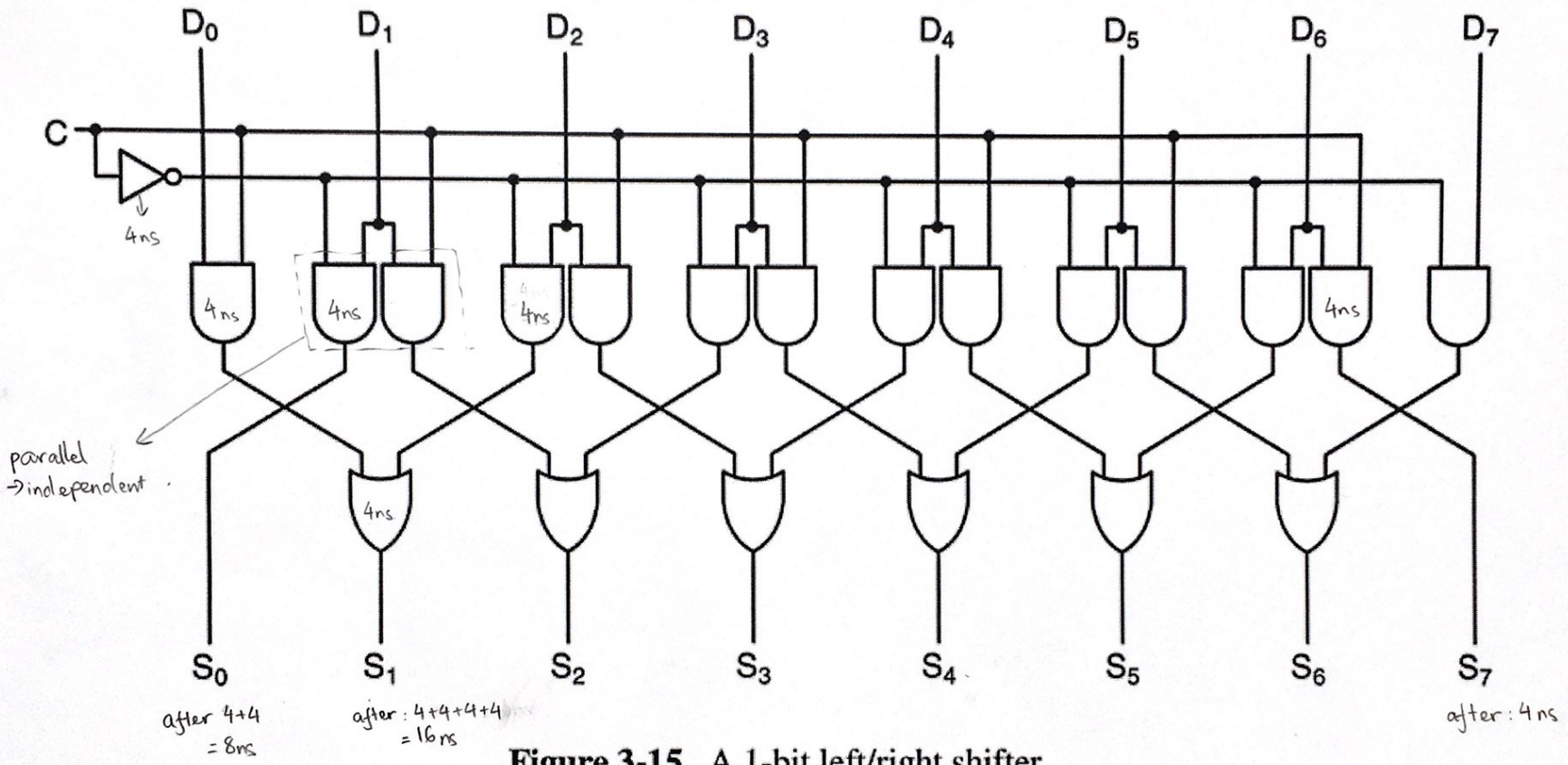


