Dual Fixed Point Calculator

ECE 595 Reconfigurable Computing

Instructor:-

Professor Daniel Llamocca

Project by:-

Andres Jacoby

Gaurav Agalave

OAKLAND UNIVERSITY™

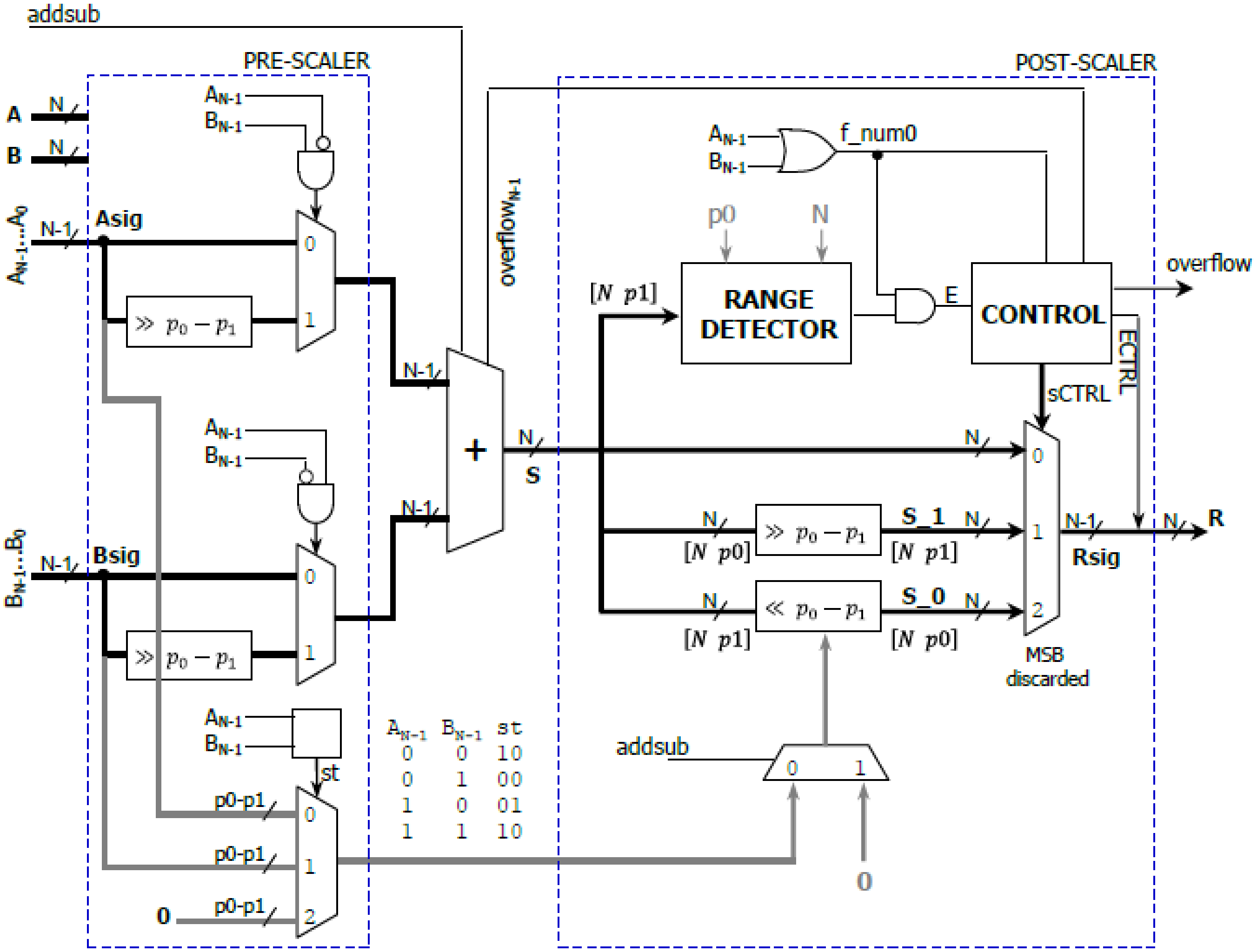
Overview

- Motivation
- DFX Adder, Subtractor
- DFX Multiplier
- DFX Divider
- DFX Test bench and verification using Matlab
- Challenges Faced
- Conclusion

