ФГБОУ ВО “Воронежский государственный
технический университет”

Кафедра полупроводниковой электроники
и наноэлектроники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ № 1-3
по дисциплине
«САПР системного уровня проектирования БИС»
для студентов направления 11.04.04 «Электроника и наноэлектроника» (направленность «Приборы и устройства в микро- и наноэлектронике») очной формы обучения

Воронеж 2016

Составители: д-р техн. наук А.В. Строгонов, канд. техн. наук Н.Н. Кошелева, канд. техн. наук А.В. Арсентьев, ассистент А.А. Винокуров

УДК 621.382

Методические указания к выполнению лабораторных работ № 1-3 по дисциплине «САПР системного уровня проектирования БИС» для студентов направления 11.04.04 «Электроника и наноэлектроника» (направленность «Приборы и устройства в микро- и наноэлектронике») очной формы обучения / ФГБОУ ВО «Воронежский государственный технический университет»; сост. А.В. Строгонов, Н.Н. Кошелева, А.В. Арсентьев, А.А. Винокуров. Воронеж, 2016. 34 с.

В методических указаниях приведены примеры по проектированию последовательных КИХ-фильтров с использованием языка Verilog и мегафункций САПР Quartus II для последующей реализации в базисе ПЛИС Altera.

Методические указания подготовлены в электронном виде и содержатся в файле МУ_послед_фильтры_2016.pdf.

Ил. 23. Библиогр.: 5 назв.

Рецензент д-р техн. наук, проф. М.И. Горлов

Ответственный за выпуск зав. кафедрой
д-р физ.-мат. наук, проф. С.И. Рембеза

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ЛАБОРАТОРНАЯ РАБОТА № 1

Проектирование последовательных КИХ-фильтров с использованием языка Verilog для последующей реализации в базисе ПЛИС

Цель работы: разработать функциональную модель последовательного КИХ-фильтра на четыре отвода в САПР Altera Quartus II с использованием языка Verilog.

Задание:

1) разработайте функциональный блок линии задержки на четыре отвода;

2) разработайте функциональный блок ПЗУ для хранения коэффициентов фильтра;

3) разработайте управляющий автомат КИХ-фильтра;

4) на основе полученных функциональных блоков на языке Verilog разработайте функциональную модель КИХ-фильтра и осуществите его моделирование;

5) сделайте выводы и оцените рабочую частоту проекта;

6) переработайте все блоки проекта с использованием языка VHDL, осуществите функциональное моделирование и убедитесь в правильности функционирования КИХ-фильтра;

7) разработайте управляющий автомат с использованием редактора состояний, осуществите функциональное моделирование и убедитесь в правильности функционирования КИХ-фильтра.

Пример выполнения лабораторной работы

На рис. 1 показана структура последовательного КИХ-фильтра на четыре отвода с использованием одного блока умножения и накопления (MAC-блока). КИХ-фильтры использующие в своей основе блоки умножения и накопления получили название в зарубежной литературе MAC FIR Filter.

Уравнение КИХ-фильтра на четыре отвода имеет вид: $y = C_0x_0 + C_1x_1 + C_2x_2 + C_3x_3$, где $C_0 = -2$, $C_1 = -1$, $C_2 = 7$ и $C_3 = 6$ коэффициенты фильтра.

Рассматриваемый проект последовательного КИХ-фильтра на четыре отвода представлен в САПР Quartus II Version 13.1. Для САПР Quartus II ver 9.0 его можно найти по адресу altera\90\qdesigns\fir_filter а для версии 13.0 - altera\13.1\quartus\qdesigns\fir_filter.

Линия задержки, ПЗУ для хранения коэффициентов, управляющий автомат разработаны на языке Verilog. Умножитель основан на мегафункции lpm_mult а аккумулятор представляет из себя иерархическое описание на языке verilog, нижний уровень которого - verilog-файл сгенерированный автоматически мегафункцией LPM_ADD_SUB. Умножитель представлен в виде символа, а линия задержки, коэффициенты и управляющий автомат в виде блок-схем.

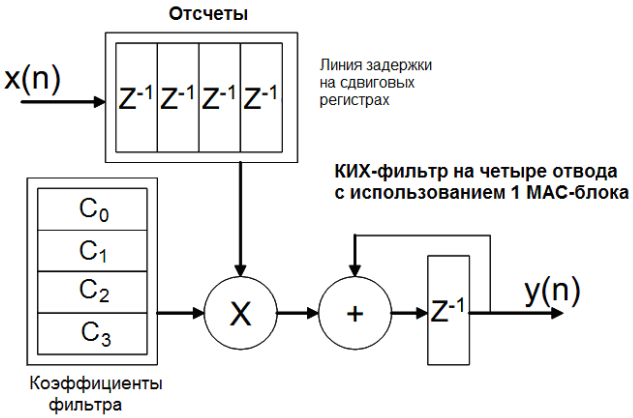


Рис. 1. КИХ-фильтр на четыре отвода с использованием одного MAC-блока

На рис. 2 показан иерархический проект последовательного КИХ-фильтра на четыре отвода в САПР Quartus II версии 13.0, а на рис. 3 структурная схема проекта, представленная на уровне регистровых передач, полученная с помощью RTL-viewer.

Входной сигнал и коэффициенты фильтра представлены в дополнительном коде с 8- и 4-битной точностью, поэтому умножитель и аккумулятор так же настроены для выполнения операций над числами со знаком.

Проект использует два синхросигнала clk и clkx2. Синхросигнал clk используется для тактирования линии задержки и внутреннего регистра аккумулятора фильтра, а clkx2 для тактирования работы регистра результата. Период синхросигнала clk в два раза больше clkx2. Умножитель и сумматор аккумулятора настроены для работы в асинхронных режимах.

Пример 1 демонстрирует verilog-код линии задержки на четыре отвода. На рис. 4 показана структурная схема линии задержки на базе регистров. Пример 2 демонстрирует verilog-код ПЗУ для хранения коэффициентов фильтра, а на рис. 5 показана структурная схема ПЗУ на базе дешифратора.

Пример 3 демонстрирует verilog-код управляющего автомата КИХ-фильтра. На рис. 6 показана структурная схема управляющего автомата (а); диаграмма состояний (б); таблица переходов и таблица кодирований состояний. Используется кодирование с одним активным или горячим состоянием (one hot encoding, HOT), т.е. в каждый конкретный момент времени активным (hot) может быть только один триггер состояния.

Метод ОНЕ применительно к ПЛИС FPGA, дает возможность строить конечные автоматы, которые в общем случае требуют меньших ресурсов и отличаются более высокими скоростными показателями, чем аналогичные конечные автоматы с двоичным кодированием состояний.

```
module taps
(       clk, reset, sel, newt, d, x);
        input clk;
        input reset;
        input [1:0] sel;
        input newt;
        input [7:0] d;
        output [7:0] x;
        reg [7:0] x, xn, xn_1, xn_2, xn_3;
// Register element
always @(posedge clk or posedge reset)
begin
if (reset)
        begin
                xn = 8'b00000000;
                xn_1 = 8'b00000000;
                xn_2 = 8'b00000000;
                xn_3 = 8'b00000000;
        end
else if (newt)
        begin
                xn_3 = xn_2;
                xn_2 = xn_1;
                xn_1 = xn;
                xn = d;
        end
end
// Mux element
always @(sel or xn or xn_1 or xn_2 or xn_3)
case (sel)
2'b 00: x = xn;
2'b 01: x = xn_1;
2'b 10: x = xn_2;
2'b 11: x = xn_3;
default: x = 8'bXXXXXXXX;
endcase
endmodule
```

