Solutions - Homework 1

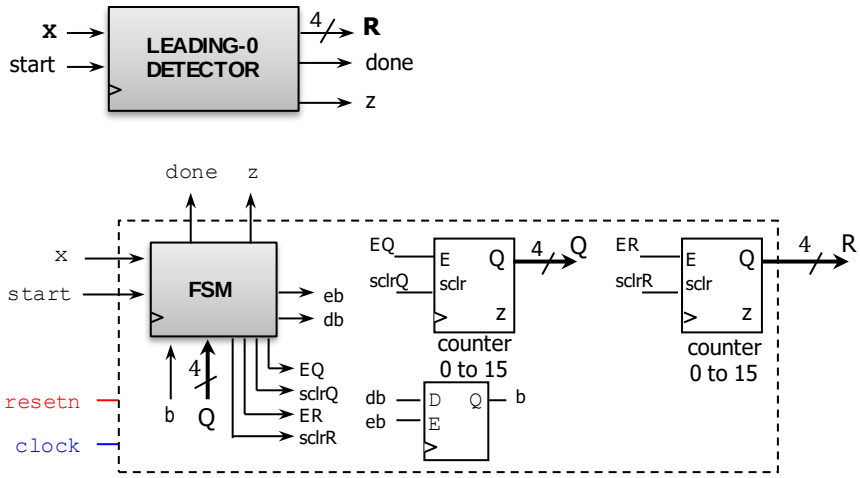
(Due date: January 31st @ 11:59 pm)

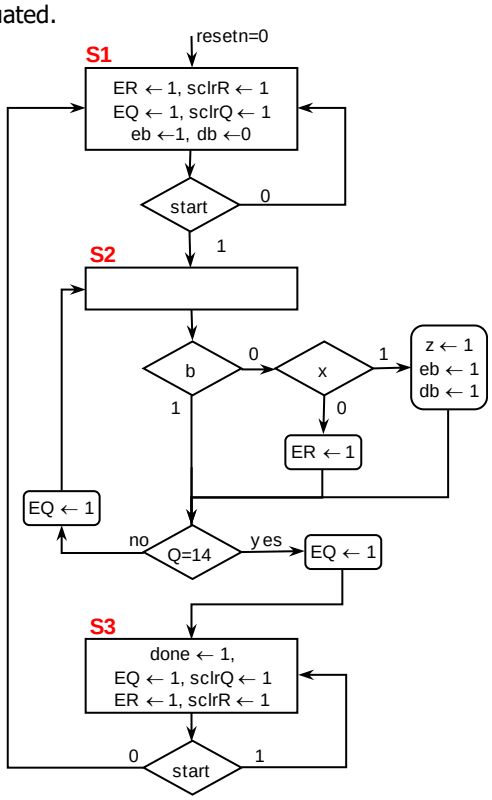
Presentation and clarity are very important! Show your procedure!

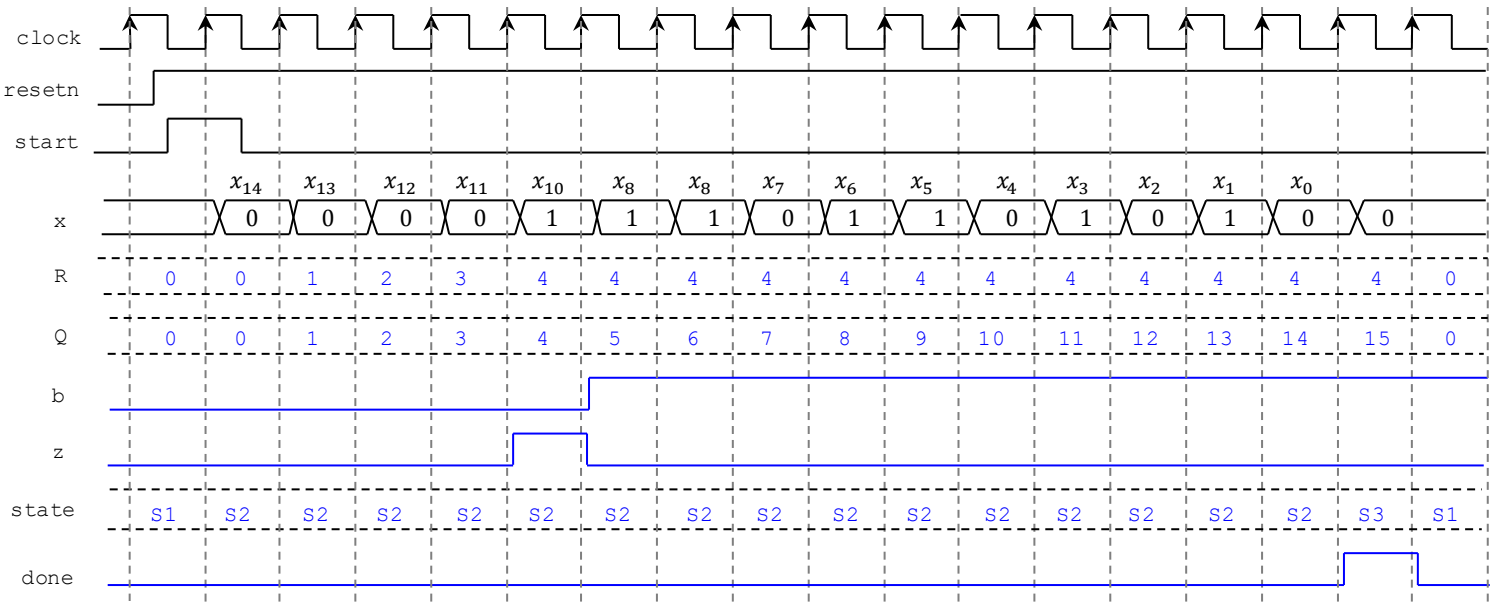
PROBLEM 1 (10 PTS)

