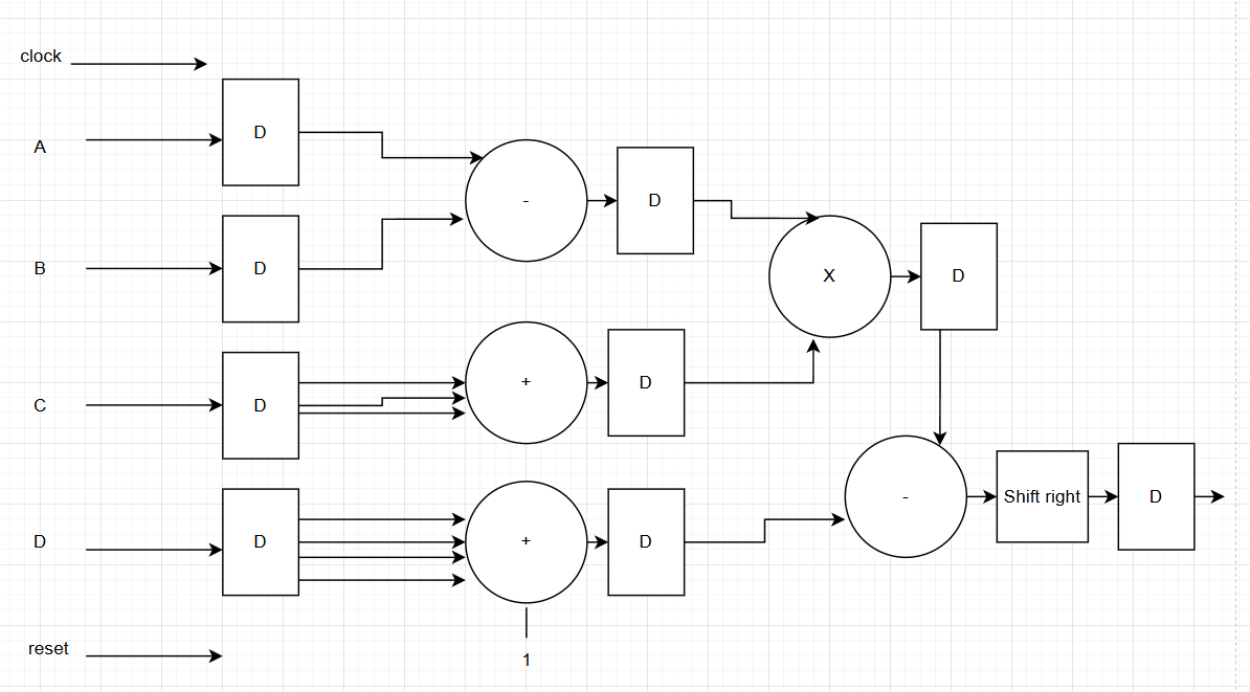
Реализуем схему с помощью 4-тактного конвейера:

1. Получаем данные

2. Простые операции суммирования и вычитания

3. Умножение

4. Вычитание и деление(сдвиг) результата



Все вычислительные блоки (сумматоры, умножители, сдвиг) должны учитывать знаки и разрядность чисел. Все регистры имеют тактирующий вход и вход сброса, подающиеся на вход устройства.