Пример 1. Verilog-код линии задержки

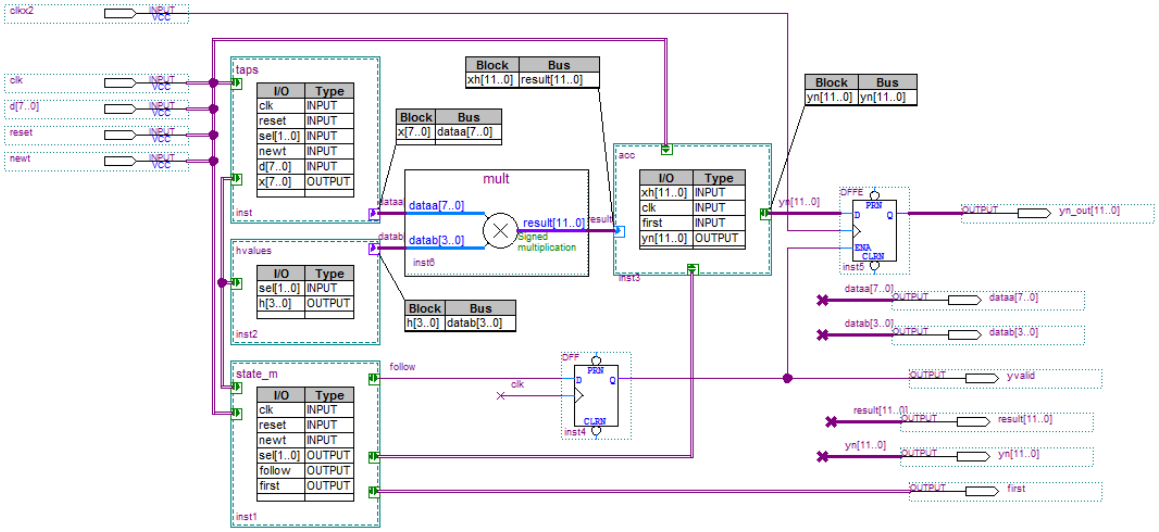


Рис. 2. Иерархический проект последовательного КИХ-фильтра на четыре отвода

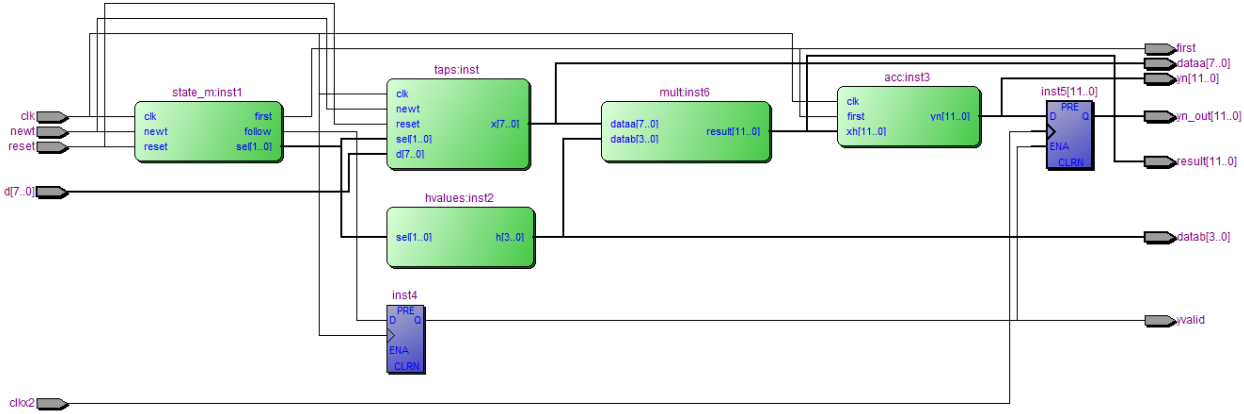


Рис. 3. Структурная схема проекта, полученная с помощью RTL-viewer

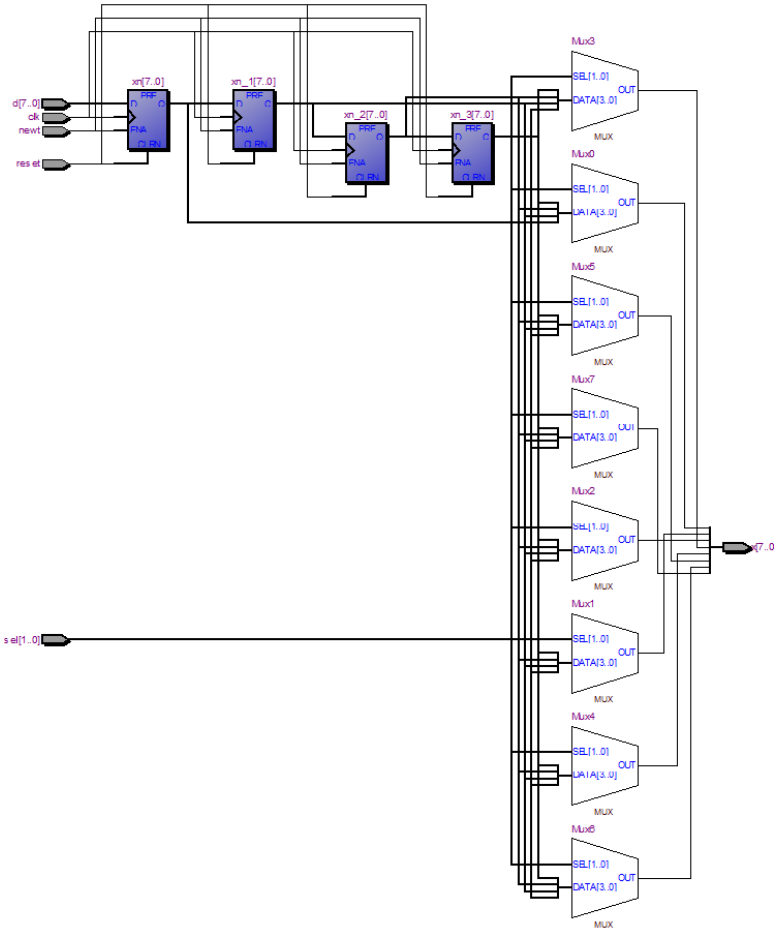


Рис. 4. Структурная схема линии задержки КИХ-фильтра

```
module hvalues
(
        sel, h
);
        input [1:0] sel;
        output [3:0] h;
```

```
reg [3:0]h;
always @(sel)
case (sel)
        2'b 00 : h = 4'b 1110;
        2'b 01 : h = 4'b 1111;
        2'b 10 : h = 4'b 0111;
        2'b 11 : h = 4'b 0110;
endcase
endmodule
```