- Leading Zero Detector: This iterative circuit processes a 15-bit input (MSB first) and generates the number of leading 0's. before the first 1. Example:
  - ✓ If the sequence is: 0000 0000 0111 010 → R = 9
  - ✓ If the sequence is: 0001 0001 0011 010 → R = 3

- The figure depicts the (in ASM form) and a datapath circuit. Note: Counters. If E=sclr=1, → Q=0.
  Input data: x (entered sequentially, MSB first). Output data: R.
  - ✓ Complete the timing diagram of the digital circuit where one sequence is evaluated.







PROBLEM 2 (45 PTS)

- **Non-restoring square root (iterative implementation)**: Given an unsigned integer D, this circuit computes the square root Q and remainder R ($R = D - Q \times Q$). It is based on the following algorithm:

Radical ($2n$ bits): $D = d_{2n-1}d_{2n-2}d_{2n-3}d_{2n-4} \dots d_1 d_0$
Square Root (n bits): $Q = q_{n-1}q_{n-2} \dots q_0$

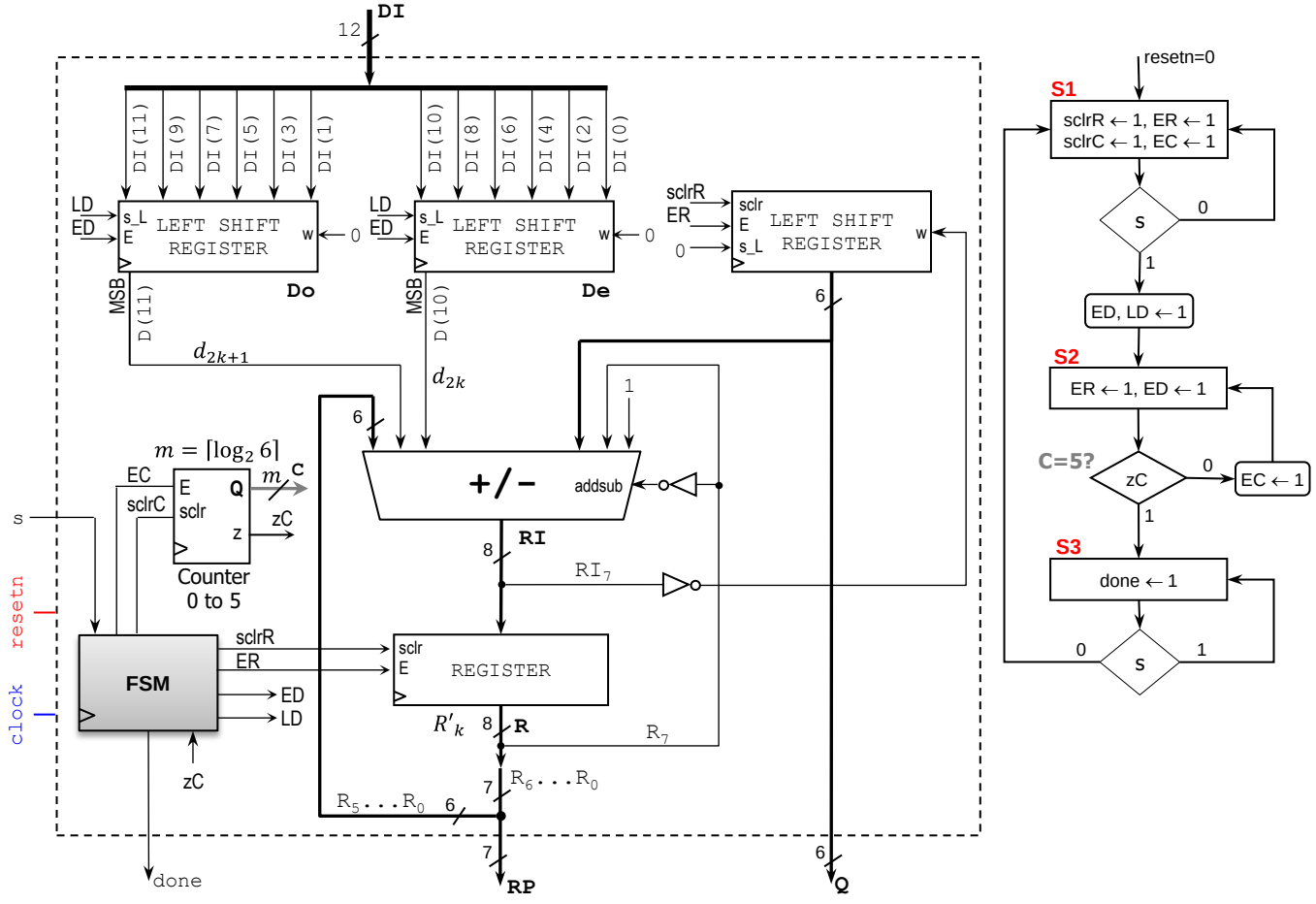
We define: $Q_k = q_{n-1}q_{n-2} \dots q_k, \quad k = 0,1,\dots,n-1$