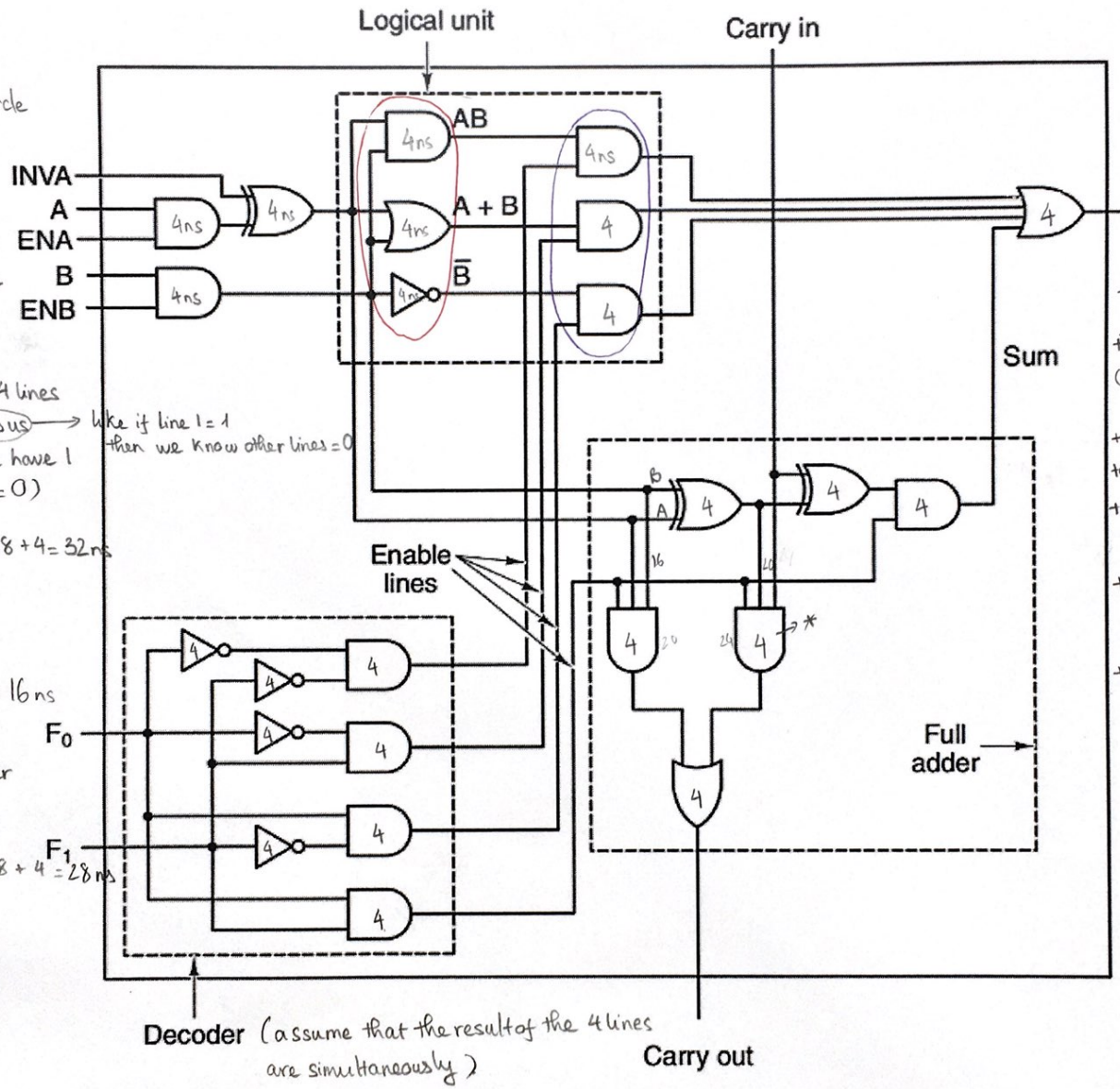
Figure 3-15. A 1-bit left/right shifter.