Пример 2. Verilog-код ПЗУ для хранения коэффициентов фильтра

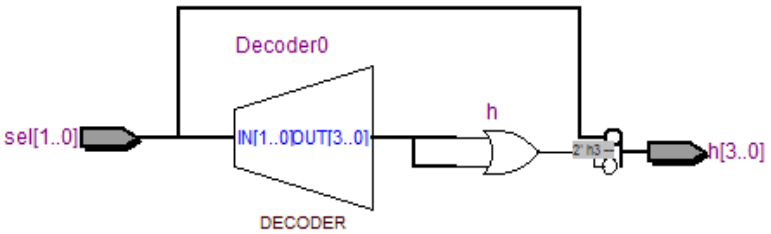


Рис. 5. Структурная схема ПЗУ

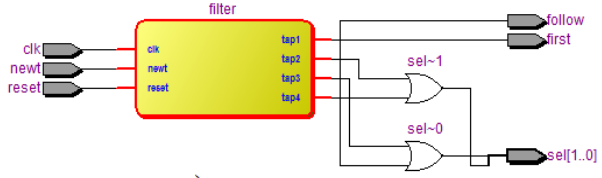
```
module state_m
(
        clk, reset, newt, sel, follow, first
);
        input clk, reset, newt;
        output follow, first;
        output [1:0]sel;
        reg follow, first;
        reg [1:0] sel;
        reg [2:0] filter;
parameter idle = 0, tap1 = 1, tap2 = 2, tap3 = 3, tap4 = 4;
always
begin
        case (filter)
                idle: begin
                                sel = 2'b0;
```

```
                        follow = 1'b0;
                        first = 1'b0;
                end
        tap1: begin
                        sel = 2'b0;
                        follow = 1'b0;
                        first = 1'b1;
                end
        tap2: begin
                        sel = 2'b01;
                        follow = 1'b0;
                        first = 1'b0;
                end
        tap3: begin
                        sel = 2'b10;
                        follow = 1'b0;
                        first = 1'b0;
                end
        tap4: begin
                        sel = 2'b11;
                        follow = 1'b1;
                        first = 1'b0;
                 end

        default : begin
                                sel = 2'b00;
                                follow = 1'b0;
                                first = 1'b0;
                                end
    endcase
end
always @(posedge clk or posedge reset)
begin
    if (reset)
        filter = idle;
    else
        case (filter)
            idle: begin
```

```
                        if (newt)
                                filter = tap1;
                         end

                tap1: begin
                                filter = tap2;
                         end
                tap2: begin
                                filter = tap3;
                         end
                tap3: begin
                                filter = tap4;
                         end
                tap4: begin
                                if (newt)
                                        filter = tap1;
                                else
                                        filter = idle;
                          end
        endcase
end
endmodule
```

Пример 3. Verilog-код управляющего автомата КИХ-фильтра

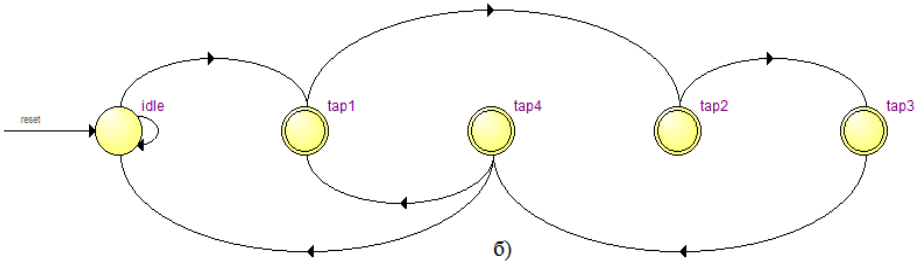
На рис. 7 показана структурная схема умножителя размерностью операндов 8x4 настроенная на асинхронный режим работы (clock=0). Умножитель проектируется на базе мегафункции LPM_MULT. Точность вычисления произведения 12-разрядов. Для реализации умножителя будем использовать аппаратные умножители ПЛИС. Всего аппаратных умножителей в ПЛИС EP3C5F256C6 с размерностью операндов 9x9 – 46 шт.



а)

	Source State	Destination State	Condition
1	idle	idle	(!newt)
2	idle	tap1	(newt)
3	tap1	tap2	
4	tap2	tap3	
5	tap3	tap4	
6	tap4	idle	(!newt)
7	tap4	tap1	(newt)

Transitions | Encoding



б)

	Name	tap3	tap2	tap1	idle	tap4
1	idle	0	0	0	0	0
2	tap1	0	0	1	1	0
3	tap2	0	1	0	1	0
4	tap3	1	0	0	1	0
5	tap4	0	0	0	1	1

в)

Рис. 6. Структурная схема управляющего автомата (а); диаграмма состояний (б); таблица переходов и таблица кодирований состояний

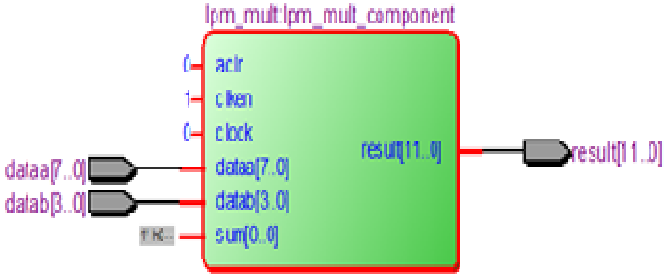


Рис. 7. Структурная схема умножителя размерностью операндов 8x4 на базе мегафункции LPM_MULT

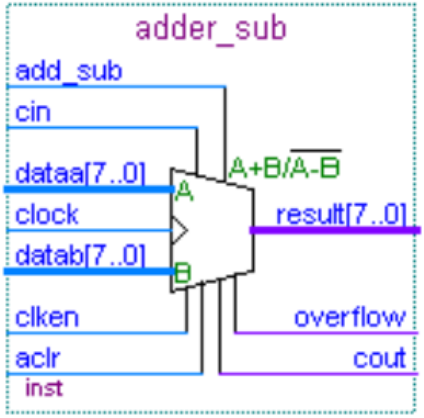


Рис. 8. Сумматор/вычитатель на базе мегафункции LPM_ADD_SUB

На рис. 8 показан сумматор/вычитатель на базе мегафункции LPM_ADD_SUB. Сумматор/вычитатель способен работать как в асинхронном, так и в синхронном режимах. По умолчанию, предполагается, что входные сигналы сумматора data[11..0] и datab[11..0] представлены в дополнительном коде.

Пример 4 демонстрирует Verilog-код аккумулятора КИХ-фильтра, а на рис. 9 показана структурная схема аккумулятора соответствующая этому коду. Аккумулятор (функциональный блок acc) выполнен на основе асинхронного накапливающего сумматора, на базе мегафункции LPM_ADD_SUB, регистра и мультиплексора для принудительной установки входа сумматора datab[11..0] в ноль. Пример 5 демонстрирует фрагмент verilog-кода функции используемой в примере 4.

На рис. 6 и 7 показано функциональное моделирование импульсной характеристики КИХ-фильтра на четыре отвода и прохождение сигнала по структуре фильтра. На вход фильтра поступает сигнал -5, 3, 1, 0, 0 и 0 т.д. Правильные значения на выходе фильтра: 10, -1, -40, -10, 25, 6 и 0 т.д.

```
module acc
(
        xh, clk, first, yn
);
        input [11:0] xh;
        input clk;
        input first;
        output [11:0] yn;

reg [11:0]yn;
reg [11:0] ynm, result, a_in;
wire [11:0] inter;
```

```
// Describe Multiplexer
always @(first or result)
begin
        case (first)
                1'b 0: ynm = result;
                1'b 1: ynm = 12'b00000000000;
        endcase
end
always @(posedge clk)
begin
        result = inter;
end
always @(xh)
begin
        a_in[11:0] = (xh);
        //a_in[12] = 0;
end
always @(result)
        begin
                yn[11:0] = result[11:0];
        end
accum  inst_13(.dataa(a_in), .datab(ynm), .result(inter));
endmodule
```

Пример 4. Verilog-код аккумулятора КИХ-фильтра

```
module accum (
        dataa,
        datab,
        result);

        input   [11:0]  dataa;
        input   [11:0]  datab;
        output  [11:0]  result;
```

```
        wire [11:0] sub_wire0;
        wire [11:0] result = sub_wire0[11:0];
        lpm_add_sub  lpm_add_sub_component (
                                .dataa (dataa),
                                .datab (datab),
                                .result (sub_wire0)
                                // synopsys translate_off
                                ,
                                .cout (),
                                .cin (),
                                .add_sub (),
                                .overflow (),
                                .clock (),
                                .clken (),
                                .aclr ()
                                // synopsys translate_on
                                );
        defparam
lpm_add_sub_component.lpm_direction = "ADD",
lpm_add_sub_component.lpm_hint =
"ONE_INPUT_IS_CONSTANT=NO,CIN_USED=NO",
lpm_add_sub_component.lpm_type = "LPM_ADD_SUB",
lpm_add_sub_component.lpm_width = 12;
    endmodule
```

