Проектирование умножителя целых чисел со знаком методом правого сдвига и сложения в базисе ПЛИС

Андрей СТРОГОНОВ,
д. т. н.
andreistrogonov@mail.ru
Александр ВИНОКУРОВ
Алан ДЖИОТИ

Показан пример проектирования последовательностного универсального умножителя целых чисел, представленных в дополнительном коде, методом правого сдвига и сложения (MAC-блок) в базисе ПЛИС. Использование этого метода для умножения чисел в базисе сигнальных процессоров чрезвычайно популярно у разработчиков РЭА. На базе этого метода реализуются схемы быстрого умножения (например, кодирование по Буту, которое позволяет уменьшать число частичных произведений вдвое, умножение по основанию 4, модифицированное кодирование по Буту) [1–4].

Введение

Существует огромное число разновидностей матричных умножителей, превосходящих по скорости последовательностные умножители, основанные на методе сдвига и сложения. Известны и более сложные процедуры, например, с представлением суммирования в древовидном формате. Авторы не ставят цель разработать умножитель, превышающий по своим техническим характеристикам матричные умножители. Мы хотим показать читателю процедуру умножения чисел, представленных в дополнительном коде, методом правого сдвига и сложения, пригодную для реализации в базисе ПЛИС.

В качестве сравнения рассмотрим один из хорошо известных умножителей чисел в дополнительном коде, так называемый умножитель Бо-Вули (Baugh-Wooley). Если A и B — целые десятичные числа со знаком, то в дополнительном двоичном коде они будут представлены в виде:

$$A = -a_{m-1}2^{m-1} + \sum_{i=0}^{m-2} a_i 2^i$$

и

$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i.$$

Тогда

$$A \times X = a_{m-1}x_{n-1}2^{m+n-2} + \sum_{i=0}^{m-2}\sum_{j=0}^{n-2} a_i x_j 2^{i+j} - \\ - \sum_{i=0}^{m-2} x_{n-1}a_i 2^{i+n-1} - \sum_{j=0}^{n-2} a_{m-1}x_j 2^{j+m-1}. \quad (1)$$

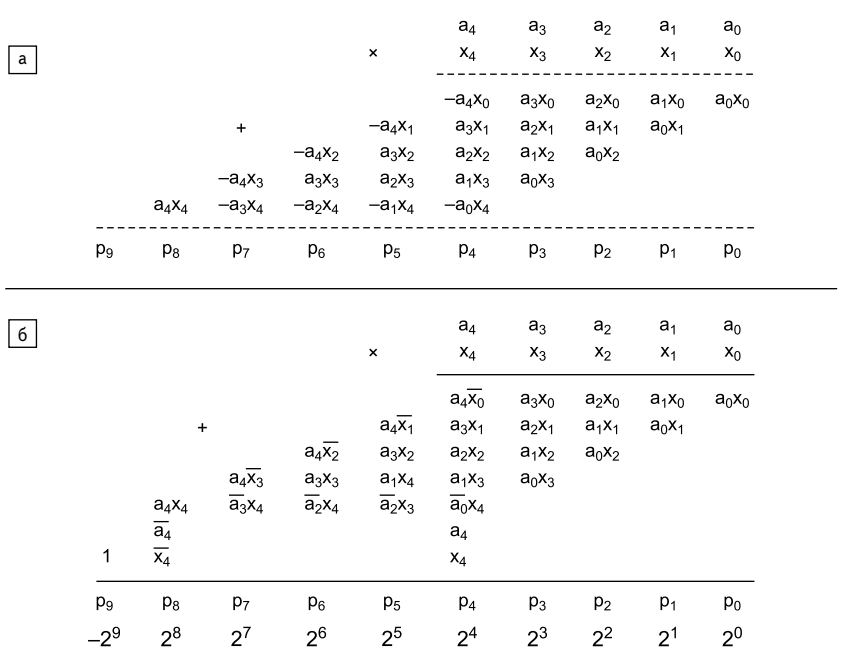

Рис. 1. Пример умножения чисел 5×5, представленных дополнительным кодом: а) по формуле (1); б) по схеме Бо-Вули

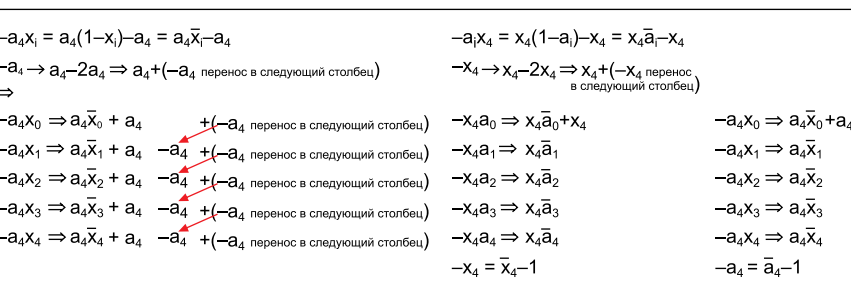

Рис. 2. Схема преобразований Бо-Вули

Таблица 1. Примеры умножения для различных случаев по схеме Бо-Вули

Случай 1	Множимое — положительное число, множитель — отрицательное число	0 1 1 0 1 = 13 × 1 1 0 1 1 = –5 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 + 1 1 1 1 1 0 1 1 1 1 1 1 = –65
Случай 2	Множимое — отрицательное число, множитель — положительное число	1 1 0 1 1 = –5 × 0 1 1 0 1 = 13 0 1 0 1 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 0 1 1 + 1 0 1 1 1 0 1 1 1 1 1 1 = –65
Случай 3	Множимое и множитель — положительные числа	0 1 1 0 1 = 13 × 0 0 1 0 1 = 5 0 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 + 1 0 0 0 0 1 0 0 0 0 0 1 = 65
Случай 4	Множимое и множитель — отрицательные числа	1 0 0 1 1 = –13 × 1 1 0 1 1 = –5 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 1 + 1 0 0 0 0 1 0 0 0 0 0 1 = 65

На рис. 1а показан пример умножения чисел 5×5, представленных дополнительным кодом по формуле (1), а на рис. 1б — пример умножения чисел 5×5, представленных дополнительным кодом по схеме Бо-Вули. На рис. 2 приведена схема преобразований, позволяющая перевести частичные произведения со знаком в беззнаковые величины. А на рис. 3 — матричный умножитель Бо-Вули размерностью 5×5 чисел, представленных в дополнительном коде. Наличие полных сумматоров (FA) в матричной структуре такого умножителя является главным достоинством при реализации в базисе заказных БИС, а его недостаток — пониженное быстродействие за счет увеличения высоты столбца с 5 до 7.