$R'_k = r'_n r'_{n-1} r'_{n-2} \dots r'_k, \quad k = 0,1,\dots,n-1$

Q_k: unsigned integer with $n-k$ bits.
R'_k: signed (2C) integer with $n-k+1$ bits.

for $k = n\text{-}1 \rightarrow 0$
  if $k = n\text{-}1$ *then*
    $R'_k = d_{2k+1}d_{2k} - 01 \qquad (R'_{n-1} = d_{2n-1}d_{2n-2} - 01)$
  else
    $R'_k = \begin{cases} R'_{k+1}d_{2k+1}d_{2k} - Q_{k+1}01, & if\ q_{k+1} = 1 \\ R'_{k+1}d_{2k+1}d_{2k} + Q_{k+1}11, & if\ q_{k+1} = 0 \end{cases}$
  end
  $q_k = \begin{cases} 1, if\ R'_k \geq 0 \\ 0, if\ R'_k < 0 \end{cases}$
end

- Q and R'_{n-1} are initialized with 0's.
- This is a non-restoring algorithm, meaning that at the last iteration we might not have the correct remainder R. To get the correct value of R, an extra operation would be required.
  Remainder: $R = R_0 = \begin{cases} R'_0, & if\ R'_0 \geq 0 \\ R'_0 + Q_0 1, & if\ R'_0 < 0 \end{cases}$
- In practice, the remainder R is rarely used, and we do not implement that extra operation here.