Пример 5. Фрагмент verilog-кода функции accum (накапливающий сумматор) на базе мегафункции lpm_add_sub

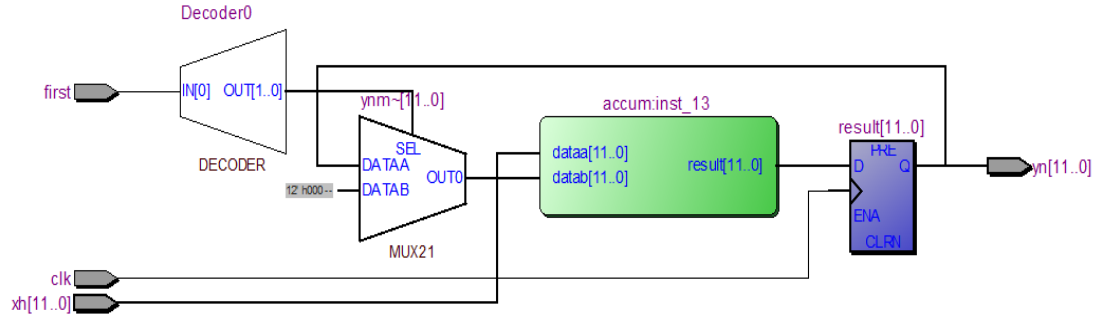


Рис. 9. Структурная схема аккумулятора на мегафункции LPM_ADD_SUB

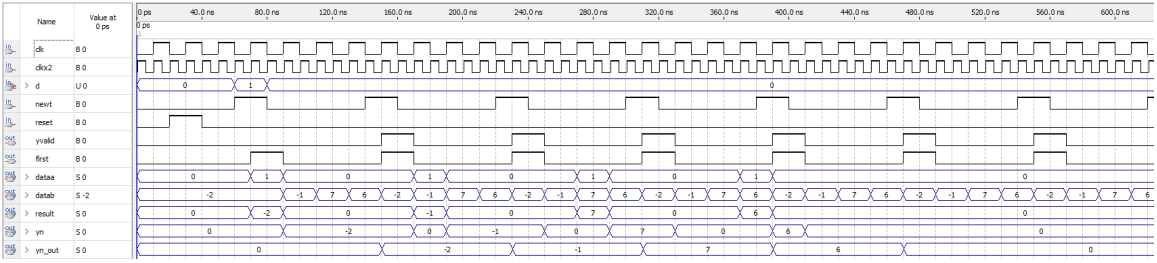


Рис. 10. Функциональное моделирование импульсной характеристики КИХ-фильтра на четыре отвода

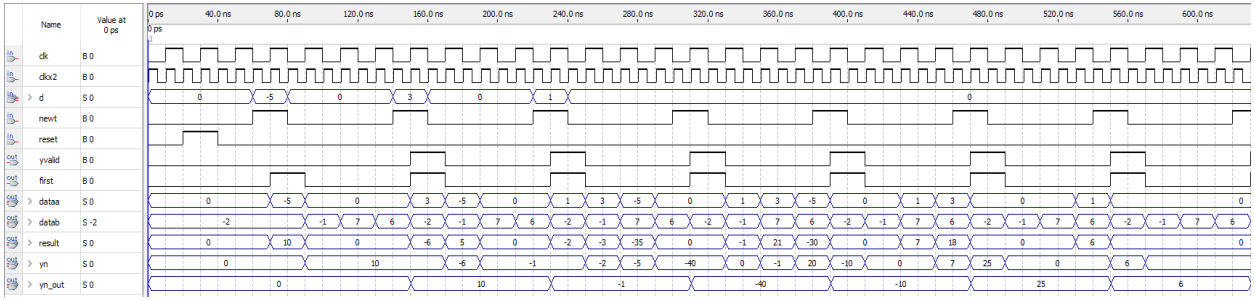


Рис. 11. Функциональное моделирование прохождения сигнала по структуре КИХ-фильтра на четыре отвода. На вход фильтра поступает сигнал -5, 3, 1, 0, 0 и 0 т.д. Правильные значения на выходе фильтра: 10, -1, -40, -10, 25, 6 и 0 т.д.

ЛАБОРАТОРНАЯ РАБОТА № 2

Проектирование последовательных КИХ-фильтров с использованием мегафункций для последующей реализации в базисе ПЛИС

Цель работы: разработать функциональную модель последовательного КИХ-фильтра на четыре отвода в САПР Altera Quartus II с использованием мегафункций для умножителя и аккумулятора.

Задание:

1) переработайте проект КИХ-фильтра для случая, когда умножитель и аккумулятор синхронизуются общим синхросигналом clk и реализуются с использованием мегафункци умножения LPM_MULT и мегафункции накопления ALTACCUMULATE;

2) переработайте проект КИХ-фильтра для случая, когда умножитель и аккумулятор реализованы на мегафункции умножения и накопления ALTMULT_ACCUM;

3) для различных модификаций базового проекта осуществите учет латентности функциональных блоков КИХ-фильтра;

4) дайте оценку быстродействия проектов и оцените количество задействованных логических ресурсов ПЛИС.

Пример выполнения лабораторной работы

Первый вариант модификации проекта. Рассмотрим вариант, когда умножитель и аккумулятор синхронизуются общим синхросигналом clk но реализуются с использованием мегафункции умножения LPM_MULT и накопления ALTACCUMULATE. Мегафункция ALTACCUMULATE не входит в разработчик мегафункций (MegaWizard Plug-In Manager) для САПР Quartus II версии 13.1 и старше. Реализуется на базе логических элементов для ПЛИС серии Cyclone III и на базе адаптивных логических элементов для ПЛИС серии Stratix III.

Входной сигнал и коэффициенты фильтра представим с 8-разрядной точностью. Все функциональные блоки проекта

на верхнем уровне иерархии представим символами. Линия задержки, ПЗУ и управляющий автомат представлены verilog-кодом и корректировки не подлежат. На рис. 12 показан проект последовательного КИХ-фильтра на четыре отвода с использованием аккумулятора на мегафункции ALTACCUMULATE а на рис. 13 демонстрируется функциональное моделирование прохождения сигнала по структуре КИХ-фильтра. Латентность умножителя и аккумулятора составляют 1 такт синхрочастоты.

Для согласования работы управляющего автомата и аккумулятора необходимо выходной сигнал first автомата задержать на один такт синхрочастоты с помощью двухтактного триггера выход которого необходимо подключить ко входу сигнала синхронной загрузки аккумулятора sload. Результат умножения и вычисления суммы произведений представляются с 16-разрядной точностью. Для получения правильных значений профильтрованного сигнала на выходе yn_out[15..0] необходимо выходной сигнал follow задержать на три такта синхрочастоты и подключить его ко входу разрешения тактирования ena регистра результата.

Второй вариант модификации проекта. Рассмотрим вариант, когда умножитель и аккумулятор реализованы на мегафункции умножения и накопления ALTMULT_ACCUM (рис. 14 и рис. 15). Данный вариант является предпочтительным для современных ПЛИС, так как предполагает использование встроенных ЦОС-блоков ПЛИС позволяя экономить логические ресурсы. Выходной сигнал автомата first необходимо подключить ко входу синхронной загрузки аккумулятора accum_sload на прямую. Последовательный КИХ-фильтр на мегафункции ALTMULT_ACCUM имеет латентность пять тактов синхрочастоты сигнала clk и результат фильтрации обновляется через каждый четвертый такт синхрочастоты (на пятый, рис. 15).

