4.6 Лабораторная работа 5
Проектирование блока умножения

4.6.1. **Цель лабораторной работы**:

овладеть знаниями и практическими навыками по проектированию вычислительных блоков последовательного действия, таких как блок умножения (MPU), включая его управляющий автомат (finite state mashine - FSM). Лабораторная работа также служит для овладения навыками программирования и отладки описания блоков последовательного действия и FSM на языке VHDL.

4.6.2. **Теоретические сведения**

Блоки умножения и деления последовательного действия применяются в простейших процессорах для выполнения соответствующих команд. Они имеют в несколько раз меньшие аппаратурные затраты, чем блоки параллельного умножения и деления. При умножении двух n-разрядных операндов получают 2n – разрядное произведение. Исходными данными для деления обычно являются 2n – разрядное делимое и n - разрядный делитель. Результатами являются n - разрядное частное и n – разрядный остаток. Различают операции умножения и деления с учетом и без учета знаков операндов.

Последовательные операции умножения и деления связаны со сдвигом одного или двух операндов и/или частного результата. В зависимости от выполнения тех или иных регистров блока регистрами сдвига, различают 4 основных схемы умножения и 2 схемы деления. Кроме того, блоки деления могут выполнять схему деления без восстановления и с восстановлением остатка.

Некоторые блоки для вычисления элементарных функций, например, выполняющие алгоритмы "цифра за цифрой" имеют принцип действия похожий на принцип действия последовательного блока умножения.

4.6.3 **Пример описания MPU**

Рассмотрим пример проектирования 16- разрядного MPU, выполняющего алгоритм выполнения умножения в дополнительном коде, начиная с младших разрядов множителя с неподвижным множимым и полным результатом. Тогда операция умножения имеет 2 операнда, поступающих по двум шинам и 2 шестнадцатиразрядных результата (старшая и младшая части произведения), выдаваемые последовательно через одну шину результата. Интерфейс блока:

```
entity MPU is
  port(C : in BIT;
        RST : in BIT;
        START:in BIT;-- пуск умножения по фронту
        OUTHL:in BIT;-- выдача старшего(0) или младшего(1) слова P
        DA : in BIT_VECTOR(15 downto 0);  -- операнд A
        DB : in BIT_VECTOR(15 downto 0);  -- операнд B
        RDY : out BIT;                    -- результат готов
        Z: out BIT;     -- признак нулевого результата
        N: out BIT;     -- 1 - отрицательный результат
        DP : out BIT_VECTOR(15 downto 0) );-- слово результата P
end MPU;
```

Блок состоит из операционной части и управляющего автомата. Операционная часть имеет структуру, представленную на рис.4.11.

