Summery

## چکیده:

در این پروژه می خواهیم فیلتر FIR را با الگوریتم های مختلفی بسازیم و کار با Matlab HDL یاد بگیریم

## مقدمه:

در این پروژه از Matlab HDL Coder و Modelsim استفاده شده است که ما ابتدا فیلتری را به صورت دستی با زبان Verilog ساختیم سپس همان فیلتر را با Matlab ساختیم و کد Verilog را با ان تولید کردیم این کار را یک بار با الگریتم CSD و یک بار به صورت DA ساخته ایم

## ابزار ها:

در این پروژه از Modelsim و Matlab و Quartes استفاده شده است و مدل FPGA را Cyclone II انتخاب کرده ایم

## قسمت1:

1) ابتدا فیلتر FIR را که در زیر نیز امده است به صورت دستی نوشته ایم. در این فیلتر ضرایب 480و-320و31 است که انها را به صورت CSD در نظر گرفته ایم

```
  f0 fil0 (
    .A(X),
    .Out(f00)
);
  f1 fil1 (
    .A(reg10),
    .Out(f10)
);
  f2 fil2 (
    .A(reg20),
    .Out(f20)
);

adder Add1 (
    .A(f00),
    .B(f10),
    .Out(A10)
);
adder A2dd(
    .A(A10),
    .B(f20),
    .Out(A20)
);

assign Out = A20;

endmodule
```

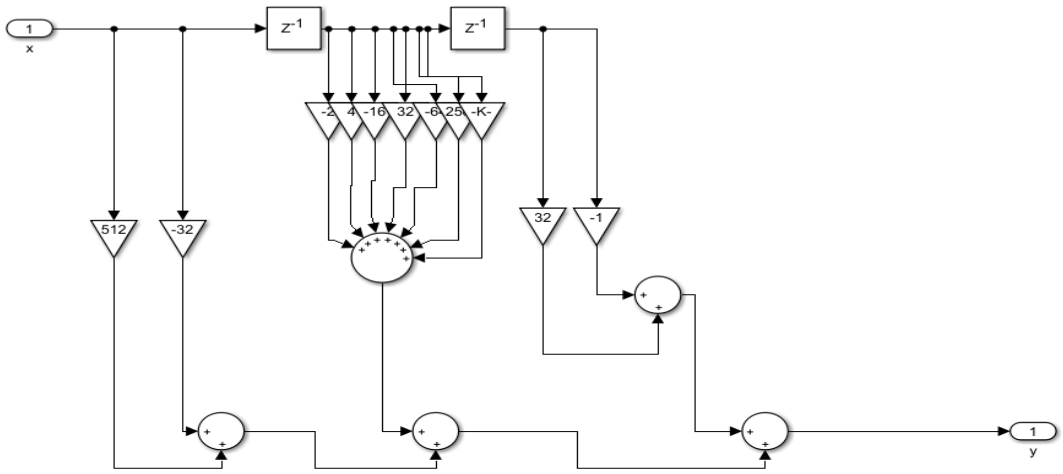
```
always @(posedge clk, posedge rst) begin
  if(rst)
    regEn = 0;
  else
    regEn = 1;

end
reg16 R1 (
  .clk(clk),
  .rst(rst),
  .regEn(regEn),
  .regIn(X),
  .regOut(reg10)
);
reg16 R2 (
  .clk(clk),
  .rst(rst),
  .regEn(regEn),
  .regIn(reg10),
  .regOut(reg20)
);
```