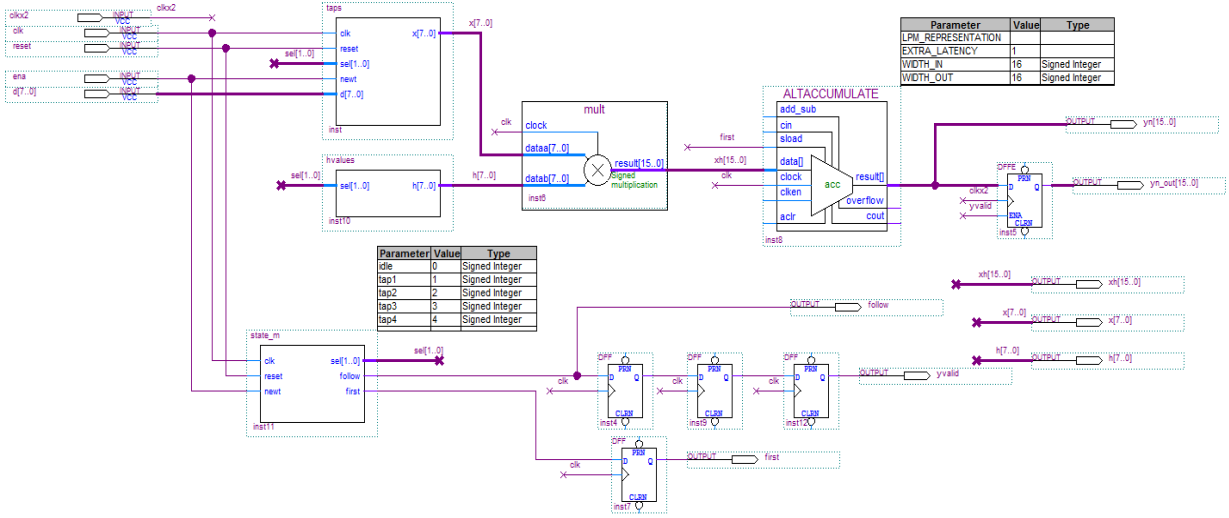


Рис. 12. Проект последовательного КИХ-фильтра на четыре отвода с использованием аккумулятора на мегафункции ALTACCUMULATE

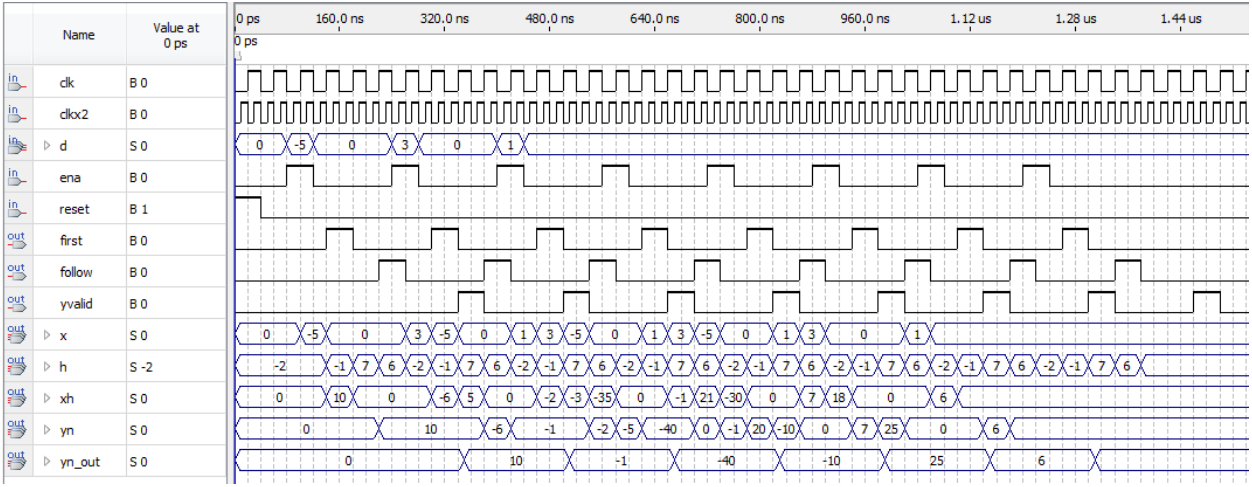


Рис. 13. Функциональное моделирование прохождения сигнала по структуре КИХ-фильтра на четыре отвода в случае использования аккумулятора на мегафункции ALTACCUMULATE

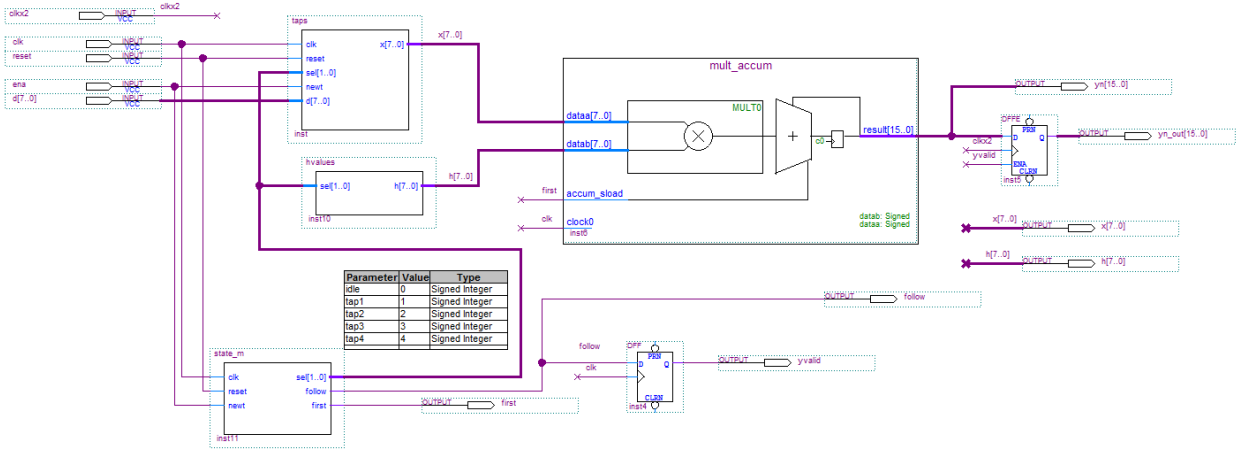


Рис. 14. Проект последовательного КИХ-фильтра на четыре отвода с использованием умножителя и аккумулятора на мегафункции ALTMULT_ACCUM