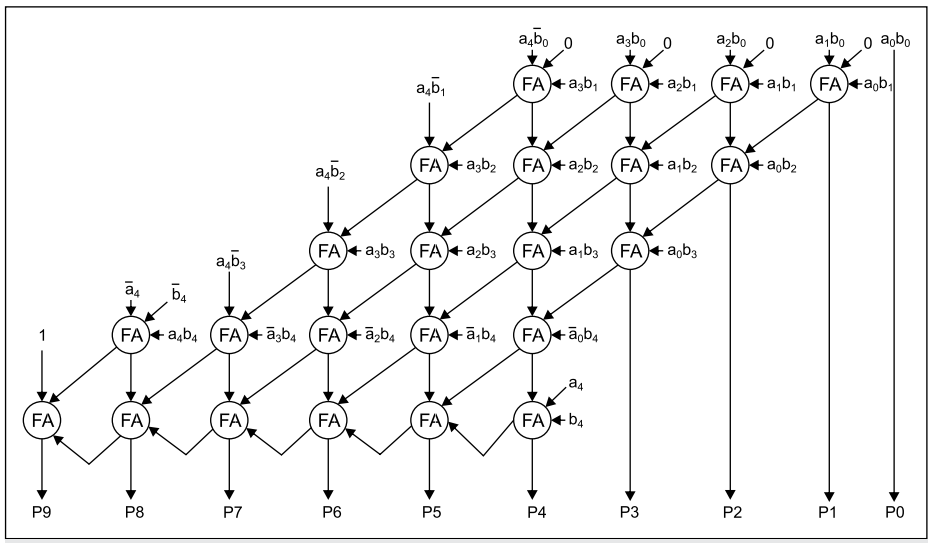

Рис. 3. Матричный умножитель Бо-Вули размерностью 5×5 чисел, представленных в дополнительном коде

В таблице 1 показаны примеры умножения для различных случаев по схеме Бо-Вули.

Процедура умножения чисел, представленных в дополнительном коде, методом правого сдвига и сложения

В качестве базовой схемы разрабатываемого МАС-блока возьмем схему умножителя целых беззнаковых чисел с управляющим автоматом на четыре состояния [5].

Рассмотрим вычитание чисел со знаком в столбик (рис. 4). Дополнение до двух можно получить, если прибавить «1» к результату обращения. Обращение логически эквивалентно инверсии каждого бита в числе. Вентили «исключающее ИЛИ» можно применить для избирательной инверсии в зависимости от значения управляющего сигнала.

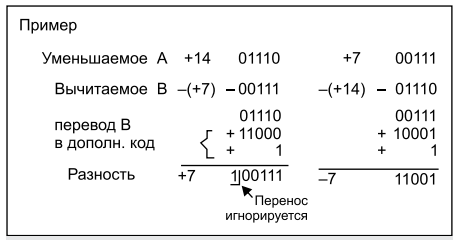

Рис. 4. Вычитание с использованием дополнительного кода (дополнение до двух). Осуществляется инвертирование вычитаемого и суммирование, а также перенос «1» в младший разряд с последующим сложением

Прибавление «1» к результату обращения можно реализовать, задавая «1» на входе переноса сумматора.

На рис. 5 показан принцип умножения чисел, представленных дополнительным кодом, на примере умножения (–5)×(–3).

Представление процесса умножения в точечной нотации, в которой под каждой точкой подразумевается «лог. 1» или «лог. 0», позволяет получить рекуррентную формулу (рис. 6).

В таблице 2 показы примеры умножения, рассматриваемые при разработке схемы универсального умножителя.

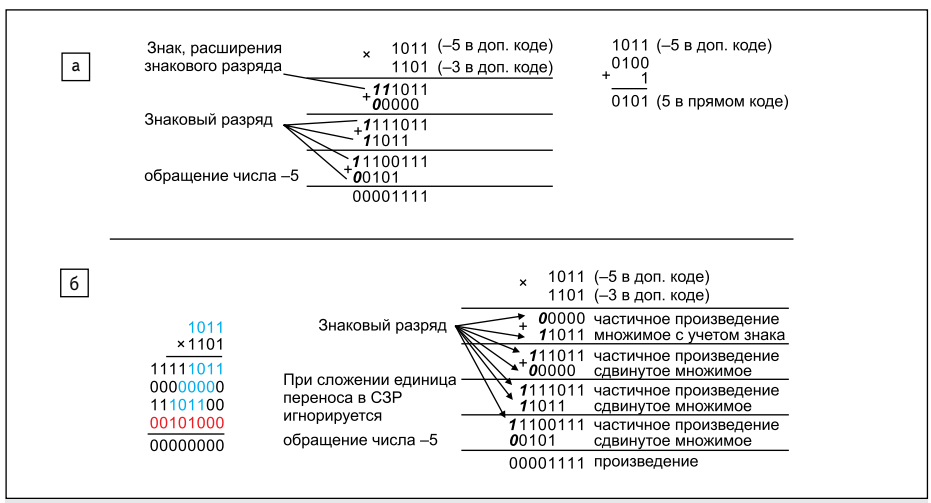

Рис. 5. Умножение: а) в столбик; б) методом сдвига множимого и последующего сложения с частичным произведением (умножение чисел (–5)×(–3), представленных в дополнительном коде)

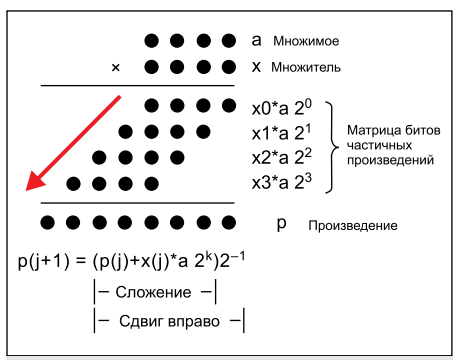

Рис. 6. Представление процесса умножения методом правого сдвига и сложения в точечной нотации

Таблица 2. Примеры умножения

Случай 1	Множимое — отрицательное число, множитель — положительное число	(–10)×11
Случай 2	Множимое и множитель — отрицательные числа	(–10)×(–11)
Случай 3	Множимое и множитель — четные отрицательные числа	(–4)×(–4)
Случай 4	Множимое — положительное число, множитель — отрицательное число	5×(–3)