Motivation

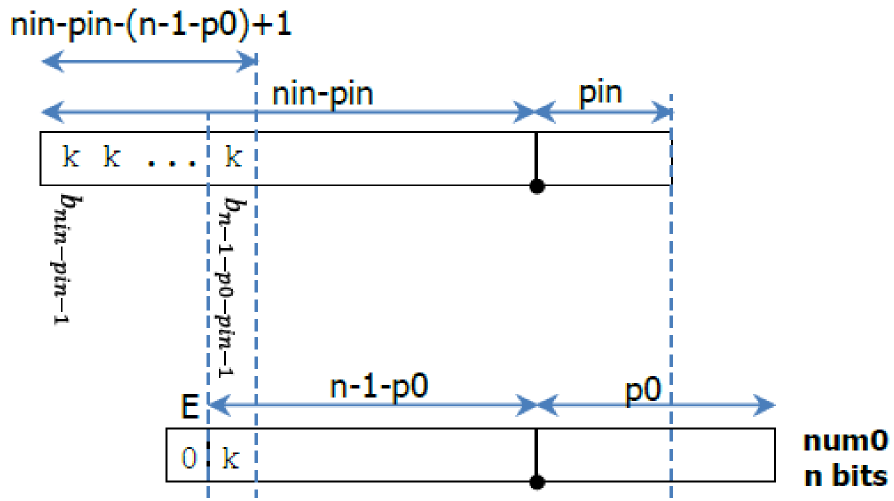
- Advantages of DFX
  - Less Resources
  - Higher precision than FX
- Experience of modelling generic system to enable DPR(Dynamic Partial Reconfiguration)
- Not Widespread