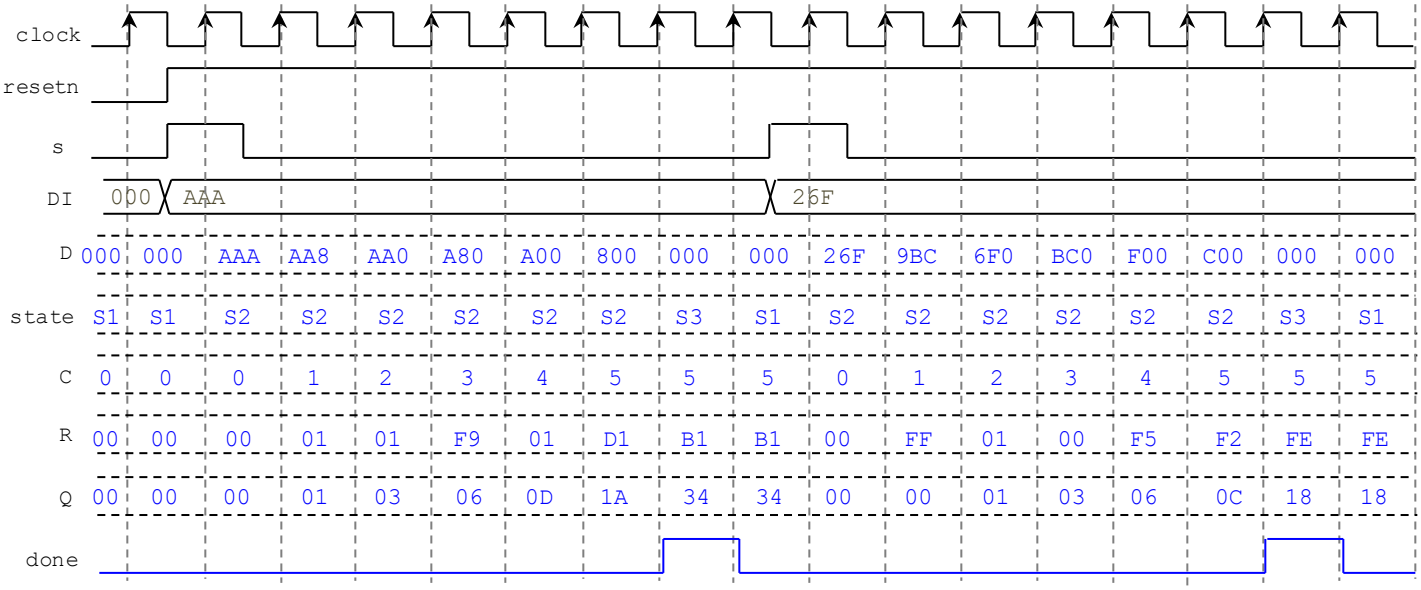
- Iterative architecture ($n = 6$): The figure depicts the FSM (in ASM form) and a Datapath circuit for $n = 6$:
- Input data: **DI**. Output Data: **Q** (6-bit unsigned square root), **RP**.
  - ✓ A square root operation starts when $s = 1$ (where the 12-bit input **DI** is captured). The signal done is asserted to indicate that the operation has been completed and the result appears in register **Q** (and R'_0 appears in **RP**)
  - ✓ The input data **DI** captured into shift register **D**. The bits $d_{2k+1}d_{2k}$ correspond to the 2 MSBs of the register **D**. At every iteration, the register **D** shifts two bits to the left.
    - Register **D**: implemented by two parallel access shift registers: **Do** (for odd bits of **D**) and **De** (for even bits of **D**).
  - ✓ The register **R** stores the 'estimated remainder' R'_k at each iteration.
  - ✓ Note that on **RP**, we get R'_0 at the end of the computations. This is a 7-bit signed number. If it is positive, it is the correct remainder. If it is negative, an extra operation is required to get the correct remainder (not implemented in this circuit).



- **Procedure:**

  - ✓ Write a structural VHDL code. You MUST create (or re-use) a file for the counter, parallel access shift register with sclr, register, adder/subtractor, FSM, and top file.
    - Suggestion: Use parametric code (set up the proper parameters with *generic map*) for these components:
      - Parallel access shift registers with sclr: my_pashiftreg_sclr
      - Counter: my_genpulse_sclr (include in the top file: `use ieee.math_real.log2; use ieee.math_real.ceil;`)
      - Register with enable: my_rege
      - Adder/subtractor: my_addsub (addsub=0: add, addsub=1: subtract)

  - ✓ Write a testbench according to the timing diagram shown (use a 100 MHz input clock). Perform Behavioral Simulation and complete the timing diagram based on the results from the Vivado Simulation window.
    - DI, D, Q and R are specified as hexadecimals.
    - Note that RP = R[6..0]. Also: D = |Do[5]|De[5]|Do[4]|De[4]|Do[3]|De[3]|Do[2]|De[2]|Do[1]|De[1]|Do[0]|De[0]|



    - For completeness, the table shows different results (including cases where **RP** does not have the correct remainder). The first two entries correspond to the cases in the testbench.