کد کامل در فایل قرار دهده شده است و ضرایب به صورت CSD هستند

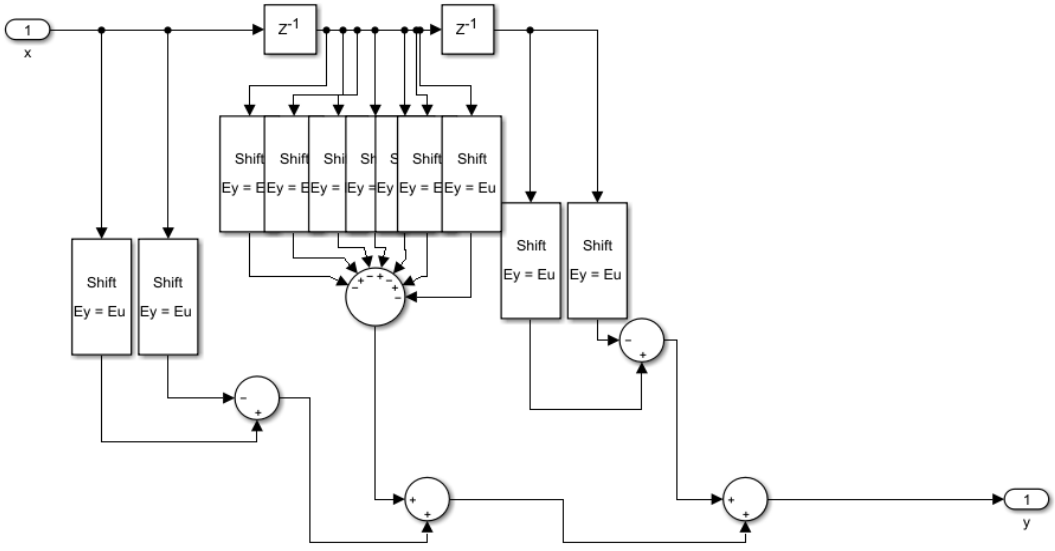
Summery

2) در این قسمت فیلتر را به کمک بخش Simulink Matlab و با استفاده از بلاک Gain ساخته ایم و ضرایب هم چنان به صورت CSD هستند




همان طور که در شکل مشخص است از بلاک های Gain , Sum , Delay استفاده شده است
سپس به کمک Matlab یک Subsystem از شکل ساخته و برای سیستم مورد نظر کد Verilog و Testbench انرا ساخته ایم که کد ها در فایل موجود اند

3) در این قسمت همتن فیلتر قسمت 2 را بجای استفاده از Gain از بلاک Shift استفاده می کنیم و دوباره مانند قبل کر Verilog و Testbench ان را می سازیم




کد ها در فایل اورده شده ان.

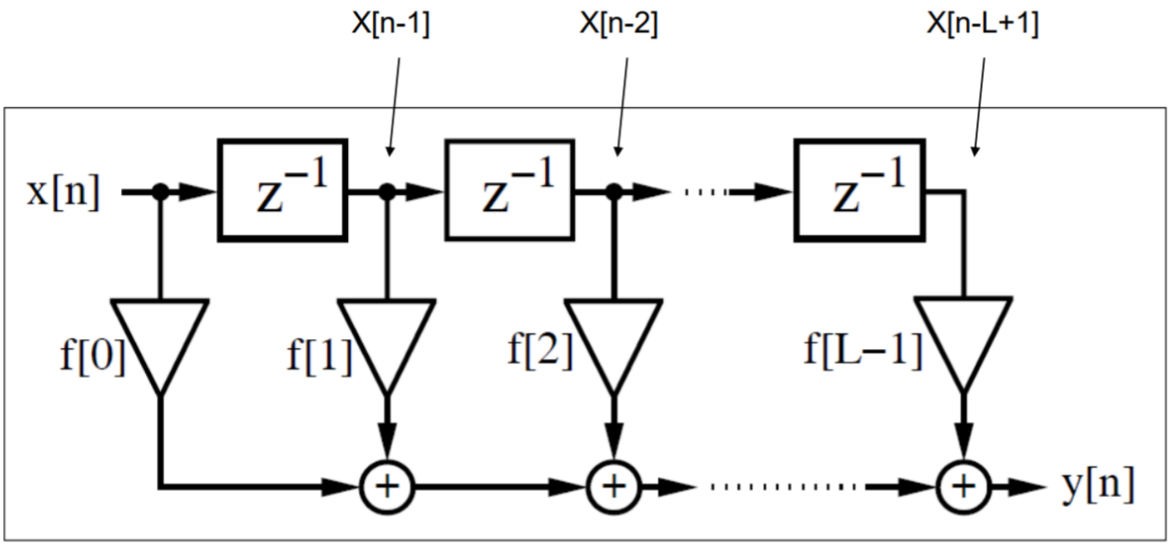
Summery

4) با استفاده از کد های Testbench تولید شده فیلتر های ساخته شده را امتحان می کنیم و می بینیم که درست کار می کنند