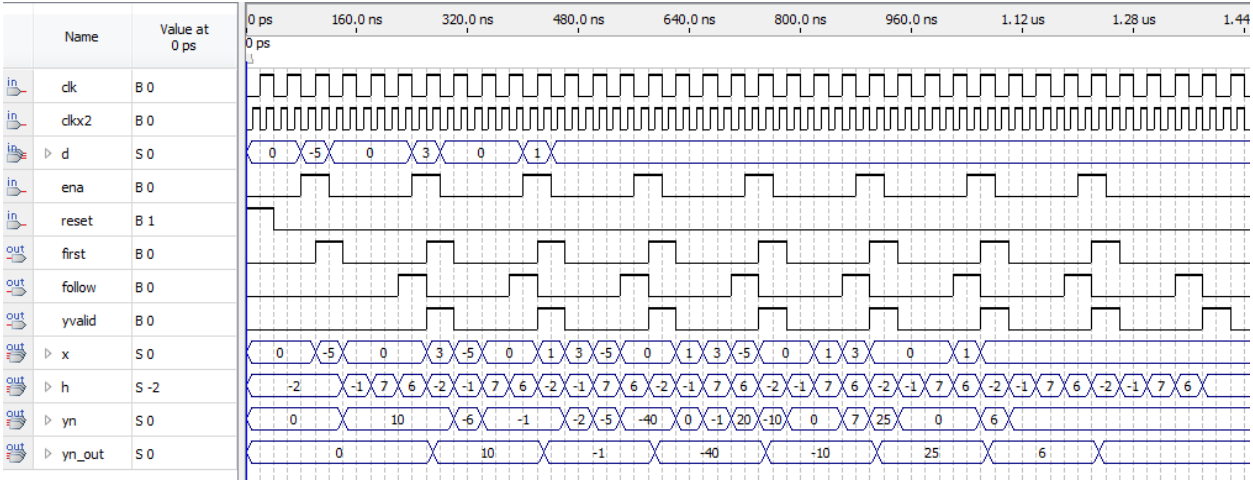


Рис. 15. Функциональное моделирование прохождения сигнала по структуре КИХ-фильтра на четыре отвода в случае использования умножителя и аккумулятора на мегафункции ALTMULT_ACCUM

ЛАБОРАТОРНАЯ РАБОТА № 3

Проектирование последовательных КИХ-фильтров с использованием линии задержки на основе двух портовой памяти ПЛИС

Цель работы: разработать функциональную модель последовательного КИХ-фильтра на четыре отвода в САПР Altera Quartus II с использованием линии задержки на основе двух портовой памяти ПЛИС.

Задание:

1) разработайте линию задержки на основе мегафункции RAM: 2-PORT. Установите режим Old memory contents appear для выходной шины q;

2) в качестве управляющего автомата используйте суммирующий счетчик;

3) для хранения коэффициентов фильтра используйте мегафункцию ROM: 1 port;

4) для различных модификаций базового проекта осуществите учет латентности функциональных блоков КИХ-фильтра;

5) осуществите функциональной моделирование и убедитесь в правильности работы КИХ_фильтра;

6) переработайте проект для случая когда линия задержки реализована на основе двух портовой памяти (мегафункция Shift Register (RAM-based)).

7) дайте оценку быстродействия проектов и оцените количество задействованных логических ресурсов ПЛИС

Пример выполнения лабораторной работы

Рассматриваемый вариант реализации проекта аналогичен реализации КИХ-фильтра на ЦОС-процессоре с использованием циклических буферов для хранения отсчетов

сигнала. Для реализации циклического буфера на базе сдвигового регистра может использоваться фиксированный объем оперативной памяти. В качестве оперативной памяти может быть использована блочная память ПЛИС в режиме ОЗУ.

На рис. 16 показана структурная схема КИХ-фильтра с использованием двух портовой памяти ПЛИС. Двух портовая память логически разбивается на два банка памяти, один из которых работает как циклический буфер для считывания и записи входных отсчетов сигнала подлежащего фильтрации, а второй для хранения коэффициентов фильтра (выполняет функцию ПЗУ). Для упрощения проекта реализуем только линию задержки на основе двух портовой памяти ПЛИС. Это позволит использовать двухразрядный счетчик в качестве простейшего управляющего автомата фильтра.

Увеличим разрядность шины данных ОЗУ с четырех до восьми бит. Для формирования лог. единицы используем сигнал cout двухразрядного счетчика пропустив его через два триггера. Он будет сигнализировать о том, что коэффициенты фильтра считаны из ПЗУ (мегафункция ROM: 1 port). Аккумулятор за это время должен вычислить “правильную” сумму произведений и новые данные могут быть записаны в память. На рис. 17 показана функциональная модель последовательного КИХ-фильтра на четыре отвода с использованием линии задержки на основе двух портовой памяти. Процессы считывания и записи информации в память разделены во времени элементом задержки (2-разрядным регистром). В структуре фильтра предусмотрена обратная связь которая заключается в подачи данных с выходной шины ОЗУ на вход через шинный мультиплексор на другой вход которого подается сигнал, подлежащий фильтрации.

Выходные шины данных ОЗУ и ПЗУ должны быть регистерные. Это обеспечивает правильность функционирования аккумулятора.

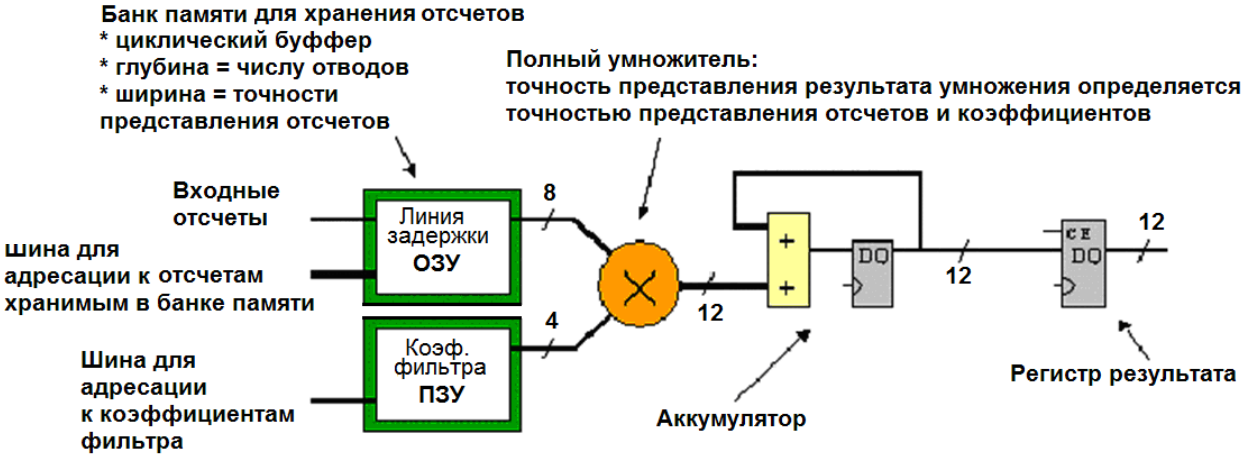


Рис. 16. Структурная схема КИХ-фильтра с использованием двух портовой памяти ПЛИС

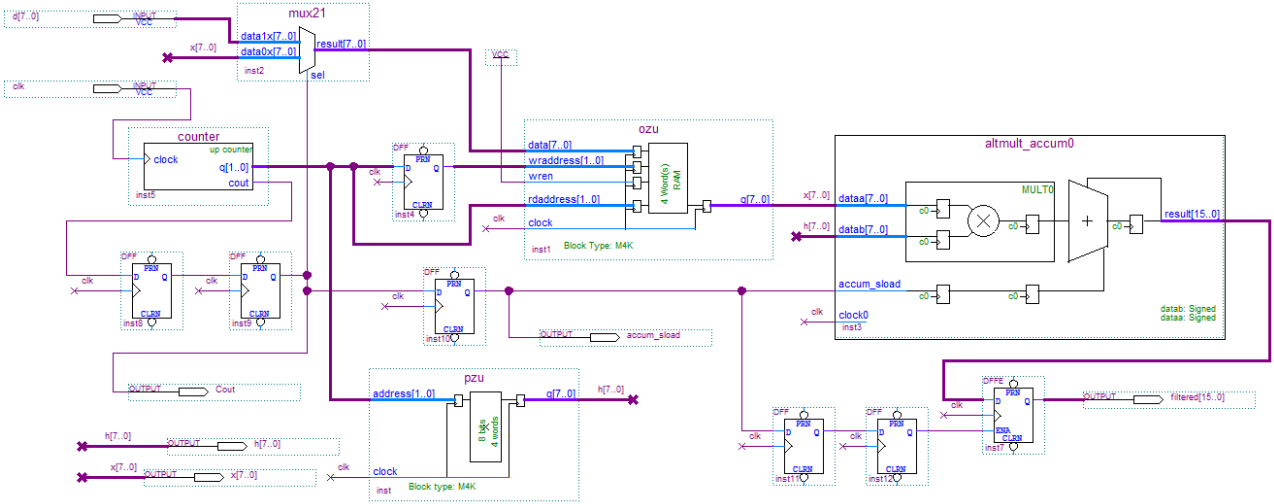


Рис. 17. Функциональная модель последовательного КИХ-фильтра на четыре отвода с использованием линии задержки на основе двух портовой памяти