From $S_1 \rightarrow S_6$: since the number of gates involve to produce the output is the same
→ delay = S_1 = 16ns (1 inverter, 2 and gates, 1 or gate)

For S_0 : delay = 4+4 = 8 ns (1 inverter, 1 and gate)

For S_7 : delay = 4ns (1 and gate)

} → Stable output after : 16 ns



→ For AB case

+ Time to get to the red circle
  4+4+4+4 = 16 ns

+ Time to wait for decoder
  4+4+4 = 12 ns

→ Time to get to the purple circle
  16 + 12 = 28 ns

Assume that results of 4 lines of decoders are simultaneous → like if line 1 = 1 then we know other lines = 0
(and that if the AND gate have 1 input = 0 then the output = 0)

→ Stable output after: 28 + 4 = 32 ns

→ For A+B and $\overline{B}$ case

+ Time to get to red circle = 16 ns
(see above)

+ Time to wait for decoder
  4+4 = 8 (ns)

→ Stable output after - 16 + 8 + 4 = 28 ns

→ For calculating sum

+ Time to wait for A & B input (the INVA, ENBA, ENB part)
  4+4+4 = 12 ns

+ Time that the final AND gate have to wait for decoder: 4 ns

+ Time to have the sum:
  12 + 4 + 4 + 4 + 4 = 28 ns

+ Stable output after: 28+4
  = 32 ns

+ For carry out

> The AND gate that marked w/ * takes longer wait time

> Time to wait for XOR output:
  12 + 4 = 16 ns

> Time to wait for decoder
Assume that it doesn't run in parallel
  4 ns

→ Time till stable carry out:
  16 + 4 + 4 + 4 = 28 ns
  (AND, OR)

Figure 3-18. A 1-bit ALU.

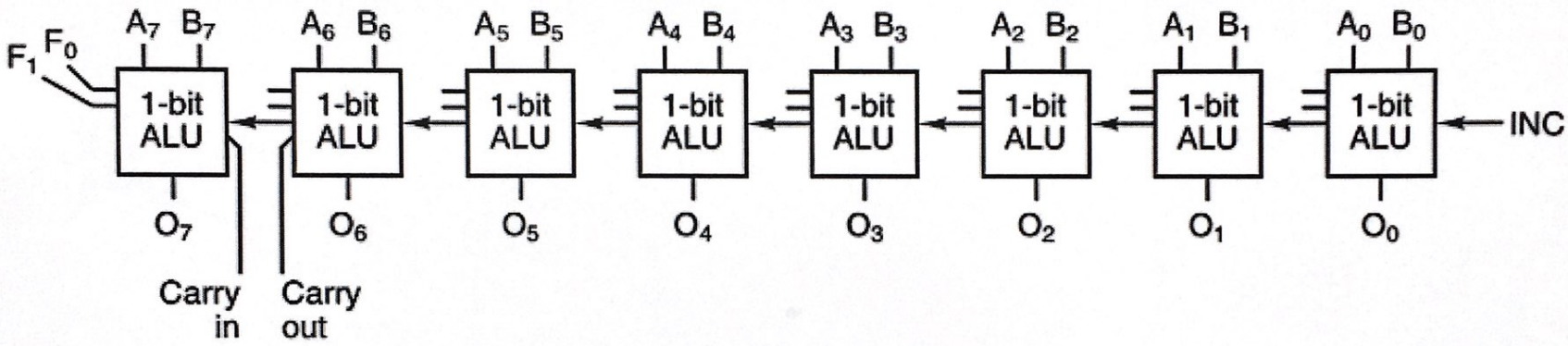


Figure 3-19. Eight 1-bit ALU slices connected to make an 8-bit ALU. The enables and invert signals are not shown for simplicity.

+ For logic calculation (AB, $A+B$, $\overline{B}$): stable output after pd × 8 = $32 \times 8 = 256$ ns / $28 \times 8 = 224$ ns.

+ For arithmetic addition: for the 1st A_0B_0 bit: 32 ns

for bit from $A_1B_1 \rightarrow A_7B_7$: $32 + 28 = 60$ ns

↓ way for carry in (assume calculation of carry in and sum are not parallel)

→ Stable output after: $60 \times 7 + 32 = 452$ ns