5) حال کد های Verilog را به Quartes برده و انها را synthesis می کنیم و داده های زیر به دست می ایند:

Handwrite_FIR_CSD☺ Total Logic Elements: 135, Total Registers: 33;
Matlab_Gain_FIR☺ Total Logic Elements: 208, Total Registers: 102;
Matlab_Shift_FIR☺ Total Logic Elements: 133, Total Registers: 32;

6) تفاوت کد های Gain , Shift در ان است که در Gain کد تولید شده به ازای هر Gain Block یک رجیستر استفاده شده و تمام مقادیر ثابت ان رجیستر شده اند ولی در Shift این مقادیر با استفاده از Assign بدست امده ان و دیگر رجیستر نیستند به امین دلیل تعداد رجیستر و LE کمتری استفاده شده است.




$$y[n] = x[n] * f[n] = \sum_{k=0}^{L-1} f[k]x[n-k]$$

## قسمت2:

1) در این قسمت فیلتر FIR را به صورت دستی با کد Verilog و با الگوریتم DA ساخته ایم که در ان از LUT استفاده شده است و ضراب ان 3,12,12,3 هستند

```
else begin
p <= (p >> 1) + (table_out << 2); // p/2+table*4
x0[0] <= x0[1];
x0[1] <= x0[2];
x0[2] <= x0[3];
x1[0] <= x1[1];
x1[1] <= x1[2];
x1[2] <= x1[3];
x2[0] <= x2[1];
x2[1] <= x2[2];
x2[2] <= x2[3];
count = count + 1;
state <= s1;
end
end
endcase
end
case3 LC_Table0( .table_in(table_in), .table_out(table_out));
assign lut = table_out; // Provide test signal
endmodule
```

```
assign table_in[0] = x0[0];
assign table_in[1] = x1[0];
assign table_in[2] = x2[0];
assign table_in[3] = x3[0];

always @(posedge clk or posedge reset)
begin : DA //----> DA in behavioral style

parameter s0=0, s1=1;

reg [0:0] state;
reg [2:0] count; // Counts the shifts

if (reset) // Asynchronous reset
state <= s0;
else
case (state)
s0 : begin // Initialization
state <= s1;
count = 0;
p <= 0;
x0 <= x_in0;
x1 <= x_in1;
x2 <= x_in2;
x3 <= x_in3;
end
s1 : begin // Processing step
if (count == 4) begin // Is sum of product done?
y <= p; // Output of result to y and
state <= s0; // start next sum of product
end
```

و LUT ان به شکل زیر است:

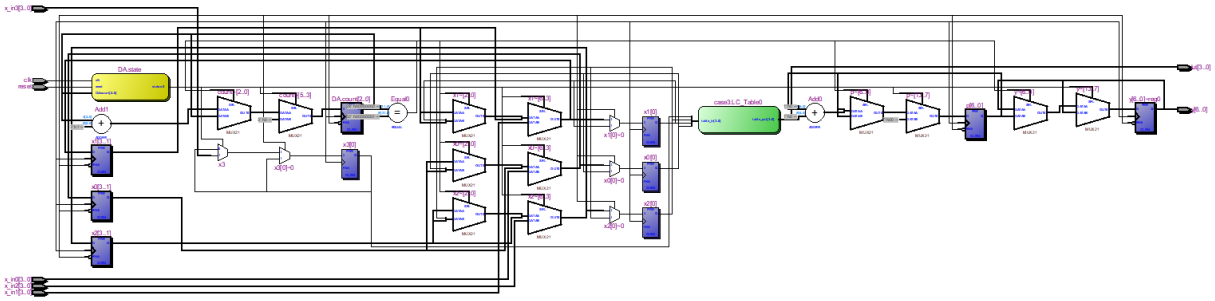
```
module case3 (input [3:0] table_in,
output reg [3:0] table_out);
// This is the DA CASE table for the 4 coefficients: 3, 12, 12, 3
always @(table_in)
begin
case (table_in)
0 : table_out = 0;
1 : table_out = 3;
2 : table_out = 12;
3 : table_out = 15;
4 : table_out = 12;
5 : table_out = 15;
6 : table_out = 24;
7 : table_out = 27;
8 : table_out = 3;
9 : table_out = 6;
10: table_out = 15;
11: table_out = 18;
12: table_out = 15;
13: table_out = 18;
14: table_out = 27;
15: table_out = 30;
default : ;
endcase
end
endmodule
```

کد ها در فایل موجود اند.