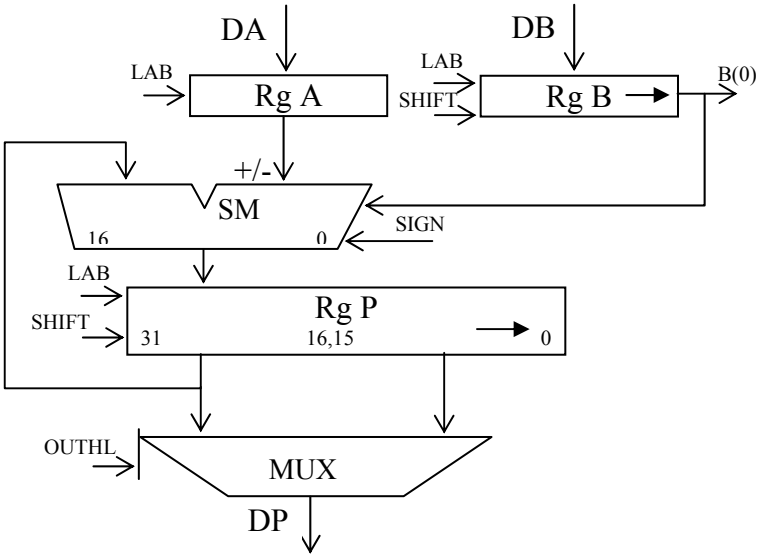

Рис.4.11. Структура операционной части блока умножения

Операционная часть описана в следующем объекте:

```
library IEEE;
use IEEE.Numeric_Bit.all;
entity MPU_OP is
  port(C : in BIT;
        RST : in BIT;
        LAB : in BIT;  -- загрузка A и B и обнуление P
        SHIFT : in BIT;  --разрешение сдвига A и P
        SIGN: in BIT;  -- знак сложения частных произведений
        OUTHL : in BIT;  --выдача старшего (0) или младшего (1) слова
        DA : in BIT_VECTOR(15 downto 0);
        DB : in BIT_VECTOR(15 downto 0);
        Z: out BIT;     -- признак нулевого результата
        N: out BIT;     -- 1 - отрицательный результат
        DP : out BIT_VECTOR(15 downto 0)); -- шина произведения
end MPU_OP;
architecture BEH of MPU_OP is
  signal A,B: bit_vector(15 downto 0); -- операнды A и B
  signal S: signed(16 downto 0);   -- сумма частных произведений
  signal P: signed(31 downto 0);   -- произведение
begin
  RG_A:process(C,RST)    -- регистр операнда A
  begin
        if RST='1' then
              A<=X"0000";
        elsif C='1' and C'event then
              if LAB='1' then
                    A<=DA;                    -- запись в регистр
              end if;
        end if;
  end process;
```

```
RG_B:process(C,RST)    -- регистр операнда B
begin
        if RST='1' then
                B<=X"0000";
        elsif C='1' and C'event then
                if LAB='1' then
                        B<=DB;                          -- запись в регистр
                elsif SHIFT='1' then
                        B<='0'& B(15 downto 1);--сдвиг регистра
                end if;
        end if;
end process;
-- Сумматор - вычитатель частных произведений
SM:S<= P(31)&P(31 downto 16)-signed(A) when B(0)='1' and SIGN='1' else
      P(31)&P(31 downto 16)+signed(A) when B(0)='1' else
      P(31)&P(31 downto 16);

RG_P:process(C,RST,P)    -- регистр P
     variable zi:bit;
begin
        if RST='1' then
                P<=X"00000000";
        elsif C='1' and C'event then
                if LAB='1' then
                        P<=X"00000000";             -- запись в регистр  0
                elsif SHIFT='1' then
                        P<=S & P(15 downto 1); --сдвиг регистра
                end if;
        end if;
        zi:='0';          -- цикл определения нулевого результата
        for i in P'range loop
                zi:=zi or P(i);
        end loop;
        Z<= not zi;             -- флаг нулевого произведения
        N<=P(31);               -- флаг знака
end process;
        --выходной мультиплексор
MUX_P:DP<= bit_vector(P(15 downto 0)) when OUTHL='1' else
        bit_vector(P(31 downto 16));
end BEH;
```

В процессах RG_A, RG_B, RG_P описаны регистры операндов и произведения. Оператором SM представлен сумматор-вычитатель, который в зависимости от очередного бита множителя B(0) прибавляет или нет к сумме частных произведений P множимое A. В последнем такте умножения, когда выполняется умножение на знаковый разряд, т.е. при SIGN='1' , из P вычитается множимое A.

Признак нулевого результата Z вычисляется в 32- разрядной схеме ИЛИ-НЕ, которая реализована в операторе **loop**. Выходной мультиплексор MUX_P выдает старшее или младшее слово произведения в зависимости от сигнала OUTHL.