	MSB	2 тетрада	1 тетрада
a(–10) x(11)		1 0 1 1 0 0 1 0 1 1	
p(0) +x0*a		0 0 0 0 0 1 0 1 1 0	
2p(1) p(1) +x1*a	1	1 0 1 1 0 1 1 0 1 1 1 0 1 1 0	0
2p(2) p(2) +x2*a	1	1 0 0 0 1 1 1 0 0 0 0 0 0 0 0	0 1 0
2p(3) p(3) +x3*a	1	1 1 0 0 0 1 1 1 0 0 1 0 1 1 0	1 0 0 1 0
2p(4) p(4) +x4*a	1	1 0 0 1 0 1 1 0 0 1 0 0 0 0 0	0 1 0 0 0 1 0
2p5 p5	1	1 1 0 0 1 1 1 1 0 0	0 0 1 0 1 0 0 1 0

а

	MSB	2 тетрада	1 тетрада
a(–10) x(–11)		1 0 1 1 0 1 0 1 0 1	
p(0) +x0*a		0 0 0 0 0 1 0 1 1 0	
2p(1) p(1) +x1*a	1	1 0 1 1 0 1 1 0 1 1 0 0 0 0 0	0
2p(2) p(2) +x2*a	1	1 1 0 1 1 1 1 1 0 1 1 0 1 1 0	0 1 0
2p(3) p(3) +x3*a	1	1 0 0 1 1 1 1 0 0 1 0 0 0 0 0	1 0 1 1 0
2p(4) p(4) (–x4*a)	1	1 1 0 0 1 1 1 1 0 0 0 1 0 1 0	1 1 0 1 1 1 0
2p5 p5	0	0 0 1 1 0 0 0 0 1 1	1 1 1 0 0 1 1 1 0

б

Рис. 7. Принцип умножения методом правого сдвига и сложения:
а) умножение (–10)×11; б) умножение (–10)×(–11)

На рис. 7 показан принцип умножения методом правого сдвига и сложения для двух случаев в формате проектируемого MAC-блока. В первом случае осуществляется умножение числа со знаком (–10) на число (11), а во втором — умножение (–10) на (–11). В первом случае множимое (–10) переводится в дополнительный код (дополнение до двух).

	MSB	2 тетрада	1 тетрада
a(–4) x(–4)		1 1 1 0 0 1 1 1 0 0	
p(0) +x0*a		0 0 0 0 0 0 0 0 0 0	
2p(1) p(1) +x1*a	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
2p(2) p(2) +x2*a	0	0 0 0 0 0 0 0 0 0 0 1 1 1 0 0	0 0 0
2p(3) p(3) +x3*a	1	1 1 1 0 0 1 1 1 1 0 1 1 1 0 0	0 0 0 0 0
2p(4) p(4) –(x4*a)	1	1 1 0 1 0 1 1 1 0 1 0 0 1 0 0	0 0 0 0 0 0 0
2p5 p5	0	0 0 0 0 1 0 0 0 0 0	0 0 0 0 1 0 0 0 0

а

	MSB	2 тетрада	1 тетрада
a(–8) x(–8)		1 1 0 0 0 1 1 0 0 0	
p(0) +x0*a		0 0 0 0 0 0 0 0 0 0	
2p(1) p(1) +x1*a	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
2p(2) p(2) +x2*a	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0
2p(3) p(3) +x3*a	1	0 0 0 0 0 0 0 0 0 0 1 1 0 0 0	0 0 0 0 0
2p(4) p(4) (–x4*a)	1	1 1 0 0 0 1 1 1 0 0 0 1 0 0 0	0 0 0 0 0 0 0
2p5 p5	0	0 0 1 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0

б

Рис. 8. Принцип умножения методом правого сдвига и сложения:
а) умножение (–4)×(–4); б) умножение (–8)×(–8)

Множитель (11) — целое положительное число, расширенное знаковым разрядом 0, представлено в прямом коде. Для удвоенных частичных произведений 2p(1), 2p(2), 2p(3), 2p(4) и 2p(5) в поле MSB (название полей произвольное) необходимо добавить «лог. 1». При формировании частичных произведений методом правого сдвига p(1), p(2), p(3), p(4) и p(5) «лог. 1» из поля MSB попадает в старший разряд второй тетрады. (Название «тетрада» в данном случае некорректно, так как это поле уже 5-разрядное, но сохранено для преемственности с принципом умножения беззнаковых чисел.) При сложении p(3) с x3*a единица переноса в старший разряд, то есть в поле MSB, игнорируется. Данная схема вычислений справедлива только для случая, когда в младшем разряде множителя находится «1».

При умножении двух отрицательных чисел, представленных дополнительным кодом, например (–10)×(–11), следует произвести два действия.

Первое: необходимо учесть знак при представлении числа в дополнительном коде, что достигается обращением произведения старшего разряда множителя на множимое с последующим прибавлением «1» к младшему разряду. Дополнительный код произведения (–x4*a) при x4 = 1 есть число «10». Перевод в дополнительный код произведения (–x4*a) должен быть осуществлен до операции сложения, то есть до получения удвоенного частичного произведения 2p(5).

Второе: при формировании удвоенного частичного произведения 2p(5) необходимо провести коррекцию, то есть в поле MSB поставить логический ноль.

Рассмотрим умножение четных чисел со знаком (рис. 8). При умножении (–4)×(–4) в поле MSB для удвоенных значений частичных произведений 2p(1) и 2p(2) должны стоять нули. А при умножении (–8)×(–8) в поле MSB должен быть еще и «0» для частичного произведения 2p(3). Принцип умножения не отличается от умножения чисел, представленных дополнительным кодом, например (–10)×(–11).

На рис. 9 показан принцип умножения методом правого сдвига и сложения в случае, когда множимое — положительное, а множитель — отрицательное число.

Разработанный MAC-блок также способен умножать числа без знака. Для этого применяется принцип, показанный на рис. 10. Единица переноса при сложении в поле «2-я тетрада» уже не игнорируется, а переносится в поле Cout. (Сигнал Cout является выходом переноса многоразрядного сумматора масштабирующего аккумулятора.)

