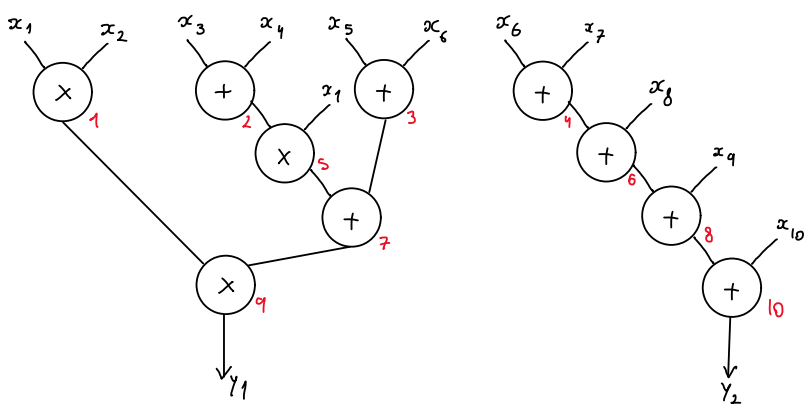
1) $Y_1 = (x_1 \times x_2) \times [(x_1 \times (x_3 + x_4)) + (x_5 + x_6)]$

$Y_2 = (((x_6 + x_7) + x_8) + x_9) + x_{10})$

Data flow graph



	Op	crit Path	Dep
1	×	2	—
2	+	4	—
3	+	3	—
4	+	4	—
5	×	3	2
6	+	3	4
7	+	2	3, 5
8	+	2	6
9	×	1	1, 7
10	+	1	8

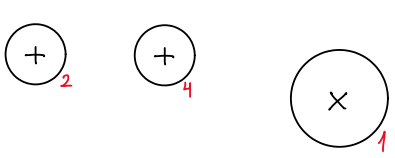
⇒

Priority List

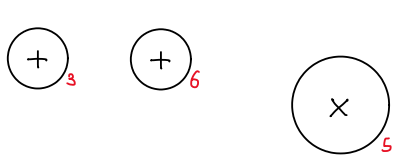
2	+	4	— ✓
4	+	4	— ✓
3	+	3	— ✓
5	×	3	2 ✓
6	+	3	4 ✓
1	×	2	— ✓
7	+	2	3, 5 ✓
8	+	2	6 ✓
9	×	1	1, 7
10	+	1	8

List scheduling

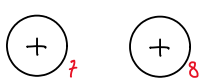
Ciclo 1



Ciclo 2



Ciclo 3



Ciclo 4

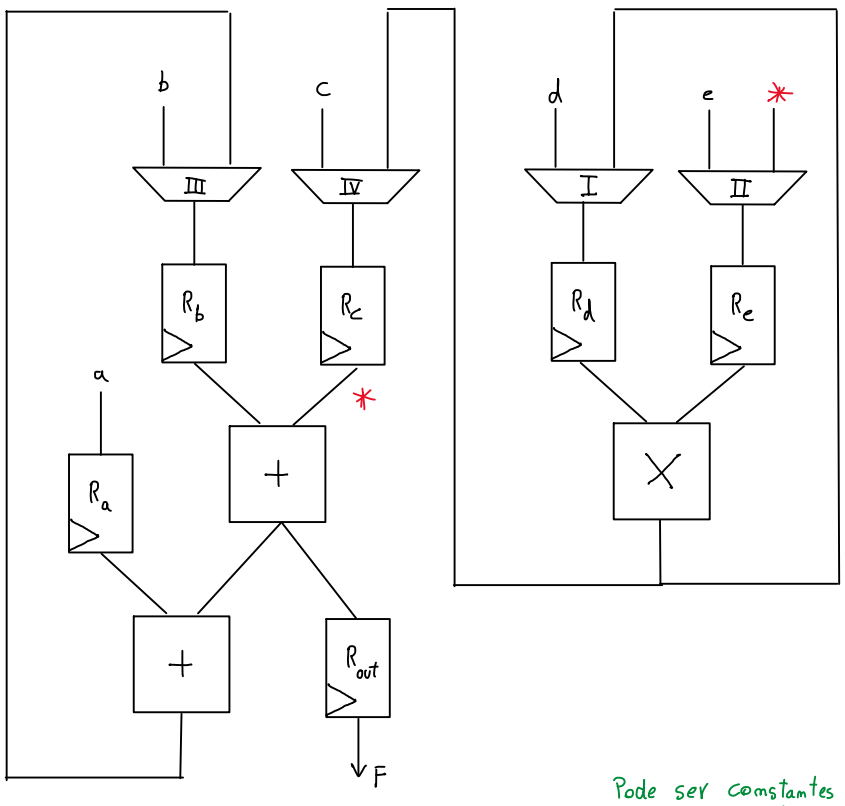
Melhor caso:

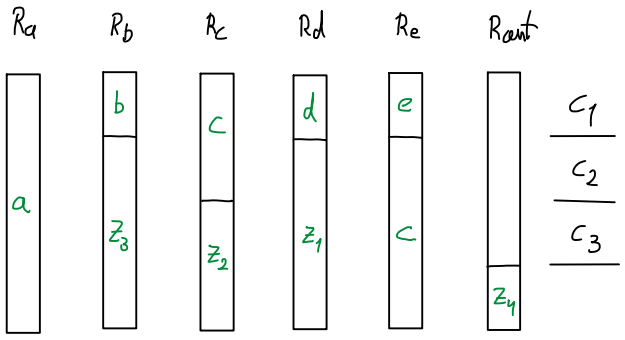
- Critical path

Registo - Mul - Registo

$t_{clk} = t_{p_{FF}} + t_{p_{Mul}} + t_{SET} = 2 + 1 + 14 = 17\,ms$

2.





$Z_1 = d \times e$

$Z_2 = c(d \times e) = c\,Z_1$

$Z_3 = a + b + c$

$Z_4 = Z_3 + Z_2$

Na saída, Z4, mais vale colocar um registo adicional do que usar um mux para colocar noutro registo já existente, porque os registos ocupam menos área que os mux's.
Esta é a datapath com melhor performance visto que o nosso critical path vai estar sempre limitado pelo multiplicador, neste caso em especifico o multiplicador tem uma saída que no ciclo a seguir é a sua entrada, logo TEMOS de ter um multiplicador seguido de um mux, e este caminho não consegue ser melhorado, logo será ele a limitar o nosso clock.
Relativamente à área, não creio que seja possível, reduzir muito mais!

Registos: 6

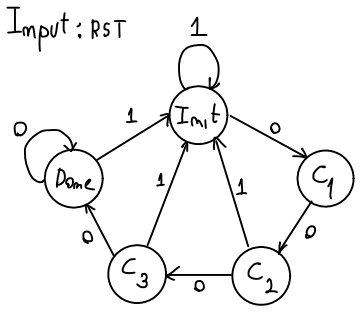
Mux 2:1: 4

Área = 2 x Add + Mul + 6 x Reg + 4 x Mux =

= 2 x 16 + 120 + 6 x 10 + 4 x 16 = 272

$$t_{clk} = \max \begin{cases} t_{P_{Mul}} + t_{P_{Mux}} + t_{P_{FF}} + t_{SET} = 14+3+2+1 = 20\,ms \rightarrow t_{clk} = 20\,ms \\ 2t_{P_{Add}} + t_{P_{Mux}} + t_{P_{FF}} + t_{SET} = 10+3+2+1 = 16\,ms \end{cases}$$