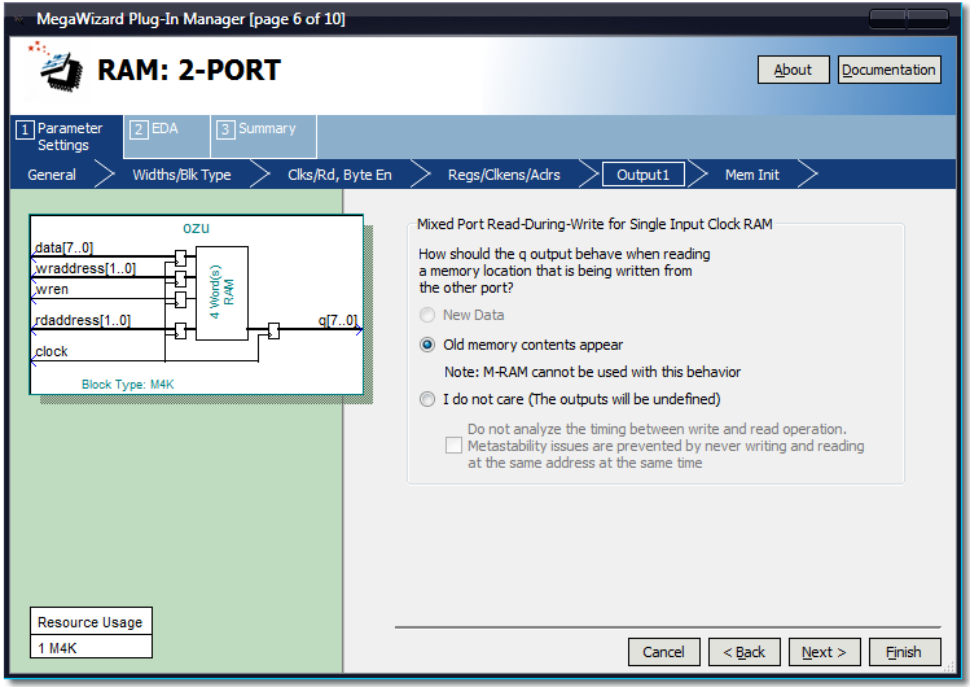


Рис. 18. Мегафункция RAM: 2-PORT. Режим Old memory contents appear для выходной шины q

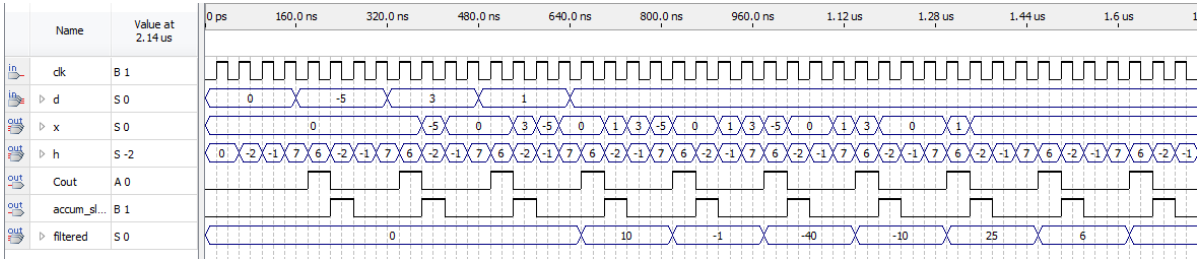


Рис. 19. Функциональное моделирование прохождения сигнала по структуре КИХ-фильтра на четыре отвода. Режим Old memory contents appear

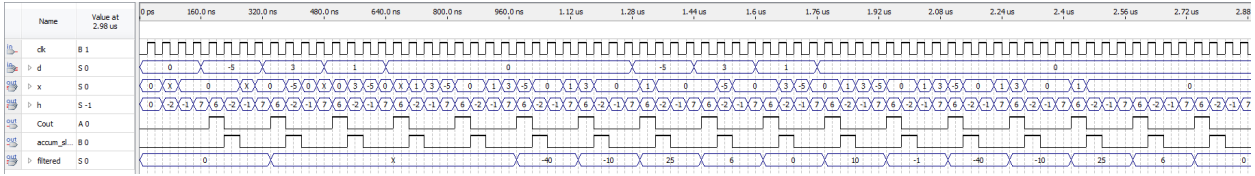


Рис. 20. Функциональное моделирование прохождения сигнала по структуре КИХ-фильтра на четыре отвода. Режим I do not care

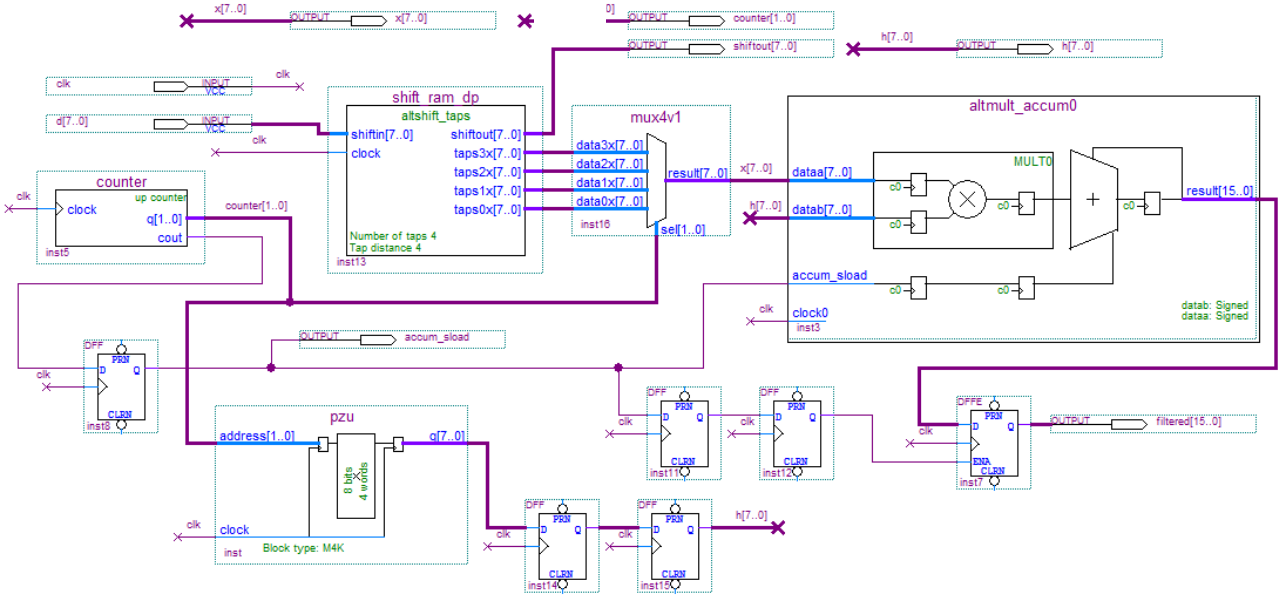


Рис. 21. Линия задержки на основе двух портовой памяти (мегафункция Shift Register (RAM-based))

