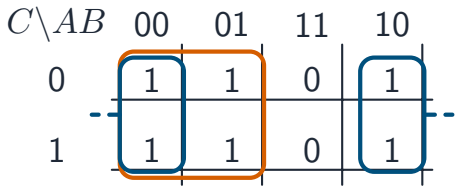
Exercise walk-through

3.1 Number representations and codes

University Electrical Engineering

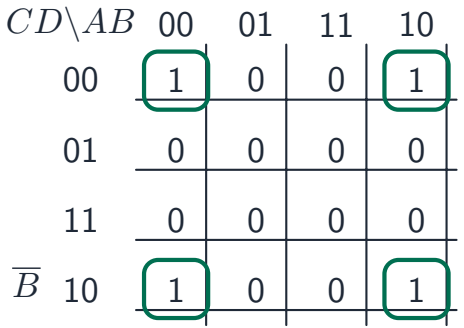
Loop the ones

three variables: the worked example



red loop of 4 (columns 00, 01): $A = 0$ throughout, so $\overline{A}$
blue loop of 4 wrapping round (columns 00, 10): $B = 0$, so $\overline{B}$
$Z = \overline{A} + \overline{B}$

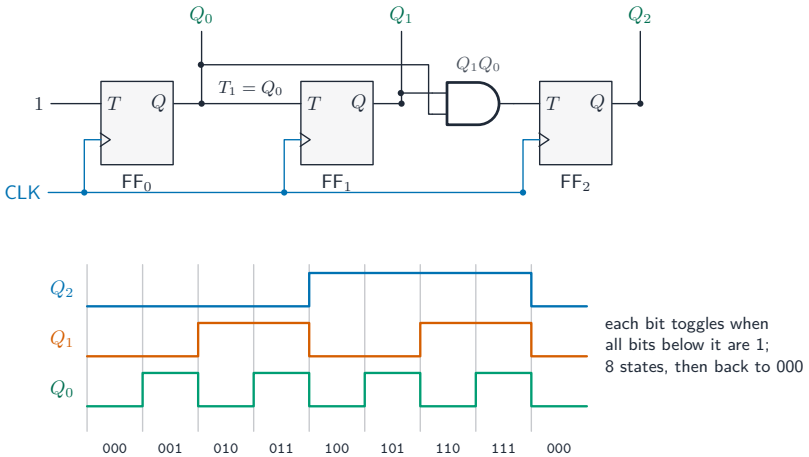
four variables: a wrap-around loop



the four corners are one loop of 4: $B = 0$ and $D = 0$
in every corner, so the term is $\overline{B}\,\overline{D}$

> Bigger loops give simpler expressions

One clock, computed toggles



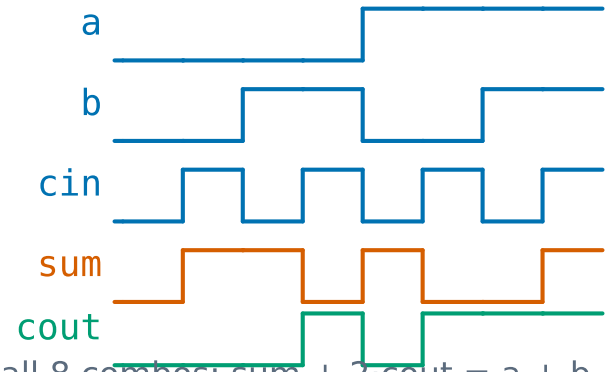
Toggle logic feeds a T flip-flop bank: T0 = 1, T1 = Q0, T2 = Q1 AND Q0 - trace 000, 001, 010, 011.

Description and proof together

One module, one exhaustive testbench

```
module full_adder(
  input a, b, cin,
  output sum, cout);
  assign sum =
      a ^ b ^ cin;
  assign cout =
      (a & b) |
      (cin & (a^b));
endmodule
```

synthesizable module

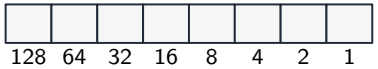


all 8 combos: sum + 2·cout = a + b + cin ✓

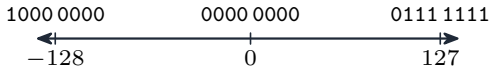
Left: the full_adder module. Right: the **testbench** waveform driving all 8 combinations with sum and cout tracking a + b + cin.