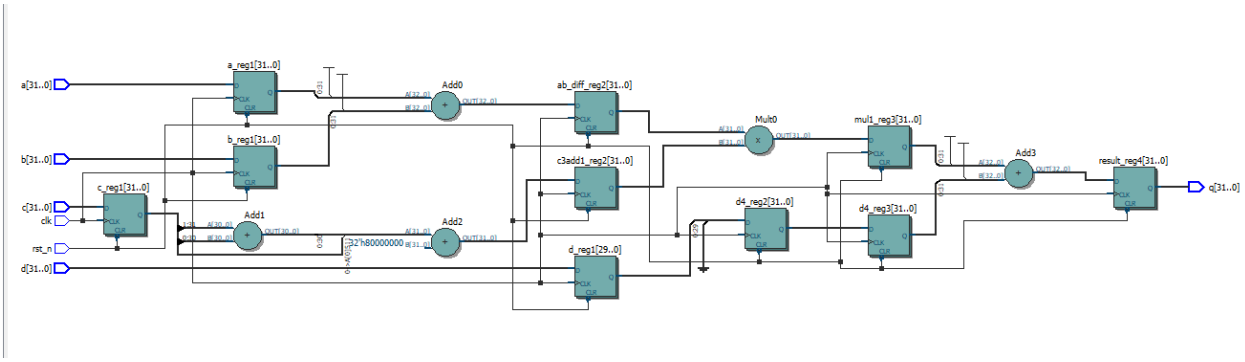
Получим, латентность = 4 такта

Код на SV:

```
module test_yadro #(
   parameter WIDTH = 32
)(
   input  logic                clk,
   input  logic                rst_n,
   input  logic signed [WIDTH-1:0] a,
   input  logic signed [WIDTH-1:0] b,
   input  logic signed [WIDTH-1:0] c,
   input  logic signed [WIDTH-1:0] d,
   output logic signed [WIDTH-1:0] q
);
```

```
logic signed [WIDTH-1:0] a_reg1, b_reg1, c_reg1, d_reg1;
logic signed [WIDTH-1:0] ab_diff_reg2, c3add1_reg2, d4_reg2;
logic signed [WIDTH-1:0] mul1_reg3, d4_reg3;
logic signed [WIDTH-1:0] result_reg4;

always_ff @(posedge clk or negedge rst_n) begin
   if (!rst_n) begin
      a_reg1 <= 0;
      b_reg1 <= 0;
      c_reg1 <= 0;
      d_reg1 <= 0;
   end else begin
      a_reg1 <= a;
      b_reg1 <= b;
      c_reg1 <= c;
      d_reg1 <= d;
   end
end

always_ff @(posedge clk or negedge rst_n) begin
   if (!rst_n) begin
      ab_diff_reg2 <= 0;
      c3add1_reg2 <= 0;
      d4_reg2 <= 0;
   end else begin
      ab_diff_reg2 <= a_reg1 - b_reg1;
      c3add1_reg2  <= 1 + 3 * c_reg1;
      d4_reg2      <= 4 * d_reg1;
   end
end

always_ff @(posedge clk or negedge rst_n) begin
   if (!rst_n) begin
      mul1_reg3 <= 0;
      d4_reg3 <= 0;
   end else begin
      mul1_reg3 <= ab_diff_reg2 * c3add1_reg2;
      d4_reg3 <= d4_reg2;
   end
```

```
  end


  always_ff @(posedge clk or negedge rst_n) begin
    if (!rst_n)
      result_reg4 <= 0;
    else
      result_reg4 <= (mul1_reg3 - d4_reg3) >>> 1;
  end


  assign q = result_reg4;


endmodule
```