Управляющий автомат для блока умножения описан в следующей архитектуре.

```
entity MPU_FSM is
  port(C : in BIT;
        RST : in BIT;
        START : in BIT;    -- начать умножение
        LAB : out BIT;     -- загрузка A и B и обнуление P
        SHIFT : out BIT;   --разрешение сдвига A и P
        SIGN: out BIT;     -- знак сложения
        RDY : out BIT);    --конец умножения
end MPU_FSM;
architecture BEH of MPU_FSM is
  type STATES is (s1,s2,s3,s4,s5,s6,s7,s8,s9,s10, --состояния автомата
  s11,s12,s13,s14,s15,s16,s17,finish);
  signal st:STATES;
begin
  STATE:process(C,RST)      -- регистр состояния
  begin
        if RST='1' then
              st<=finish;
        elsif C='1' and C'event then
              case st is      -- определение следующего состояния
                    when s1=> st<=s2;
                    when s2=> st<=s3;
                    . . .
                    when s16=> st<=s17;
                    when s17=> st<=finish;
                    when finish=> if START ='1' then
                                        st<=s1;
                                  else
                                    st<=finish;
                                  end if;
              end case;
        end if;
  end process;
-- логика выходных сигналов
  LAB<='1' when st=s1 else '0';
  SHIFT<='0' when st=s1 or st=finish else '1';
  SIGN<='1' when st=s17 else '0';
  RDY<='1' when st=finish else '0';
end BEH;
```

Автомат имеет 18 состояний, заданных типом . В операторе STATE описан регистр состояния st и функция изменения состояния, заданная оператором **case**. Каждая строка оператора **case** определяет, какое новое состояние примет регистр состояния, если автомат находится в данном состоянии (несколько строк заменено многоточием). Начальное состояние автомата управления умножением после сброса - finish. Далее состояния автомата изменяются циклически и он останавливается в состоянии finish с циклом ожидания прихода сигнала START.

Параллельные операторы присваивания задают логику выходных сигналов автомата, которая зависит от его текущего состояния. Таким образом, данный автомат – это автомат Мура.

Архитектура блока умножения описана ниже.

```
architecture BEH of MPU is
    component MPU_OP is   port(C : in BIT;
                RST : in BIT;
                LAB : in BIT;  -- загрузка A и B и обнуление P
                SHIFT : in BIT;  --разрешение сдвига A и P
                SIGN: in BIT;  -- знак сложения
                OUTHL : in BIT;  --выдача старшего(0) или младшего(1) слова
                DA : in BIT_VECTOR(15 downto 0);
                DB : in BIT_VECTOR(15 downto 0);
                Z: out BIT;    -- признак нулевого результата
                N: out BIT;    -- 1 - отрицательный результат
                DP : out BIT_VECTOR(15 downto 0)); -- шина произведения
    end component;
    component MPU_FSM is port(C : in BIT;-- управляющий автомат
                RST : in BIT;
                START : in BIT; -- начать умножение
                LAB : out BIT;   -- загрузка A и B и обнуление P
                SHIFT : out BIT; --разрешение сдвига A и P
                SIGN: out BIT;   -- знак сложения
                RDY : out BIT);  --конец умножения
    end component ;
    signal lab,shift,sign:bit;
begin
    U_OP:MPU_OP port map(c,RST,          -- Операционная часть
          LAB=>lab, SHIFT=>shift,
          SIGN=>sign,OUTHL=>OUTHL,
          DA=>DA,     DB=>DB,
          Z=>Z,N=>N,
          DP=>DP);
    U_FSM:MPU_FSM port map(C,RST,  -- Управляющий автомат
          START=>START,    LAB=>lab,
          SHIFT=>shift,  SIGN=>sign,
          RDY=>RDY);
end BEH;
```

Структура блока умножения, полученная с помощью утилиты Code2Graphics, показана на рис. 4.12.