Exercises 2.1·2.2

2.1 Convert $B7_{16}$ to binary and to decimal. Then convert 200 to hexadecimal. [3]

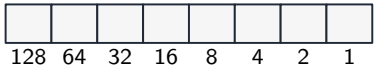


2.2 Write -100 as an 8-bit two's complement number. Then find the signed value of $1110\,0110_2$. [3]



Exercises 2.1·2.2

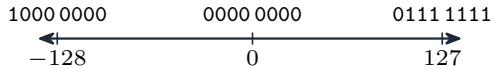
2.1 Convert $B7_{16}$ to binary and to decimal. Then convert 200 to hexadecimal. [3]



2.1

$B7_{16} = 1011\,0111_2$; $= 128 + 32 + 16 + 4 + 2 + 1 = \mathbf{183}$; $200 = 12 \times 16 + 8 = \mathbf{C8_{16}}$

2.2 Write -100 as an 8-bit two's complement number. Then find the signed value of $1110\,0110_2$. [3]

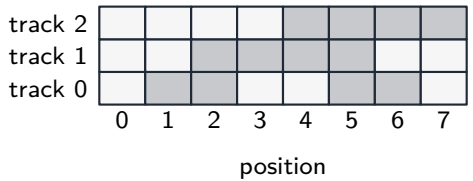


2.2

$100 = 0110\,0100$; invert $1001\,1011$, add 1; $-100 = \mathbf{1001\,1100_2} = 9C_{16}$; $1110\,0110 = -128 + 64 + 32 + 4 + 2 = \mathbf{-26}$

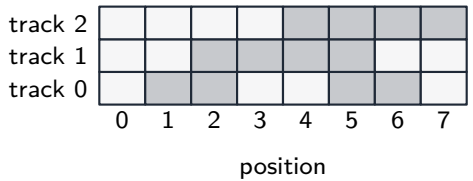
Exercise 2.3

2.3 A rotary encoder reports the shaft position in Gray code. Convert the binary number 1011 to Gray code, and the Gray code 1101 back to binary. Why does an encoder use Gray code rather than plain binary? [3]



Exercise 2.3

2.3 A rotary encoder reports the shaft position in Gray code. Convert the binary number 1011 to Gray code, and the Gray code 1101 back to binary. Why does an encoder use Gray code rather than plain binary? [3]

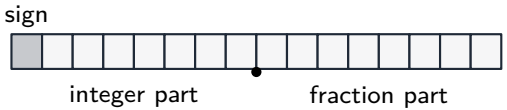


2.3

$g_3 = 1$, $g_2 = 1 \oplus 0 = 1$, $g_1 = 0 \oplus 1 = 1$, $g_0 = 1 \oplus 1 = 0$: **1110**; $b_3 = 1$, $b_2 = 1 \oplus 1 = 0$, $b_1 = 0 \oplus 0 = 0$, $b_0 = 0 \oplus 1 = 1$: **1001**; only one bit changes between neighbouring positions, so a sensor that switches a little early or late can never give a code far from the true position

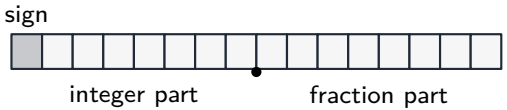
Exercise 2.4

2.4 A controller stores a signed value in a 16-bit word with 8 fraction bits (format Q8.8: 8 integer bits including the sign, 8 fraction bits). Find the resolution and the range. Write -3.75 in this format, in hex. [3]



Exercise 2.4

2.4 A controller stores a signed value in a 16-bit word with 8 fraction bits (format Q8.8: 8 integer bits including the sign, 8 fraction bits). Find the resolution and the range. Write -3.75 in this format, in hex. [3]



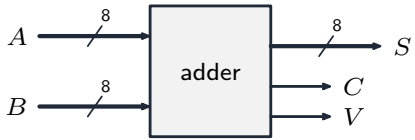
2.4

resolution $2^{-8} = \mathbf{0.0039}$; range $\mathbf{-128}$ to $\mathbf{+127.996}$; $-3.75 \times 256 = -960$; $65536 - 960 = 64576 = \mathbf{FC40_{16}}$

Exercise 3.1

3.1 An 8-bit microcontroller adds two signed sensor readings, $A = 64_{16}$ and $B = 50_{16}$.

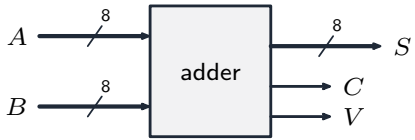
Find the sum in hex and its signed meaning. Say whether overflow happened and give the carry-out. A saturating adder clamps the result to the largest value it can store. What would it give? [5]



Exercise 3.1

3.1 An 8-bit microcontroller adds two signed sensor readings, $A = 64_{16}$ and $B = 50_{16}$.

Find the sum in hex and its signed meaning. Say whether overflow happened and give the carry-out. A saturating adder clamps the result to the largest value it can store. What would it give? [5]