	MSB	2 тетрада	1 тетрада
a(–5) x(–3)		0 0 1 0 1 1 1 1 0 1	
p(0) +x0*a		0 0 0 0 0 0 0 1 0 1	
2p(1) p(1) +x1*a	0	0 0 1 0 1 0 0 0 1 0 0 0 0 0 0	1
2p(2) p(2) +x2*a	0	0 0 0 1 0 0 0 0 0 1 0 0 1 0 1	1 0 1
2p(3) p(3) +x3*a	0	0 0 1 1 0 0 0 0 1 1 0 0 1 0 1	0 1 0 0 1
2p(4) p(4) –(x4*a)	0	0 1 0 0 0 0 0 1 0 0 1 1 0 1 1	0 0 1 0 0 0 1
2p5 p5	1	1 1 1 1 1 1 1 1 1 1	0 0 0 1 1 0 0 0 1

а

	MSB	2 тетрада	1 тетрада
a(–2) x(–2)		0 0 0 1 0 1 1 1 1 0	
p(0) +x0*a		0 0 0 0 0 0 0 0 0 0	
2p(1) p(1) +x1*a	0	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	0
2p(2) p(2) +x2*a	0	0 0 0 1 0 0 0 0 0 1 0 0 0 1 0	0 0 0
2p(3) p(3) +x3*a	0	0 0 0 1 1 0 0 0 0 1 0 0 0 1 0	0 0 1 0 0
2p(4) p(4) (–x4*a)	0	0 0 0 1 1 0 0 0 0 1 1 1 1 1 0	1 0 0 1 1 0 0
2p5 p5	1	1 1 1 1 1 1 1 1 1 1	1 1 0 0 1 1 1 0 0

б

Рис. 9. Принцип умножения методом правого сдвига и сложения:
а) умножение 5×(–3); б) умножение 2×(–2)

	Cout	2 тетрада	1 тетрада	1 и 2 тетрады
a x		1 1 1 1 1 1 1 1		
$p^{(0)}$ $+x_0a$		0 0 0 0 1 1 1 1		
$2p^{(1)}$ $p^{(1)}$ $+x_1a$	0	1 1 1 1 0 1 1 1 1 1 1 1	1 0 0 0	120
$2p^{(2)}$ $p^{(2)}$ $+x_2a$	1	0 1 1 0 1 0 1 1 1 1 1 1	0 1 0 0	180
$2p^{(3)}$ $p^{(3)}$ $+x_3a$	1	1 0 1 0 1 1 0 1 1 1 1 1	0 0 1 0	210
$2p^{(4)}$ $p^{(4)}$	1	1 1 0 0 1 1 1 0	0 0 0 1	225

Рис. 10. Принцип умножения методом правого сдвига и сложения (умножение 15×15)

Разработка схемы умножителя в базисе ПЛИС

Поскольку принципы умножения чисел со знаком и без знака отличаются, то необходимо откорректировать код языка VHDL цифрового автомата (пример). В коде содержится шесть операторов process, которые выполняются параллельно.

Первые три оператора process реализуют цифровой автомат Мура на четыре состояния с логикой переходов и логикой формирования выхода для управления процессом умножения. Память состояний (регистр состояний) автомата тактируется фронтом синхроимпульса. Следует принимать во внимание, что на синхровход автомата clk подключен инвертор. Автомат формирует три управляющих сигнала ena_add_temp (сигнал разрешения суммирования многоразрядному сумматору масштабирующего аккумулятора), load_acc (сигнал разрешения загрузки в регистр со сдвигом вправо) и ena_shift_temp (сигнал разрешения сдвига) (рис. 11).

Четвертый оператор process представляет собой 5-разрядный суммирующий счетчик на сигнале cnt_sub_add (тактируется срезом синхроимпульса clk'event and clk = '0'). Таким образом, удается обеспечить конвейерный режим работы масштабирующего аккумулятора, при этом формируемые сигналы ena_add_temp, load_acc и ena_shift_temp являются синхронными для всех регистров умножителя. Так, второй передний фронт синхроимпульса оказывается ровно над половиной высокого уровня сигнала ena_add_temp. Это обеспечивает корректную работу синхронного многоразрядного сумматора масштабирующего аккумулятора.

Согласно схемам на рис. 7–9 удвоенные частичные произведения 2p(1), 2p(2), 2p(3), 2p(4) и 2p(5) будут получены на 2-м, 6-м, 10-м и 14-м тактах сигнала cnt_sub_add.

Пятый оператор process, в списке чувствительности которого стоит сигнал cnt_sub_add, используется как дешифратор случаев 4 и 1, 2 и 3. На сигнале cnt_sub_add осуществляется подсчет синхроимпульсов.

Рассмотрим подробно случаи 1, 2 и 3: when "00010", "00110" и "01010". Приведем примеры для множителя X:

- X=*XXXX*0 (например, 11110 BIN или -2D);
- X=*XXX*00 (например, 11100 BIN или -4D, или же число со знаком –12 (10100 BIN));
- X=*XX*000 (11000 BIN или -8D),

где *X* — или «лог. 1», или «лог. 0».

В случае обнаружения этих чисел цифровой автомат сформирует два сигнала sign<='1' и sub_add<='0', для того чтобы в поле MSB (2p[5]) установился «лог. 0». В этих случаях умножение идет согласно принципу, показанному на рис. 8.

Существует еще случай when "10001", когда будет подсчитан 17-й такт синхроимпульса, при достижении которого формируется двоичное дополнение. Это распространяется на умножение двух четных отрицательных чисел. И когда оба числа оказываются отрицательными, одно из них может быть четное, а другое нечетное и наоборот (рис. 8б).

Сигнал sub_add используется для подачи «лог. 1» на вход переноса многоразрядного сумматора Cin масштабирующего аккумулятора, в случае обнаружения «1» в старшем разряде X(4)=1, а также для селективной инверсии числа A при переводе его в обратный код.

Когда будет подсчитан 18-й такт синхросигнала (when "10010"), цифровой автомат сформирует сигналы sign<='1' и sub_add<='0', и в поле MSB будет установлен «лог. 0».

Шестой оператор process реализует схему останова процесса умножения на суммирующем счетчике тактируемым срезом синхроимпульса. При достижении 21-го синхроимпульса вырабатывается сигнал останова Stop.