Summery

سپس این کد را به Quartes برده و داده های زیر را می گیریم:

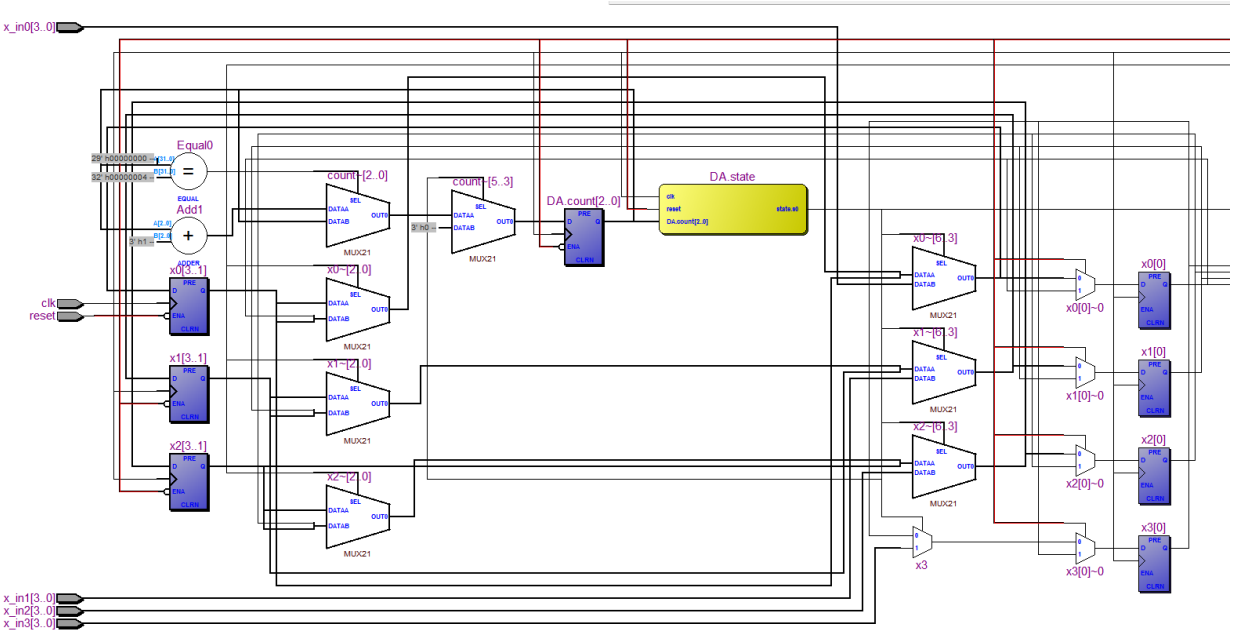
Handwrite_FIR_DA☺ Total Logic Elements: 39, Total Registers: 31;



2) در این قسمت همان کد قبلی را با BCF=2 به معنی اینکه 2بیت از ورودی وارد LUT می شود را می نویسیم کد ان در فایل موجود است

سپس کد را به Quartes می بریم و داریم:

Matlab_Gain_FIR☺ Total Logic Elements: 39, Total Registers: 31;




Activate Windows
Go to Settings to activate Windows.