Latency = $3 \times t_{clk} = 60\,ms$

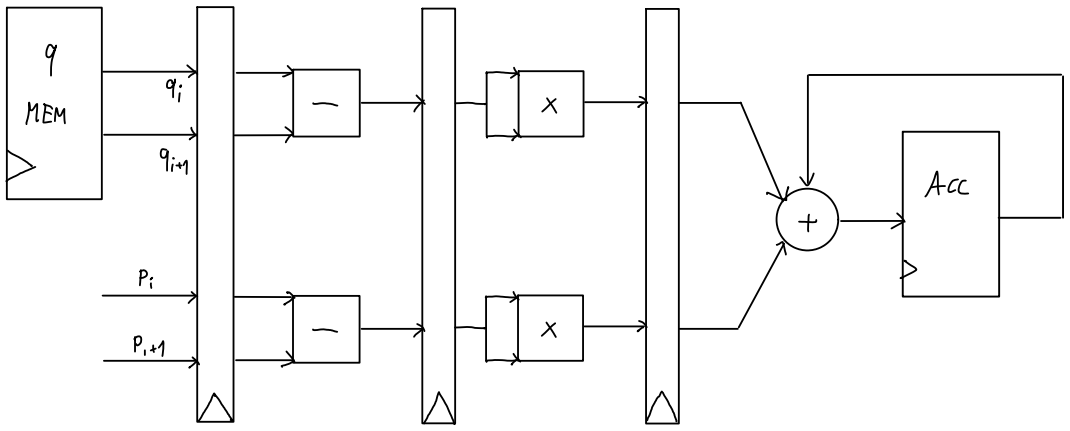


Pode ser comstantes

Pode ser tudo apenas um sinal!

	em_a	em_b	em_c	em_d	em_e	em_{Out}	Sel I	Sel II	Sel III	Sel IV	Estado
Init	1	1	1	1	1	—	0	0	0	0	1 1 0 0
C_1	—	1	0	1	1	—	1	1	1	—	0 1 1 0
C_2	—	0	1	—	—	—	—	—	—	1	1 0 1 0
C_3	—	—	—	—	—	1	—	—	—	—	1 1 1 0
Done	—	—	—	—	—	0	—	—	—	—	0 0 1 1
		b_2	b_3			b_2	b1				↳ Valid

3.



3 Barramentos

2 Subs

2 Mults

Throughput = 2 MAC

O ultimo andar pode ser só 1 DSP

4.

b)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity compd is
	port ( p, q : in std_logic_vector ( 31 downto 0 );
		rst, en, clk : in std_logic;
		addr : out std_logic_vector ( 8 downto 0 );
		d : out std_logic_vector ( 34 downto 0 )
	);
end compd;

architecture behavioral of compd is
	signal p1, p2, q1, q2 : unsigned(15 downto 0);
	signal s1, s2 : signed(16 downto 0);
	signal m1, m2 : signed(33 downto 0);
	signal SigD: signed(34 downto 0);
	signal SigAddr : signed(8 downto 0);

begin
p2 <= unsigned(p(31 downto 16));
p1 <= unsigned(p(15 downto 0));
q2 <= unsigned(q(31 downto 16));
q1 <= unsigned(q(15 downto 0));

-- arithmetic operations
s1 <= signed(p1) - signed(q1);
s2 <= signed(p2) - signed(q2);

m1 <= s1 * s1;
m2 <= s2 * s2;
SigD <= m1 + m2

process(clk)
begin
	if (clk = '1' and clk'event) then
		if (rst = 1) then
			d = ( others => '0' );
		elsif (en=1) then
			d = std_logic_vector(SigD);
	end if;
end process;


SigAddr = Sig(addr) + 1;

process (clk)
begin
	if (clk'event and clk = '1') then
	if (rst = 1) then
		addr = ( others => '0' );
	elsif (en=1) then
		addr = std_logic_vector(SigAddr);
	end if;
end process;
end behavioral;
```

peq: $Q_{12.4}$

a) $s_i : Q_{13.4}$

m: $Q_{26.8}$

D: $Q_{27.8}$

c) $P_1 = FFF0 = 1111\ 1111\ 1111.0000 = (2^{12}-1).0 = 4095.0$

$P_2 = 0018 = 0000\ 0000\ 0001.1000 = 1.5$

5. a) $t_{clk} = t_{P_{FF}} + t_{SET} + Max\{t_{PLog}\} = 4 + 2 + 10 = 16\ ms$

b) S.m $t'_{clk} = t_{clk} - \delta = 16 - 3 = 13\ ms$

c) $\delta < Min(t_{PLog}) + t_{P_{FF}} - t_{Hold}$

$\delta < 3 + 4 - 1 \iff \delta < 6\ ms \Rightarrow$ Não Vai funcionar independente do clk