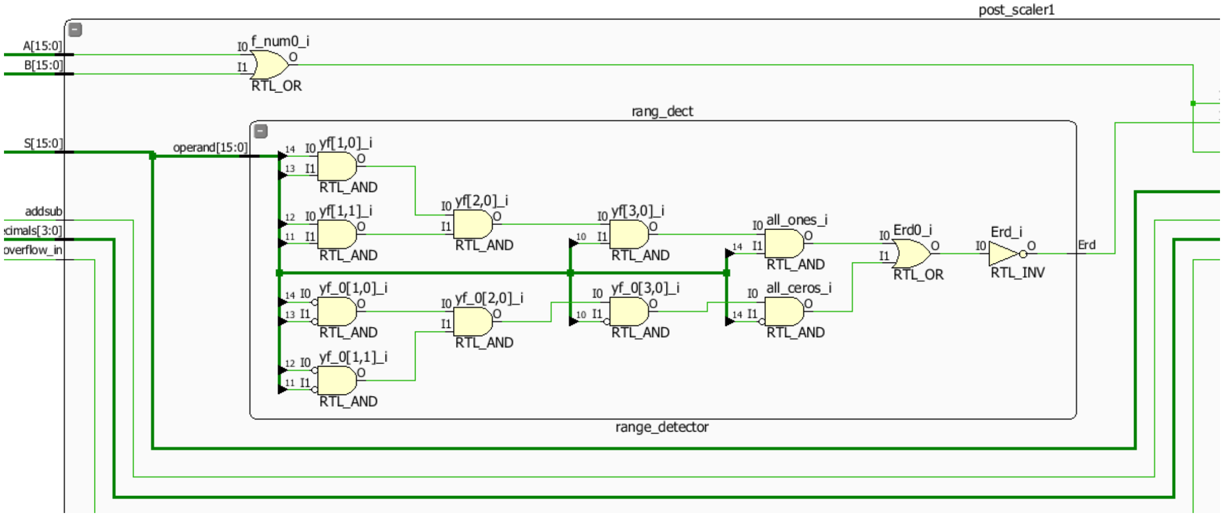
DFX Adder Subtractor



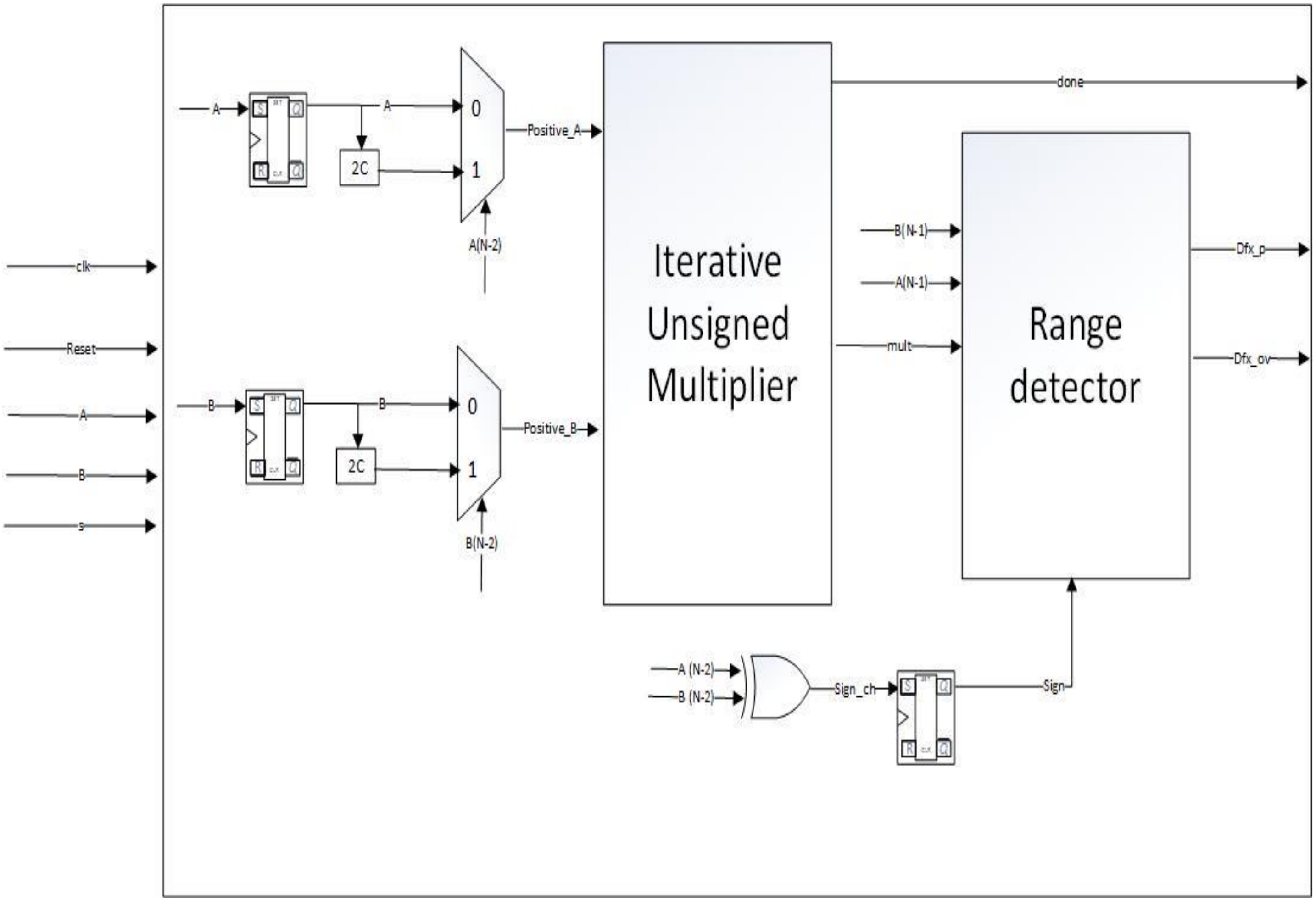
A_{N-1}	B_{N-1}	st
0	0	10
0	1	00
1	0	01
1	1	10