Приведем пример — код языка VHDL цифрового автомата умножителя:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
ENTITY avtomat IS
      PORT(
      Res,clk        : IN       STD_LOGIC;
      X,A: IN STD_LOGIC_VECTOR(4 downto 0);
      Ena_Add, LoadPSC,ena_shift,Stop,sub_add, sign: OUT  STD_
LOGIC;
      Q_sub_add  : OUT STD_LOGIC_VECTOR(4 downto 0);
      Q_stop        : OUT STD_LOGIC_VECTOR(4 downto 0));
END avtomat;
ARCHITECTURE a OF avtomat IS
TYPE state_values IS (SA, SB, SC, SD);
signal state,next_state: state_values;
SIGNAL cnt_sub_add: STD_LOGIC_VECTOR(4 downto 0);
SIGNAL cnt_stop: STD_LOGIC_VECTOR(4 downto 0);
BEGIN
statereg: process(clk,Res)
begin
   if (Res = '1') then state<=SA;
      elsif (clk'event and clk='1') then
                  state<=next_state;
   end if;
end process statereg;
process(state)
begin
case state is
when SA=> next_state<=SB;
when SB=> next_state<=SC;
when SC=>  next_state<=SD;
when SD=> next_state<=SA;
end case;
end process;
process (state)
begin
case state is
when SA=>Ena_Add<='0';
       LoadPSC<='0';
       ena_shift<='1';
when SB=>
      Ena_Add<='1';
      LoadPSC<='0';
      ena_shift<='0';
when SC=>
      Ena_Add<='0';
      LoadPSC<='0';
      ena_shift<='0';
when SD=>
      Ena_Add<='0';
      LoadPSC<='1';
      ena_shift<='1';
end case;
end process;
process (clk, res)
   begin
      if (res = '1') then cnt_sub_add <=(others=>'0');
      elsif (clk'event and clk = '0') then
      cnt_sub_add <= cnt_sub_add+'1';
      end if;
end process;
        Q_sub_add <= cnt_sub_add;
process(cnt_sub_add,X,A)
      begin
      -- Случай 4
      if A(4)='0' and X(4)='1' then
      sign<='1'; sub_add<='0';
      case cnt_sub_add is
      when "10001" =>
sign<='0'; sub_add<='1';
      when "10010" =>
      sign<='1'; sub_add<='1';