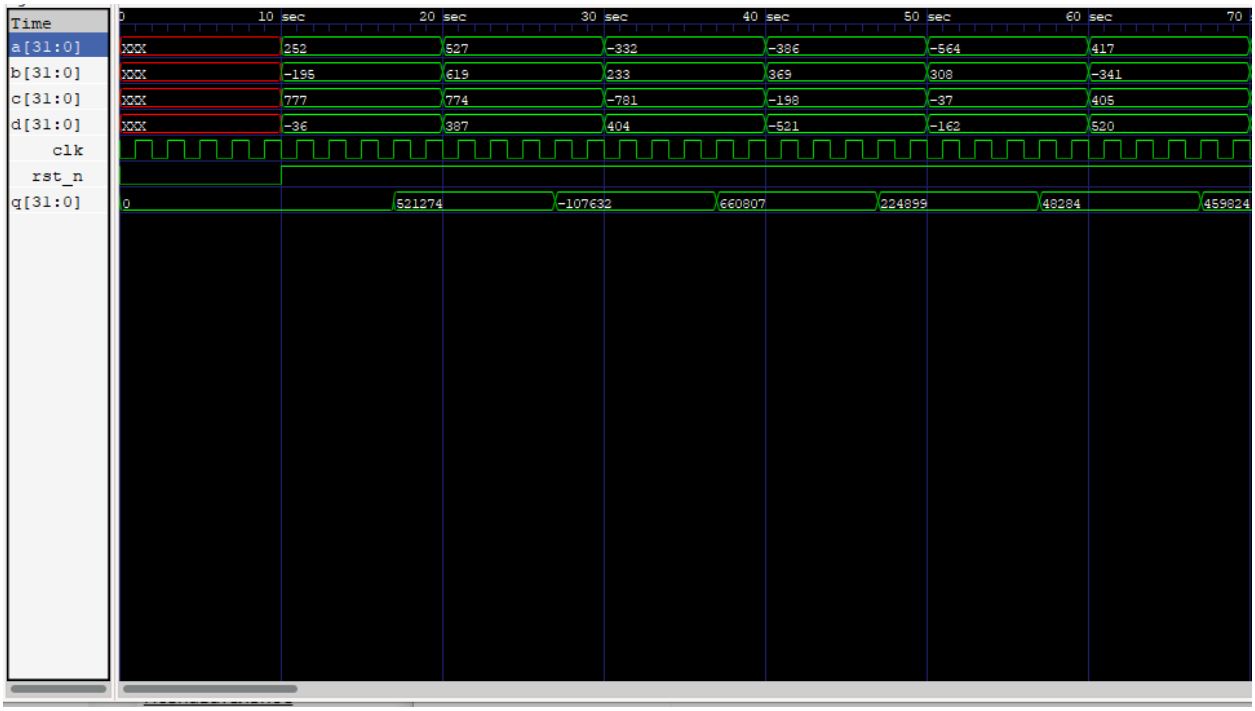


Синтез с схемой совпадает

Тестбенч с выводом всех групп сигналов и результатов в текстовый файл.

```
module tb ();
  parameter WIDTH    = 32;
   logic               clk;
   logic               rst_n;
   logic signed [WIDTH-1:0] a;
   logic signed [WIDTH-1:0] b;
   logic signed [WIDTH-1:0] c;
   logic signed [WIDTH-1:0] d;
  logic signed [WIDTH-1:0] q;
  test_yadro #(
    .WIDTH (WIDTH)
  ) DUT (
    .clk              ( clk),
    .rst_n             ( rst_n),
    .a                ( a    ),
```

```
    .b              ( b    ),
    .c              ( c    ),
    .d              ( d    ),
    .q              ( q    )
  );
  initial begin
    $dumpfile("wave.vcd");
    $dumpvars(0, tb);
  end
  initial begin
    clk <= 1'b0;
    forever begin
      #1 clk = ~clk;
    end
  end
  integer outfile;
  initial begin
    outfile = $fopen("output.txt", "w");
    rst_n = 0;
    #10;
    rst_n = 1;
    repeat(100) begin
    a = ($urandom_range(0,1)) ? $urandom_range(0,99999) : -$urandom_range(0,99999);
    b = ($urandom_range(0,1)) ? $urandom_range(0,99999) : -$urandom_range(0,99999);
    c = ($urandom_range(0,1)) ? $urandom_range(0,99999) : -$urandom_range(0,99999);
    d = ($urandom_range(0,1)) ? $urandom_range(0,99999) : -$urandom_range(0,99999);
    #10;
    $fdisplay(outfile, "time=%d|clk=%b|rst_n=%b|a=%d|b=%d|c=%d|d=%d|result=%d", $time, clk, rst_n, a, b, c, d, q);
  end
    $finish();
  end
endmodule
```



Программа на Python, реализующая ту же функцию и сверяющая ожидаемые результат с действительным:

```
def f(a: int, b:int, c: int, d: int) -> int:
        return ((a-b)*(1+3*c)-4*d)//2
def main():
        with open('output.txt') as file:
                data = file.read().split('\n')
        for row in data:
                if row:
                        time, clk, rst_n, a, b, c, d, result = list(map(int,map(lambda x: x.split('=')[1], row.split('|'))))
                        if rst_n == 1:
                                if f(a, b, c, d) != result:
                                        print('error', a, b, c, d)
if __name__=='__main__':
        main()
        print('all tests checked')
```

Возможные способы защиты от переполнения: на каждом вычислительном блоке добавить выходной сигнал overflow, показывающий переполнении, а на выходе схемы дизъюнкция всех сигналов в один выходной.

Аппаратные ресурсы

Total logic elements	287 / 114,480 (< 1 %)
Total combinational functions	122 / 114,480 (< 1 %)
Dedicated logic registers	249 / 114,480 (< 1 %)
Total registers	249
Total pins	162 / 529 (31 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	6 / 532 (1 %)
Total PLLs	0 / 4 (0 %)

Максимальная частота устройства в Slow модели

Slow 1000mV 100C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	105.19 MHz	105.19 MHz	clk	