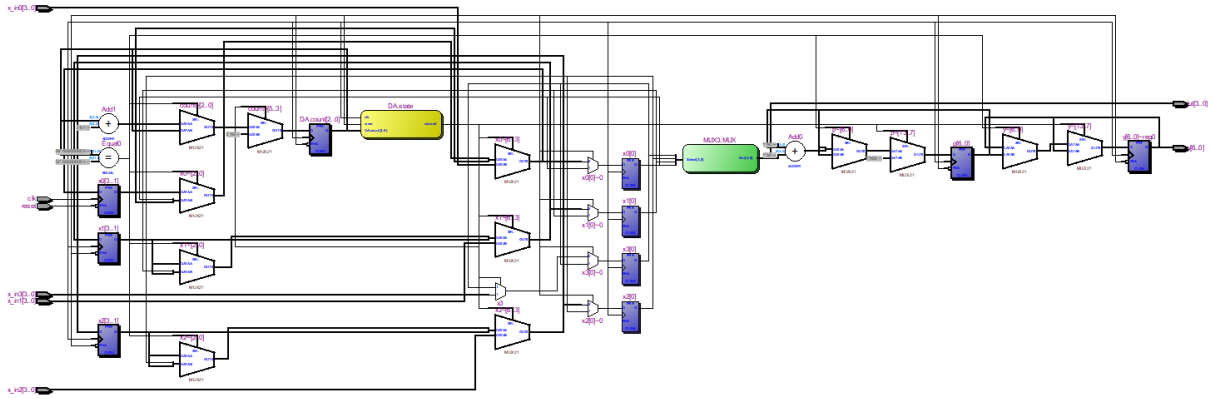
Summery

3) در این قسمت فیلتر را بدون LUT می سازیم و به جای نا از یک MUX استفاده می کنیم که تغریبا همان RTL مربوط به LUT است

```
module MUX3 (input [3:0] Select,
output reg [3:0] Out);
// This is the DA CASE table for the 4 coefficients: 3, 12, 12, 3
// MUX with 4 Select bit
always @(Select)
begin
case (Select)
0 : Out = 0;
1 : Out = 3;
2 : Out = 12;
3 : Out = 15;
4 : Out = 12;
5 : Out = 15;
6 : Out = 24;
7 : Out = 27;
8 : Out = 3;
9 : Out = 6;
10: Out = 15;
11: Out = 18;
12: Out = 15;
13: Out = 18;
14: Out = 27;
15: Out = 30;
default : ;
endcase
end
endmodule
```

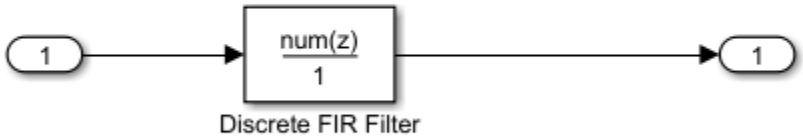
اینجا همانند قبا فیلتر را ساخته ایم ولی به جای LUT از MUX با 4 بیت Select استفاده کرده ایم. کد را به Quartesبرده و داریم:

Handwrite_FIR_DA_No LUT☺ Total Logic Elements: 39, Total Registers: 31;



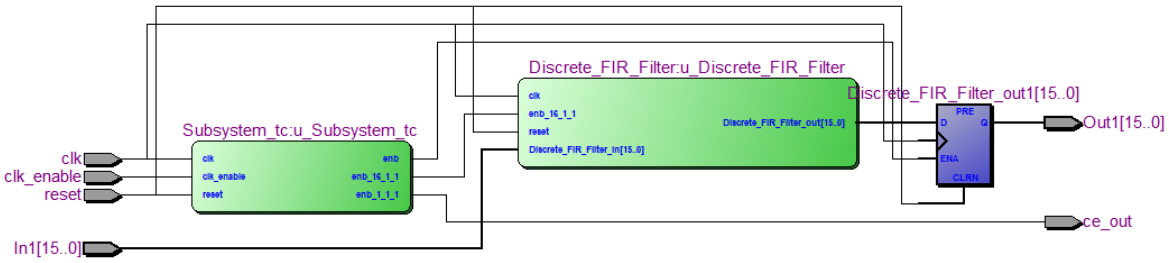
Summery

4) در این قسمت فیلتر را با استفاده از Matlab و HDL Code می خواهیم به صورت DA بسازیم برای این کار از Block FIR Filter استفاده کرده و در قسمت Properties ان را به صورت DA انتخاب می کنیم و کد Verilog و Testbench ان را ما سازیم




سپس کد را به Quartes برده و داریم:

Matlab_FIR_DA ☺ Total Logic Elements: 178, Total Registers: 143;



Summery

5) برای مقایسه این روش ها داریم:

Logic elements: HDL Code > No LUT > BCF=1 > BCF=2

Registers: HDL Code > No LUT > BCF=1 > BCF=2

دلیل ان این است که در LUT BCF=2 ها 8 بیتی است و به تعدهد کمتری رجیستر نیاز دارید و در No LUT که از MUX استفاده شده چون LUT ندارد و RTL است تعداد بیشتر دارد و در نهایت HDL Code بیشترین است