      when others => sign<='1'; sub_add<='0';
      end case;
      -- Для случаев 1, 2 и 3
         elsif A(4)='1' then
         case cnt_sub_add is
         when "00010" =>
         if (X(0)='0' ) then sign<='1'; sub_add<='0'; else
         sign<='0'; sub_add<='0'; end if;
         when "00110" =>
         if (X(1)='0' and X(0)='0') then sign<='1'; sub_add<='0';
else sign<='0'; sub_add<='0'; end if;
         when "01010" =>
         if (X(2)='0' and X(1)='0' and X(0)='0') then sign<='1';
sub_add<='0'; else sign<='0'; sub_add<='0'; end if;
         when "01110" =>
         if (X(3)='0' and X(2)='0' and X(1)='0' and X(0)='0') then
sign<='1'; sub_add<='0'; else sign<='0'; sub_add<='0'; end if;
         when "10001" =>
         --2S complement else no 2S complement
         if (X(4)='1') then sign<='1'; sub_add<='1'; else
sub_add<='0';sign<='0';end if;
         when "10010" =>
         ----2S complement and 2p[5]=0 else no 2S complement
and 2p[5]=1
         if (X(4)='1')
         or (X(4)='0' and X(3)='0' and X(2)='0' and X(1)='0'
and X(0)='0')
         then sign<='1'; sub_add<='0'; else sub_add<='0';sign<='0';
end if;
         when others => sign<='0';
         sub_add<='0';
         end case;
         end if;
         end process;
process (clk, res)
   begin
      if (res = '1') then cnt_stop <=(others=>'0');
      elsif (clk'event and clk = '1') then
      if cnt_stop = "10101" then Stop<='1';
      else cnt_stop <= cnt_stop+'1';
      end if;
      end if;
end process;
        Q_stop <= cnt_stop;
END a;
```

На рис. 11 показан верхний уровень иерархии проекта универсального умножителя двух чисел в дополнительном коде, как со знаком, так и без него, в САПР ПЛИС Quatus II Web Edition 13.0.1sp1 (сборка 232). В отличие от схемы умножения беззнаковых чисел, в эту схему введена дополнительная