3.1

$64_{16} + 50_{16} = \mathbf{B4_{16}}$; read as signed: $-128 + 52 = \mathbf{-76}$

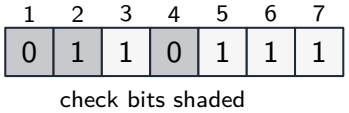
two positives gave a negative: **overflow**, $V = 1$; $100 + 80 = 180 < 256$, so the carry-out is $C = \mathbf{0}$

saturated result: $\mathbf{7F_{16}} = \mathbf{127}$

Exercise 3.2

3.2 A memory protects each 4-bit value with a Hamming $(7, 4)$ code. The check bits are at positions 1, 2 and 4, with even parity. The word 0 1 1 0 1 1 1 is read (positions 1 to 7, left to right).

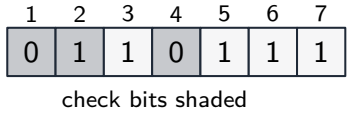
Do the three checks. Find the position of the error, the corrected word and the 4 data bits. [5]



Exercise 3.2

3.2 A memory protects each 4-bit value with a Hamming $(7, 4)$ code. The check bits are at positions 1, 2 and 4, with even parity. The word 0 1 1 0 1 1 1 is read (positions 1 to 7, left to right).

Do the three checks. Find the position of the error, the corrected word and the 4 data bits. [5]



3.2

check 1 covers $1, 3, 5, 7$; check 2 covers $2, 3, 6, 7$; check 4 covers $4, 5, 6, 7$

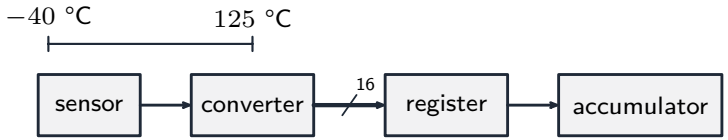
check 1: $0+1+1+1$ odd, fails; check 2: $1+1+1+1$ even, passes; check 4: $0+1+1+1$ odd, fails; error at $1+4=$ position **5**

corrected word **0110011**; data bits (positions $3, 5, 6, 7$) **1011**

Exercise 3.3 (a)

3.3 A temperature logger must cover -40 °C to $+125$ °C.

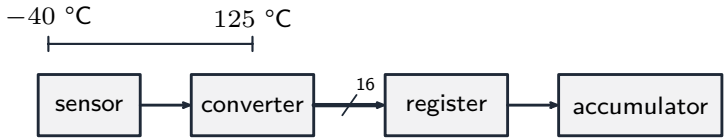
(a) Readings are stored as signed integers that count steps of 0.05 °C. Find the smallest two's complement word width that covers the range. [3]



Exercise 3.3 (a)

3.3 A temperature logger must cover −40 °C to +125 °C.

(a) Readings are stored as signed integers that count steps of 0.05 °C. Find the smallest two's complement word width that covers the range. [3]



(a) $-40/0.05 = -800$ and $125/0.05 = 2500$; 11 magnitude bits reach only 2047, 12 reach 4095; 1 sign bit $+12 =$ **13 bits**

Exercise 3.3 (b)

3.3 A temperature logger must cover -40 °C to $+125$ °C.

(b) The design team prefers a 16-bit word with f fraction bits. Choose f, giving its resolution. Write -40 °C in hex. [3]

Exercise 3.3 (b)

3.3 A temperature logger must cover -40 °C to $+125$ °C.

(b) The design team prefers a 16-bit word with f fraction bits. Choose f, giving its resolution. Write -40 °C in hex. [3]

(b) 125 needs 8 integer bits with the sign (-128 to 127), so $f = 16 - 8 = \mathbf{8}$; resolution $2^{-8} = 0.0039$ °C, finer than 0.05 °C; $-40 \times 256 = -10240$, $65536 - 10240 = \mathbf{D800_{16}}$

Exercise 3.3 (c)

3.3 A temperature logger must cover −40 °C to +125 °C.

(c) The logger adds 64 readings in an accumulator before it divides to find the mean. What width must the accumulator have? Recommend a width, with a reason. [2]

Exercise 3.3 (c)

3.3 A temperature logger must cover -40 °C to $+125$ °C.

(c) The logger adds 64 readings in an accumulator before it divides to find the mean. What width must the accumulator have? Recommend a width, with a reason. [2]

(c) $64 = 2^6$, so the sum needs 6 more bits: $16 + 6 =$ **22 bits**; use a 32-bit accumulator and divide by shifting right 6 places; a 16-bit sum would overflow after only two readings near 125 °C ($2 \times 32000 > 32767$)