DFX Range Detector

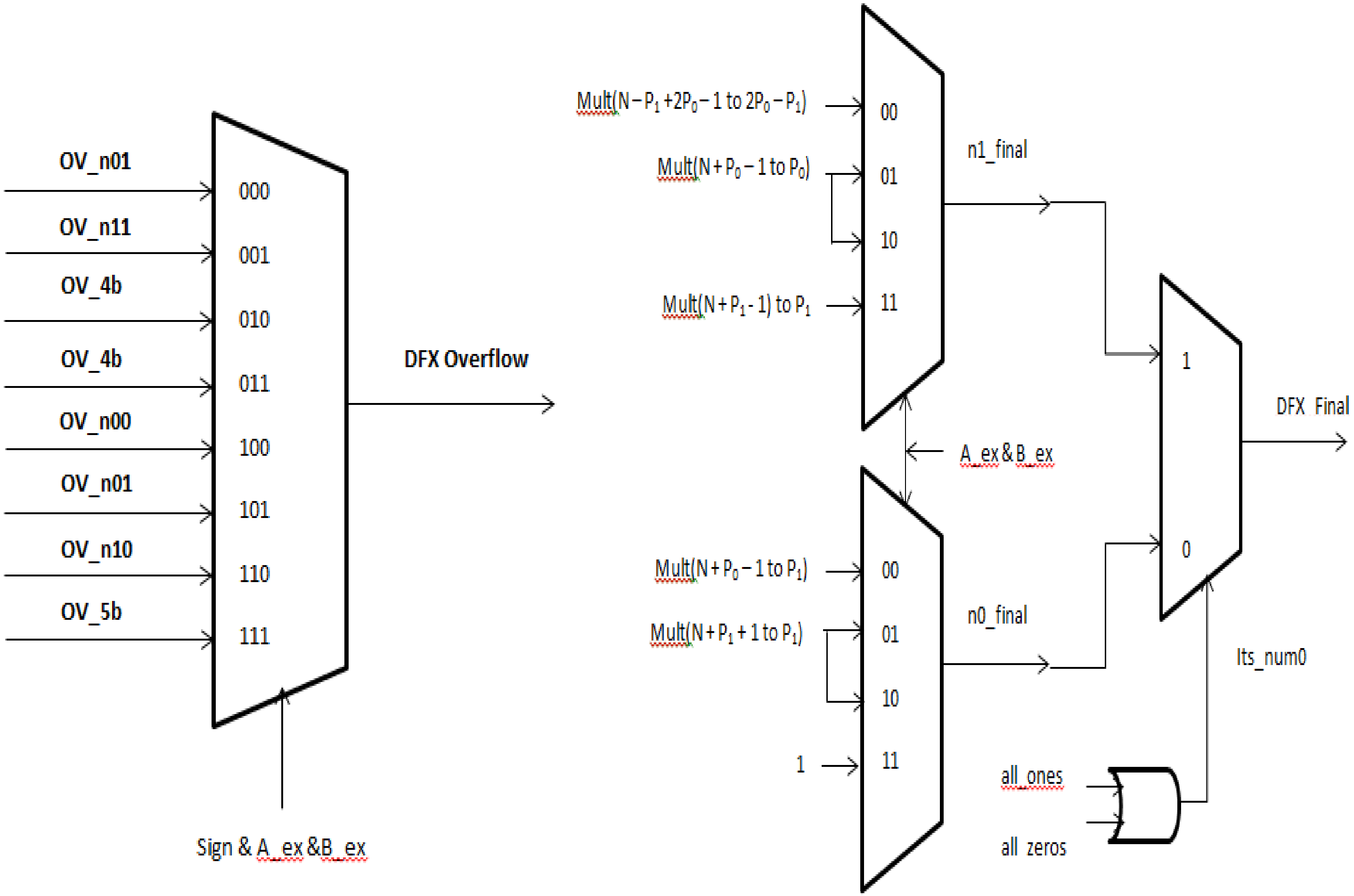




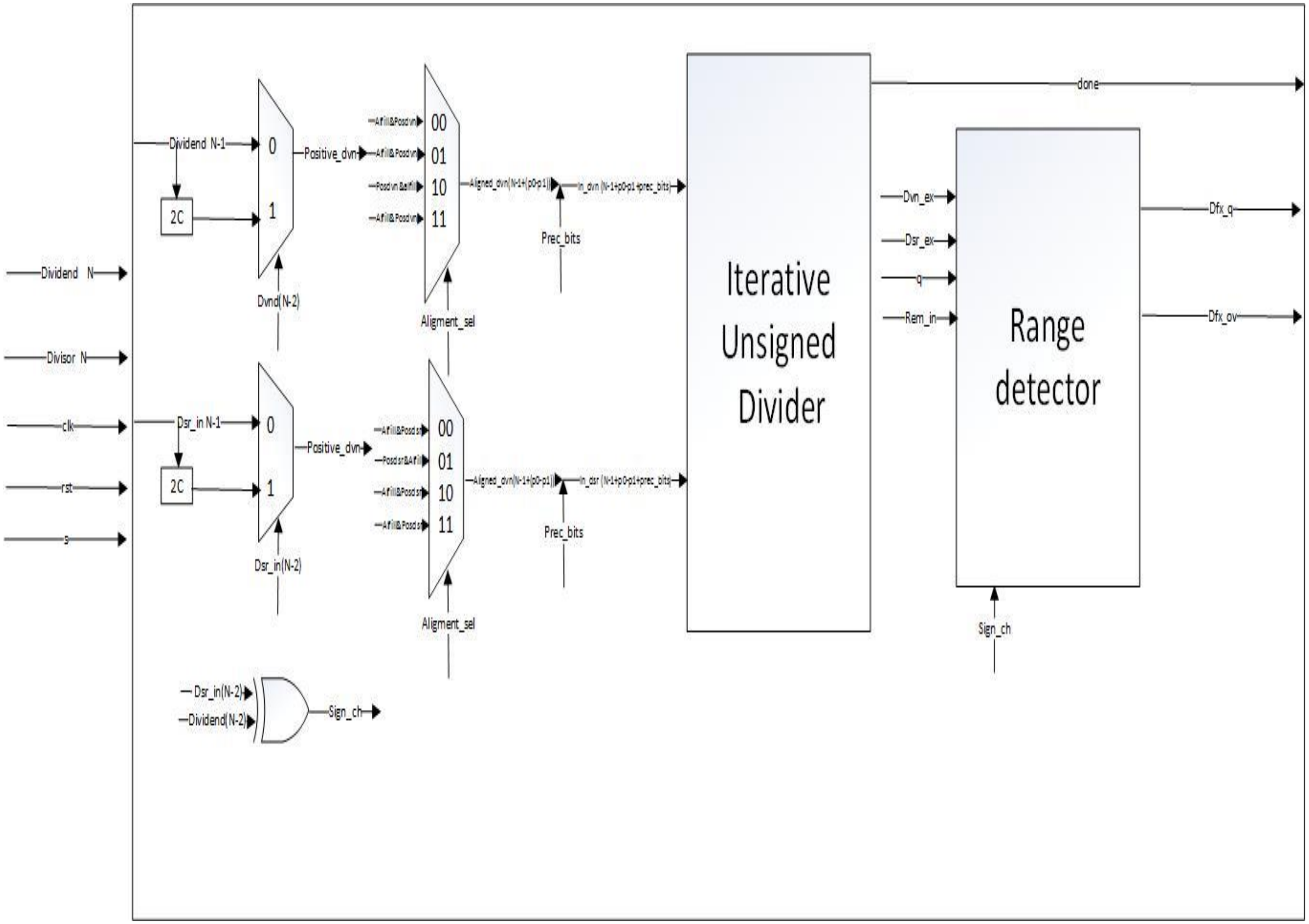
DFX Multiplier



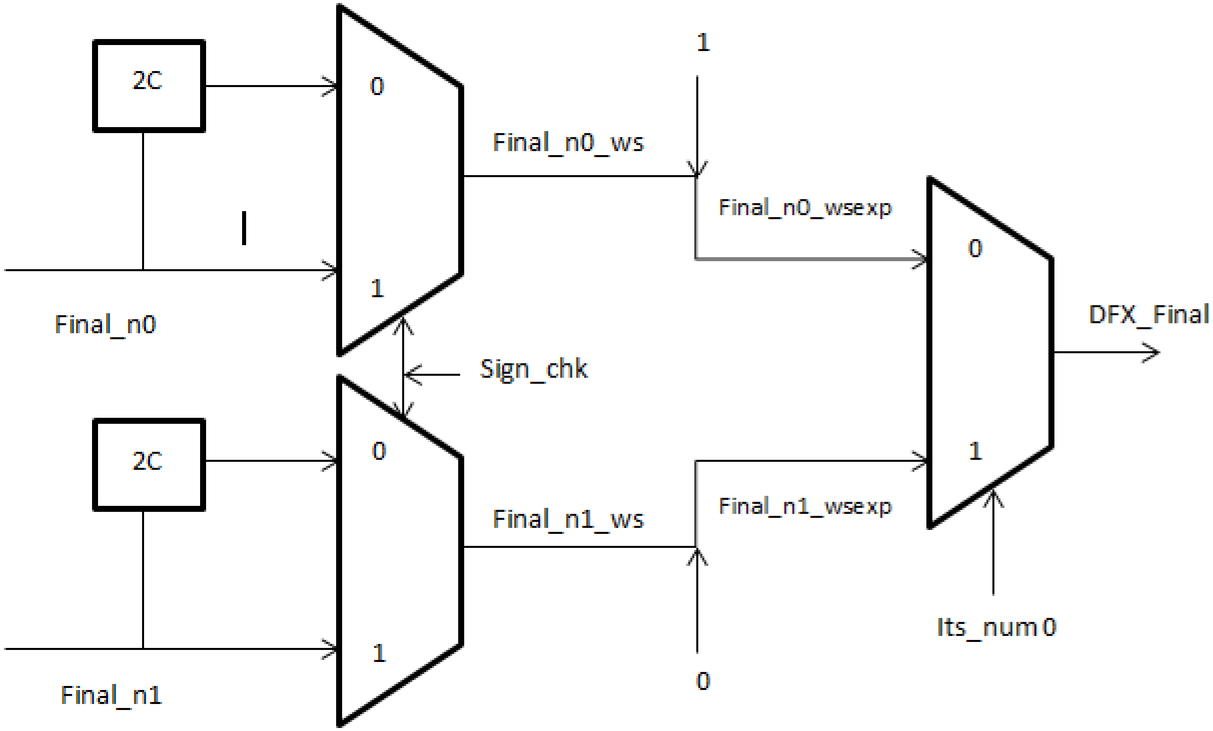
DFX Multiplier Range Detector & Slicing



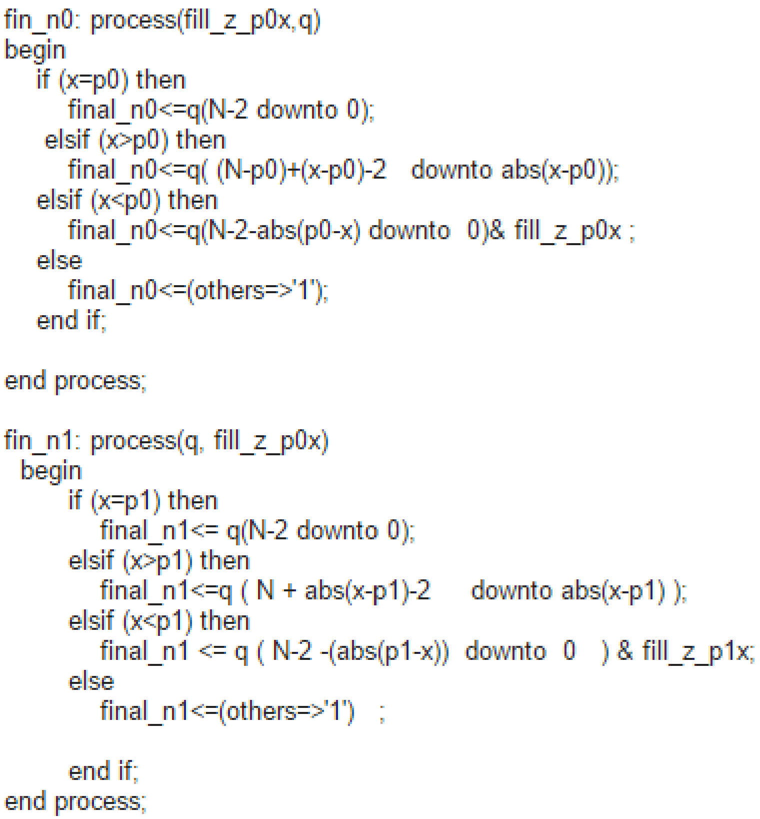
DFX Divider


Dividend N
Divisor N
clk
rst
s
Dividend N-1
2C
0
1
Positive_dvn
Dvnd(N-2)
Afill&Posdvn
Afill&Posdvn
Posdvn&afill
Afill&Posdvn
00
01
10
11
Aligned_dvn(N-1+(p0-p1))
Prec_bits
in_dvn (N-1+p0-p1+prec_bits)
Aligment_sel
Dsr_in N-1
2C
0
1
Positive_dvn
Dsr_in(N-2)
Afill&Posdsr
Posdsr&Afill