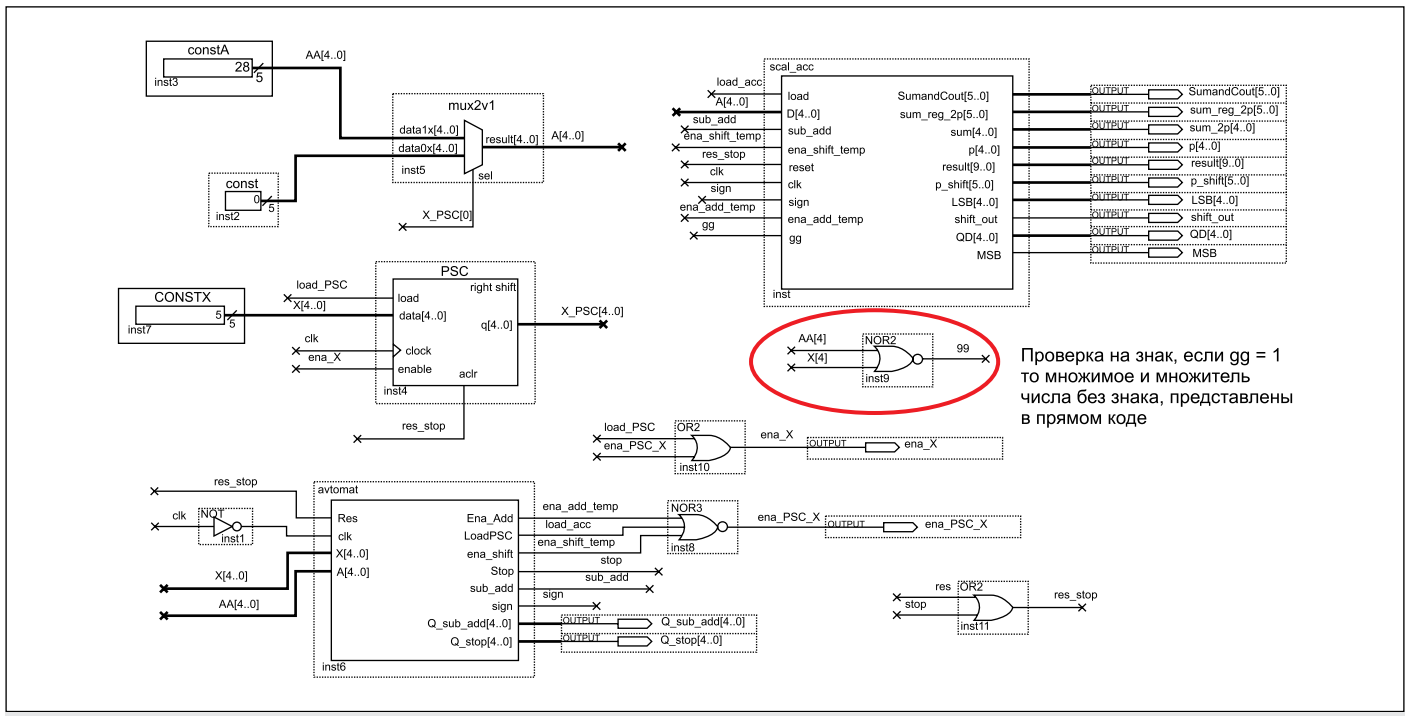

Рис. 11. Схема умножителя в САПР ПЛИС Quatus II (верхний уровень иерархии)

проверка на знак. Если сигнал gg=1, то множимое A и множитель X — беззнаковые числа. На рис. 12 черным, красным и зеленым цветом отмечена дополнительная логика, обеспечивающая умножение чисел как со знаком, так и без него. Блок complementer обеспечивает формирование обратного кода. Переключение между беззнаковыми и знаковыми числами осуществляет мультиплексор mux21 (выделен на рис. 12 зеленым цветом), на адресный вход которого подается сигнал gg. В целом принцип работы масштабирующего аккумулятора не отличается от описанного в работе [5], за исключением увеличения разрядности всех блоков на один бит.

Поле MSB (рис. 12) формируется с помощью двух элементов «исключающее ИЛИ» (XOR). Вспомогательная схема выделена на рис. 12 красным цветом. Один из входов элемента XOR (inst10) подключен к напряжению питания. Это необходимо для заполнения поля MSB «лог. 1» или «лог. 0», как требуется для случаев 1–4.

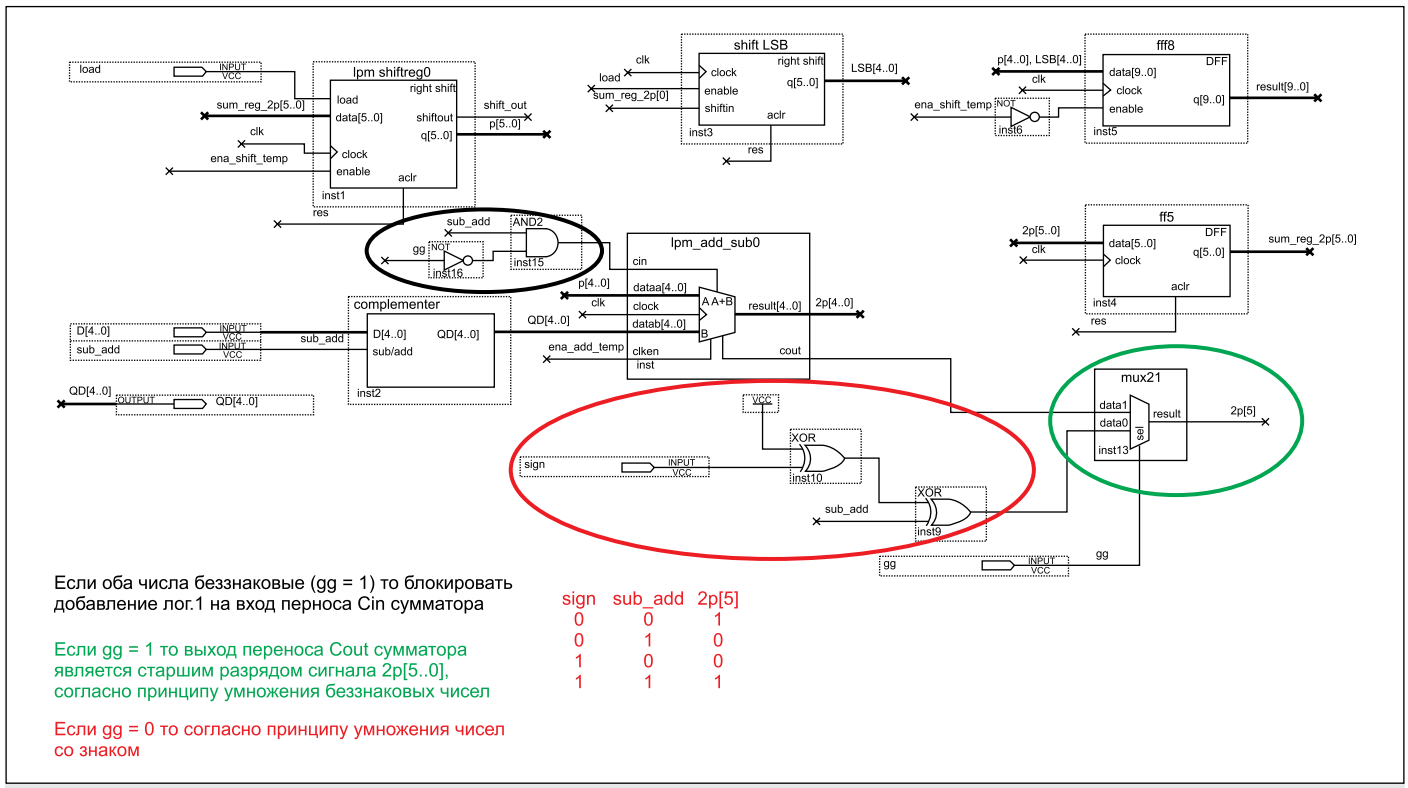

Рис. 12. Схема масштабирующего аккумулятора

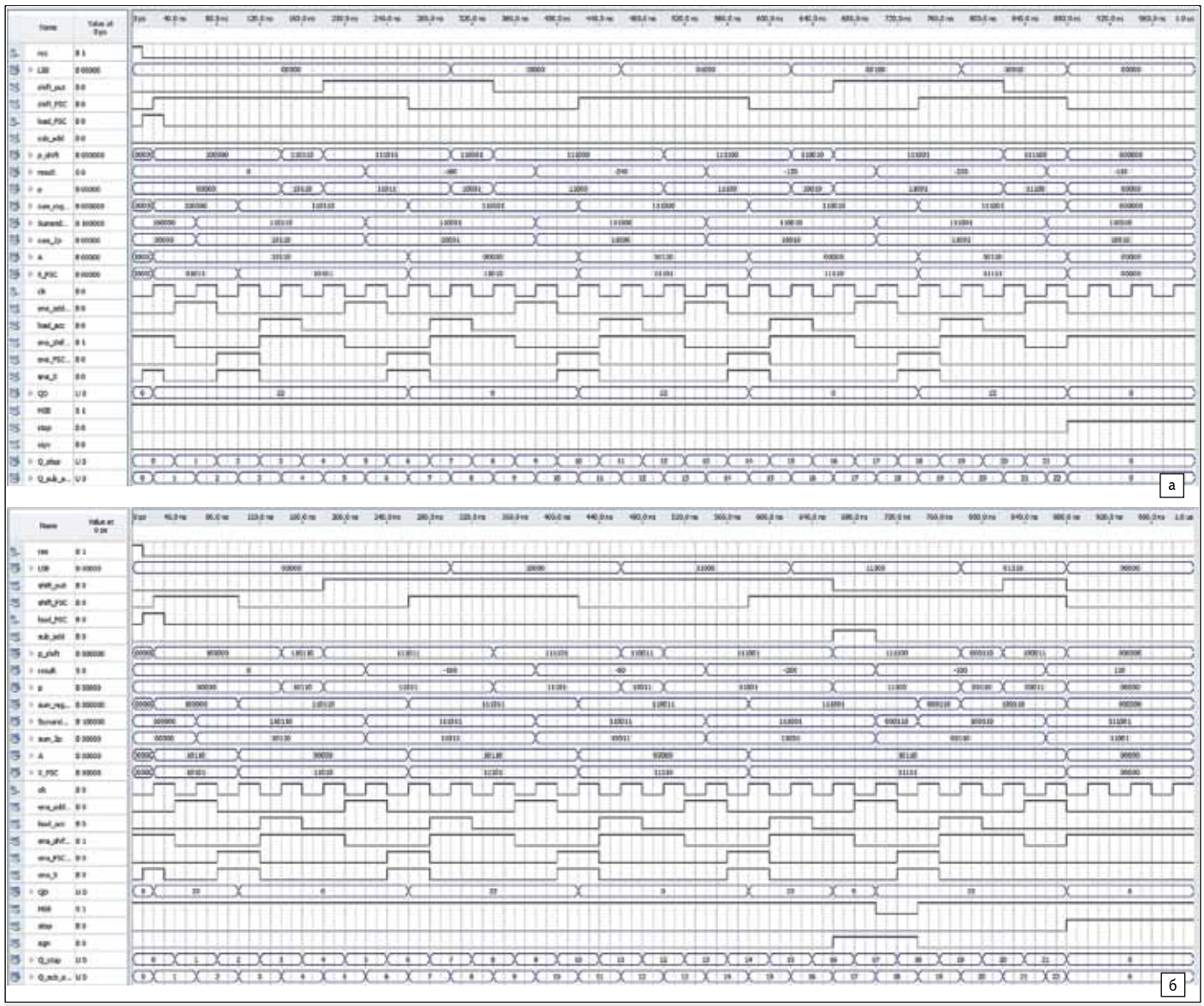

Рис. 13. Временные диаграммы процесса умножения: а) (−10)×11 (результат −110); б) (−10)×(−11) (результат 110)

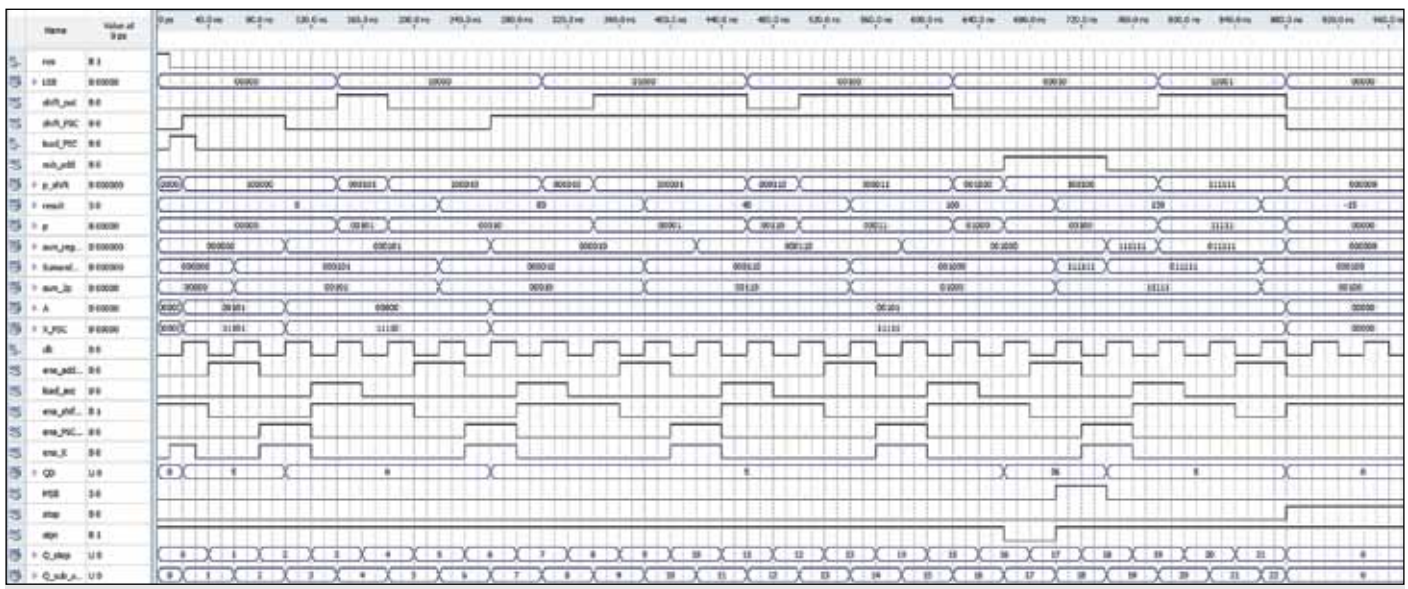

Рис. 14. Временные диаграммы процесса умножения 5×(−3) (результат −15)

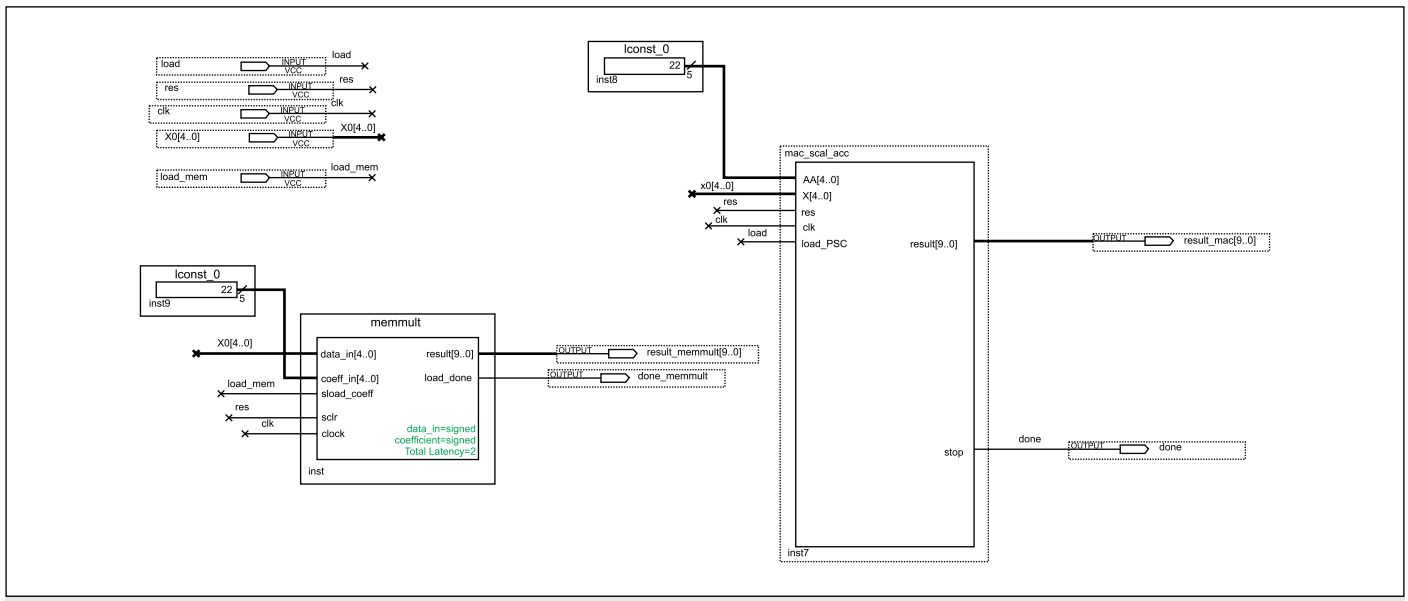

Рис. 15. Тестирование мегафункции ALTMEMMULT и разработанного MAC-блока

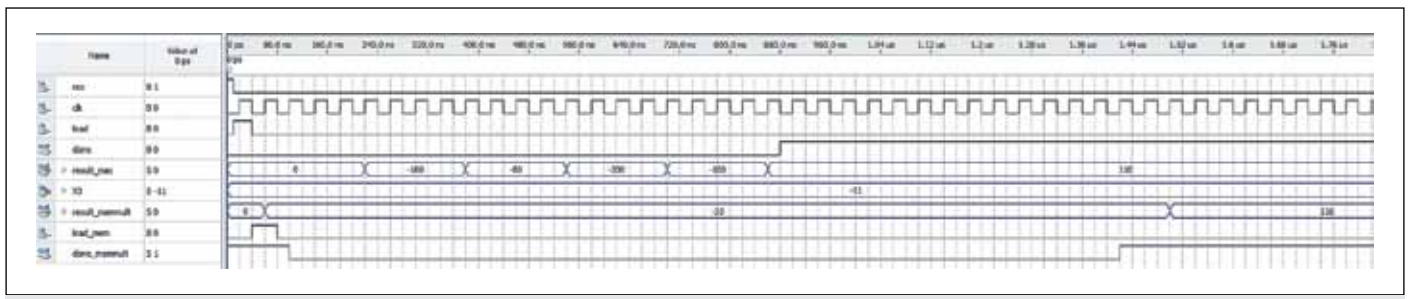

Рис. 16. Временные диаграммы процесса умножения (−10)×(−11) MAC-блоком и мегафункцией ALTMEMMULT (результат — 110)