DI	**Q**	**RP**	Comments
101010101010 (2730)	110100 (52)	0110001 (0x31)	Remainder ✓
001001101111 (623)	011000 (24)	1111110 (0x7E)	Correct remainder: 1111110+0110001 = AF = **2F**
111110000010 (3970)	111111 (63)	0000001 (0x01)	Remainder ✓
000100100001 (289)	010001 (17)	0000000 (0x00)	Remainder ✓
010110111110 (1470)	100110 (38)	1001101 (0x4D)	Correct remainder: 1001101+1001101 = 9A = **1A**
100010100001 (2209)	101111 (47)	0000000 (0x00)	Remainder ✓

      - Note that **RP** is a 7-bit signed number. If it is positive, **RP** has the correct remainder. If it is negative, the correct remainder is an unsigned 7-bit number, that can be obtained by taking the 7 LSBs out of the operation $R'_0 + Q_0 1$.

    - Verify that the simulation is correct by comparing Q and RP in your simulation window with those in the table.

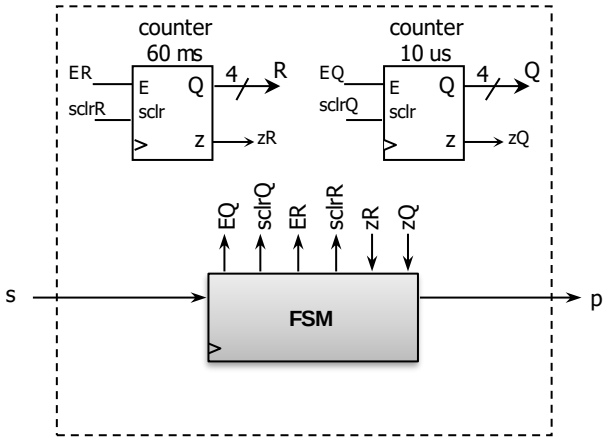
  - ✓ Upload (as a .zip file) the following files to Moodle (an assignment will be created):
    - VHDL code files
    - VHDL testbench

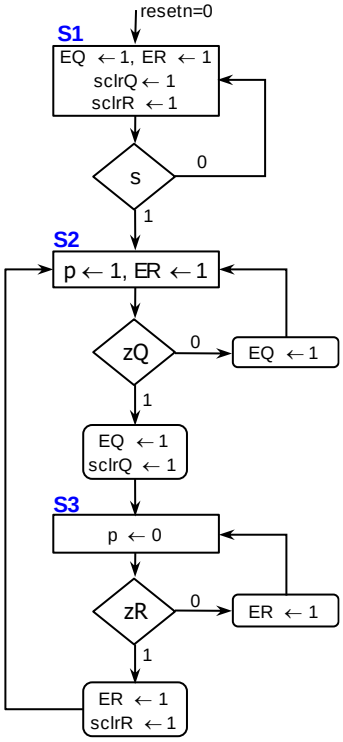
PROBLEM 3 (20 PTS)

- **Periodic Pulse Generator**: This system generates an active-high pulse (10 us) every 60 ms.

- Operation: The circuit starts generating the periodic pulse when the s signal (usually a one-cycle pulse) is asserted.
  - ✓ Input: s (start signal).
  - ✓ Outputs: p (60ms periodic pulse).
  - ✓ Clock frequency: 100 MHz.

- Sketch the circuit: FSM + Datapath components. Specify all the I/Os of the FSM, as well as the signals connecting the FSM and the Datapath components (as in Problem 2).
  - ✓ Suggestion: The Datapath only needs two counters (one for 60 ms and one for 10 us). You can use the parametric counter with enable and synchronous clear (my_genpulse_sclr).
    - Counter Q: 10 us counter. For a clock period of 10 ns, it counts from 0 to 999.
    - Counter R: 60 ms counter. For a clock period of 10 ns, it counts from 0 to 6×10^6 - 1.
  - ✓ Provide the State Diagram (in ASM form) of the FSM.

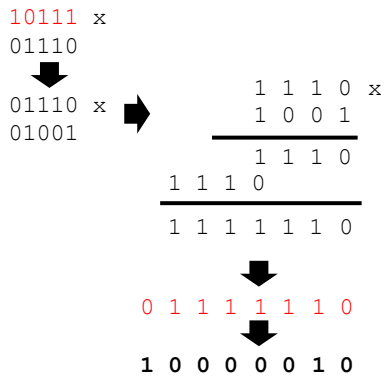


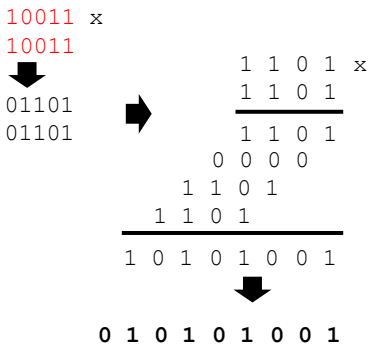


PROBLEM 4 (15 PTS)