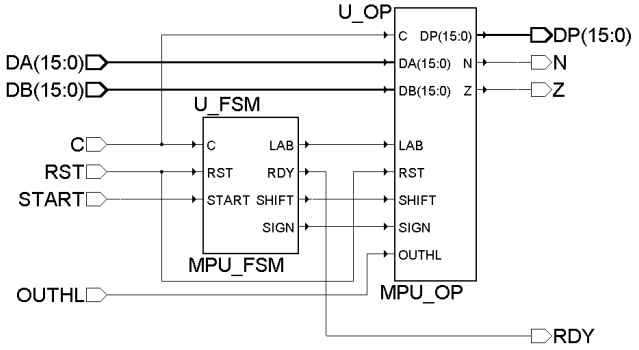

Рис.4.12. Структура блока умножения

4.6.4. Испытательный стенд для MPU

Испытательный стенд для MPU включает в себя контролируемый объект – блок умножения, источники тестовых сигналов и процесс контроля результатов умножения. Его описательная часть представлена ниже.

```
begin
  c<=not c after 5 ns;       --генератор синхросигнала с периодом 10 нс
  rst<='1', '0' after 33 ns;     -- генератор сигнала сброса
  DA<=X"1234", X"8910" after 400 ns;      -- подача операнда A
  DB<=X"f234", X"7654" after 400 ns;      --  подача операнда B
  OUTHL<='1' after 350 ns, '0' after 450 ns, '1' after 790 ns;
  START<= '1' after 40 ns, '0' after 50 ns, '1' after 500 ns, '0' after 600 ns;
  UUT : mpu port map (C => C,RST => RST,--тестируемое устройство
                      START => START,OUTHL => OUTHL,
                      DA => DA,DB => DB,
                      RDY => RDY,        Z => Z,
                      N => N,       DP => DP);
   TST:process begin  -- проверка результатов умножения
                      wait for 300 ns;
          Assert DP=X"FF04" report "Ошибка в старшем слове P";
                      wait for 100 ns;
          Assert DP=X"DA90" report "Ошибка в младшем слове P";
                      wait for 300 ns;
          Assert DP=X"C906" report "Ошибка в старшем слове P";
                      wait for 100 ns;
          Assert DP=X"5940" report "Ошибка в младшем слове P";
                      wait;
  end process;
end TB_ARCHITECTURE;
```

Правильность выполнения алгоритма умножения проверяется в процессе TST. При этом через соответствующие промежутки времени состояние шины сравнивается с ожидаемым результатом и в случае несовпадения на консоль выдается сообщение об ошибке. Также правильность выходных сигналов можно проверить, исследуя графики входных и выходных сигналов, а также состояние st его управляющего автомата, которые представлены на рис. 4.13.

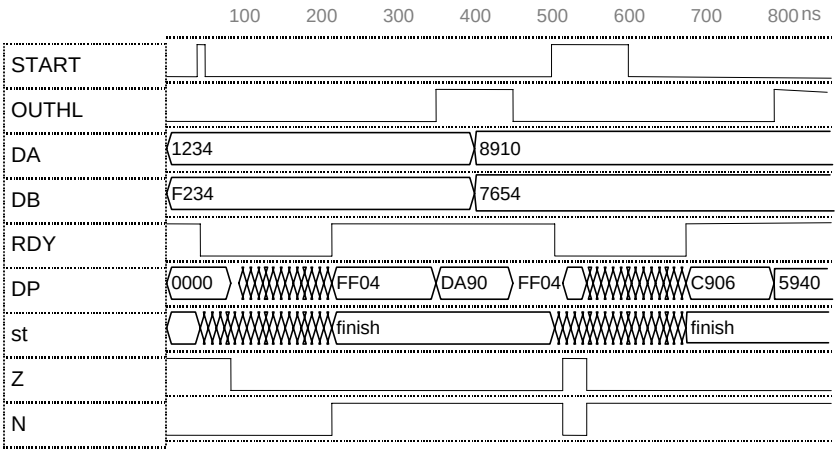

Рис.4.13. Графики входных и выходных сигналов блока умножения.

4.6.5. Вопросы по лабораторной работе.

Какие алгоритмы последовательного умножения и деления вы знаете?

Что такое алгоритм "цифра за цифрой" и как он связан с алгоритмом последовательного умножения?

Почему вычислительное устройство проектируют в виде совокупности операционного и управляющего автоматов?

Для чего при описании сумматора применяют подтип signed?

Приведите 2 способа программирования регистра сдвига в VHDL.

Приведите пример программирования n-разрядной схемы И в VHDL.

Приведите 4 способа программирования мультиплексора в VHDL.

Нарисуйте граф-схему управляющего автомата блока умножения.

Опишите на VHDL управляющий автомат блока умножения, выполненный на основе счетчика.

Опишите на VHDL блок умножения, выполненный на основе параллельного умножителя.