На рис. 13 показаны случаи 1 и 2, а на рис. 14 — случай 4. В случае 1 в поле MSB в процессе умножения сохраняется «лог. 1». В случае 2 в поле MSB при вычислении удвоенного частичного произведения 2p(5) прописывается «лог. 0» по 18-му такту синхроимпульса. В случае 4 в поле MSB все происходит с точностью до наоборот.

На рис. 15 показаны тестовые схемы для умножения (−10)×(−11) для реализации в ПЛИС серии Cyclone II. На вход coeff_in[4..0] мегафункции ALTMEMMULT подключается константа (−10) (22D). Такое же значение подается и на вход AA[4..0] разработанного MAC-блока (рис. 11 и 12). На информационный вход data_in[4..0] мегафункции ALTMEMMULT и на вход X[4..0] MAC-блока подается число (−11).

На рис. 16 показан процесс умножения (−10)×(−11) MAC-блоком и мегафункцией для случая, когда константа загружается с внешнего порта. В мегафункцию предварительно загружена константа (режим загрузки из блочной памяти ПЛИС) — число 3.

Заключение

Разработан MAC-блок с использованием метода правого сдвига и сложения для умножения чисел, представленных в дополнительном коде, как четных, так и нечетных, со знаком и без него, для реализации в базисе ПЛИС. Предложенная схема реализации MAC-блока умножает за 21 такт синхрочастоты и может быть использована при проектировании КИХ-фильтров.

Литература

1. Computer Arithmetic: Algorithms and Hardware Designs. Oxford U. Press, 2010.
2. http://khadivi.iut.ac.ir/
3. Koren I. Digital Computer Arithmetic. ECE 666. Part 3. Sequential Algorithms for Multiplication and Division. University of Massachusetts Dept. of Electrical & Computer Engineering. http://euler.ecs.umass.edu/arith/parts-pdf/Part3-seq.pdf
4. Cheng X. Multiplication and Division Multiplication and Division. Csci 136. Computer Architecture II.
5. Строгонов А., Быстрицкий А. Проектирование умножителя методом правого сдвига и сложения с управляющим автоматом в базисе ПЛИС // Компоненты и технологии. 2013. № 12.