- Calculate the result of the following operations, where the operands are signed integers. For the division, calculate both the quotient and the residue. **No procedure ≡ zero points.**

10111 × 01110	10011 × 10011	01101 ÷ 1001	1000011 ÷ 01010	101001 ÷ 10101





✓ $\frac{1000011}{01010} = \frac{-61}{10}$

```
      000110
1010 ) 111101
       1010
       ----
        1010
        1010
        ----
          01
```

To unsigned: $\frac{0111101}{01010}$

Unsigned Integer Division: $Q' = 110, R' = 1$
$\rightarrow Q = -Q' = 2C(0110) = 1010, \rightarrow R = -R' = 2C(01) = 11$

Verification: $-61 = (10 \times -6) - 1$

✓ $\frac{01101}{1001} = \frac{13}{-7}$

```
     0001
111 ) 1101
      111
      ---
      110
```

To unsigned: $\frac{01101}{0111}$

Unsigned Integer Division: $Q' = 1, R' = 110$
$\rightarrow Q = -Q' = 2C(01) = 1, \rightarrow R = 0110$

Verification: $13 = (-7 \times -1) + 6$

✓ $\frac{101001}{10101} = \frac{-23}{-11}$

```
     00010
1011 ) 10111
       1011
       ----
         01
```

To unsigned: $\frac{010111}{01011}$

Unsigned Integer Division: $Q' = 10, R' = 1$
$\rightarrow Q = Q' = 010, \rightarrow R = -R' = 2C(01) = 1$

Verification: $-23 = (-11 \times 2) - 1$

PROBLEM 5 (10 PTS)

- Compute the result of the additions and subtractions for the following fixed-point numbers.

UNSIGNED (1 pt. each)		**SIGNED**	
0.101011 + 1.01101	1.01011 − 0.0001101	10.001 + 1.011101	0.0101 − 1.1110101
	1100.1 + 0.1010101	1000.0101 − 101.01001	011.0101 + 1.0111101

UNSIGNED:

c_8=1	c_7=1	c_6=1	c_5=1	c_4=0	c_3=1	c_2=0	c_1=0	c_0=0	
	0.1	0	1	0	1	1	0		+
	1.0	1	1	0	1	0	0		
1	0.0	0	0	1	0	1	0		

b_7=0	b_6=0	b_5=1	b_4=1	b_3=1	b_2=1	b_1=1	b_0=0	
1.0	1	0	1	1	0	0		−
0.0	0	0	1	1	0	1		
1.0	0	1	1	1	1	1		

c_{11}=0	c_{10}=0	c_9=0	c_8=0	c_7=1	c_6=0	c_5=0	c_4=0	c_3=0	c_2=0	c_1=0	c_0=0	
1	1	0	0.1	0	0	0	0	0	0			+
			0.1	0	1	0	1	0	1			
1	1	0	1.0	0	1	0	1	0	1			

SIGNED:

c_9=1	c_8=1	c_7=0	c_6=0	c_5=1	c_4=1	c_3=0	c_2=0	c_1=0	c_0=0	
1	1	0.0	0	1	0	0	0			+
1	1	1.0	1	1	1	0	1			
1	0	1.1	0	0	1	0	1			

0.0	1	0	1	0	0	0		−
1.1	1	1	0	1	0	1		

➡

c_8=0	c_7=0	c_6=0	c_5=0	c_4=1	c_3=0	c_2=0	c_1=0	c_0=0	
0.0	1	0	1	0	0	0			+
0.0	0	0	1	0	1	1			
0.0	1	1	0	0	1	1			

1	0	0	0.0	1	0	1	0		−
1	1	0	1.0	1	0	0	1		

➡

c_9=0	c_8=0	c_7=0	c_6=0	c_5=1	c_4=1	c_3=1	c_2=1	c_1=0	c_0=0	
1	0	0	0.0	1	0	1	0	0		+
0	0	1	0.1	0	1	1	1			
1	0	1	1.0	0	0	0	1			

c_{10}=1	c_9=1	c_8=1	c_7=0	c_6=1	c_5=1	c_4=1	c_3=0	c_2=0	c_1=0	c_0=0	
	0	1	1.0	1	0	1	0	0	0		+
	1	1	1.0	1	1	1	1	0	1		
0	1	0	0.1	1	0	0	1	0	1		