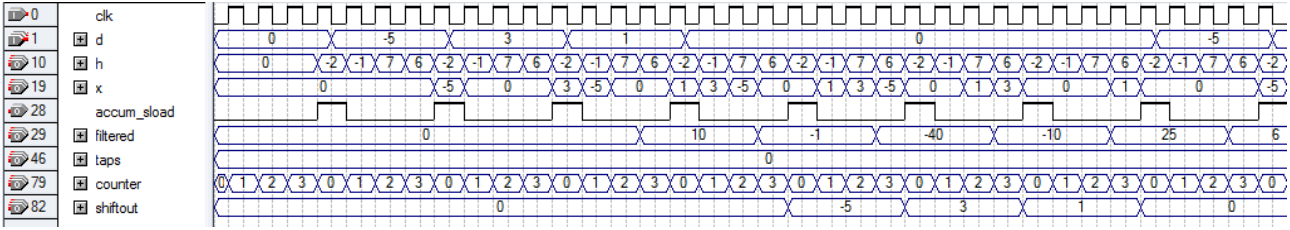


Рис. 22. Функциональное моделирование прохождения сигнала по структуре КИХ-фильтра (линия задержки на основе мегафункции Shift Register (RAM-based))

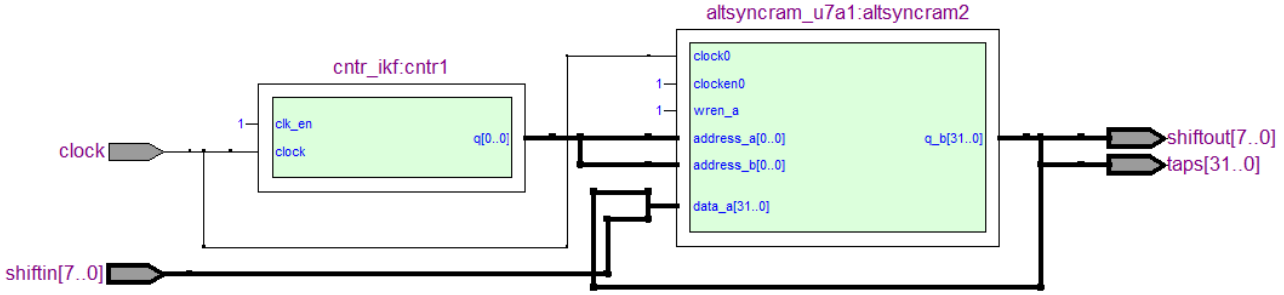


Рис. 23. RTL-представление линии задержки на основе мегафункции Shift Register (RAM-based)

Блоки памяти ПЛИС фирмы Altera для отображения значений на выходной шине q в случае одновременного чтения и записи по одинаковому адресу могут работать в трех режимах new_data, old_data и don't_care. При выборе режима old_data на выходной шине q отображаются старые данные, хранящиеся в ОЗУ по конкретному адресу прежде, чем новые данные будут записаны, поэтому же адресу в память. У блоков памяти Xilinx этот режим известен как read_first.

В режиме don't_care при одновременном чтении и записи по одинаковому адресу новые данные записываются в память, а на выходной шине q считываемые значения отображаются ввиде символа 'x' (unknown values). Такой режим для блоков памяти Xilinx не доступен.

Режим don't_care не является критическим для проекта. В режиме new_data новые данные записываются в память и одновременно отображаются на выходе блока памяти. У Xilinx это режим write_first. Выбор режимов new_data, old_data и don't_care зависит от типа блока памяти TriMatrix. В данном проекте будем использовать блоки памяти типа M4K. Для ПЛИС серий Cyclone II и Stratix II эти блоки функционально эквивалентны.

На рис. 18 показано, что мегафункция RAM: 2-PORT для выходных значений на шине q настроена на режим Old memory contents appear. На рис. 19 представлено функциональное моделирование прохождения сигнала по структуре КИХ-фильтра в режиме Old memory contents appear.

Рис. 20 показывает функциональное моделирование прохождения сигнала по структуре КИХ-фильтра для режима I do not care. Сравнивая рис. 20 и рис. 19 видим, что при подачи сигнала -5, 3, 1, 0, 0 и 0 т.д. на вход фильтра, в случае если двух портовая память работает в режиме I do not care, на выходе формируются не требуемые значения 10, -1, -40, -10, 26, 6 и 0 т.д. а значения -40, -10, 26, 6 и 0 т.д., т.е. значения 10, -1 пропущены. Тем не менее, фильтр после 980 нс начинает формировать правильные значения. До этого момента времени выход фильтра не определен.

Линию задержки можно так же реализовать на основе двух портовой памяти с использованием мегафункции altshift_taps (Shift Register (RAM-based)) (рис. 21) со следующими установками: число отводов – 4; дистанция между отводами – 4. Коммутация отводов линии задержки осуществляется с помощью шинного мультиплексора четыре в один на адресный вход которого подключается выход счетчика counter. Для согласования работы линии задержки и ПЗУ используемого для хранения коэффициентов фильтра с блоком умножения и накопления на основе мегафункции ALTMULT_ACCUM к выходу ПЗУ требуется дополнительно подключить два 8-разрядных регистра.

Для выходного порта q используется только режим old. В мегафункции altshift_tap не предоставляется изменение режимов работы выходного порта при одновременном чтении и записи по одинаковому адресу как для мегафункции RAM: 2-PORT. Результаты функционального моделирования показаны на рис. 22. На рис. 23 показано RTL-представление линии задержки на основе счетчика и блока памяти.

Использование двух портовой памяти в качестве линии задержки настроенной на режим Old memory contents appear для выходного порта q позволяет получить наивысшее быстродействие для последовательного КИХ-фильтра.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Uwe Meyer-Baese. Digital Signal Processing with Field Programmable Gate Arrays. Fourth Edition. http://www.springer.com/series/4748.

2. Xilinx. DSP: Designing for Optimal Results High-Performance DSP Using Virtex-4 FPGAs. DSP Products Advanced Design Guide. Edition 1.0. March 2005.

3. Строгонов А.В. Проектирование последовательных КИХ-фильтров в системе визуально-имитационного моделирования Matlab/Simulink с использованием Altera DSP Builder. [Текст] / А.В. Строгонов, С.А. Цыбин, П.С. Городков // Компоненты и технологии. – 2015. - N11. - С.22-27.

4. Строгонов А.В. Проектирование последовательных КИХ-фильтров в САПР ПЛИС Quartus II [Текст] / А.В. Строгонов, С.А. Цыбин, П.С. Городков // Компоненты и технологии. – 2016. - N1. - С.10-15.

5. Строгонов А.В. Особенности использования двух портовой памяти при проектировании последовательных КИХ-фильтров в САПР ПЛИС Quartus II [Текст] / А.В. Строгонов, С.А. Цыбин, П.С. Городков // Компоненты и технологии. - 2016. - N4. - С.40-46.

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ № 1-3
по дисциплине
«САПР системного уровня проектирования БИС»
для студентов по направлению подготовки 11.04.04
«Электроника и наноэлектроника» (направленность «Приборы и устройства в микро- и наноэлектронике»)
очной формы обучения

Составители:
Строгонов Андрей Владимирович
Кошелева Наталья Николаевна
Арсентьев Алексей Владимирович
Винокуров Александр Александрович

В авторской редакции

Компьютерный набор А.В. Строгонова

Подписано к изданию 26.10.2016.
Уч.-изд. л. 2,1.

ФГБОУ ВО “Воронежский государственный технический университет”
394026 Воронеж, Московский просп., 14