Afill&Posdsr
Afill&Posdsr
00
01
10
11
Aligned_dvn(N-1+(p0-p1))
Prec_bits
in_dsr (N-1+p0-p1+prec_bits)
Aligment_sel
Dsr_in(N-2)
Dividend(N-2)
Sign_ch
Iterative Unsigned Divider
done
Dvn_ex
Dsr_ex
q
Rem_in
Range detector
Dfx_q
Dfx_ov
Sign_ch

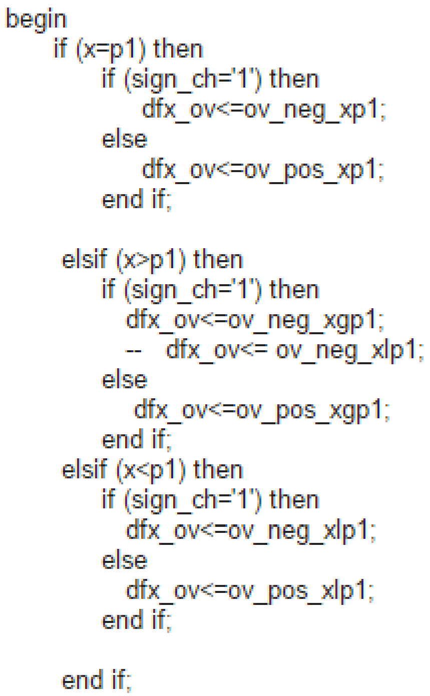
DFX Divider Range Detector



DFX Divider Slicing



```
fin_n0: process(fill_z_p0x,q)
begin
    if (x=p0) then
        final_n0<=q(N-2 downto 0);
     elsif (x>p0) then
        final_n0<=q( (N-p0)+(x-p0)-2   downto abs(x-p0));
    elsif (x<p0) then
        final_n0<=q(N-2-abs(p0-x) downto  0)& fill_z_p0x ;
    else
        final_n0<=(others=>'1');
    end if;

end process;

fin_n1: process(q, fill_z_p0x)
  begin
        if (x=p1) then
            final_n1<= q(N-2 downto 0);
        elsif (x>p1) then
            final_n1<=q ( N + abs(x-p1)-2     downto abs(x-p1) );
        elsif (x<p1) then
            final_n1 <= q ( N-2 -(abs(p1-x))  downto  0    ) & fill_z_p1x;
        else
            final_n1<=(others=>'1')   ;

        end if;
end process;
```



```
begin
      if (x=p1) then
            if (sign_ch='1') then
                 dfx_ov<=ov_neg_xp1;
            else
                 dfx_ov<=ov_pos_xp1;
            end if;

       elsif (x>p1) then
            if (sign_ch='1') then
               dfx_ov<=ov_neg_xgp1;
               --   dfx_ov<= ov_neg_xlp1;
            else
                dfx_ov<=ov_pos_xgp1;
            end if;
       elsif (x<p1) then
            if (sign_ch='1') then
               dfx_ov<=ov_neg_xlp1;
            else
               dfx_ov<=ov_pos_xlp1;
            end if;

       end if;
```

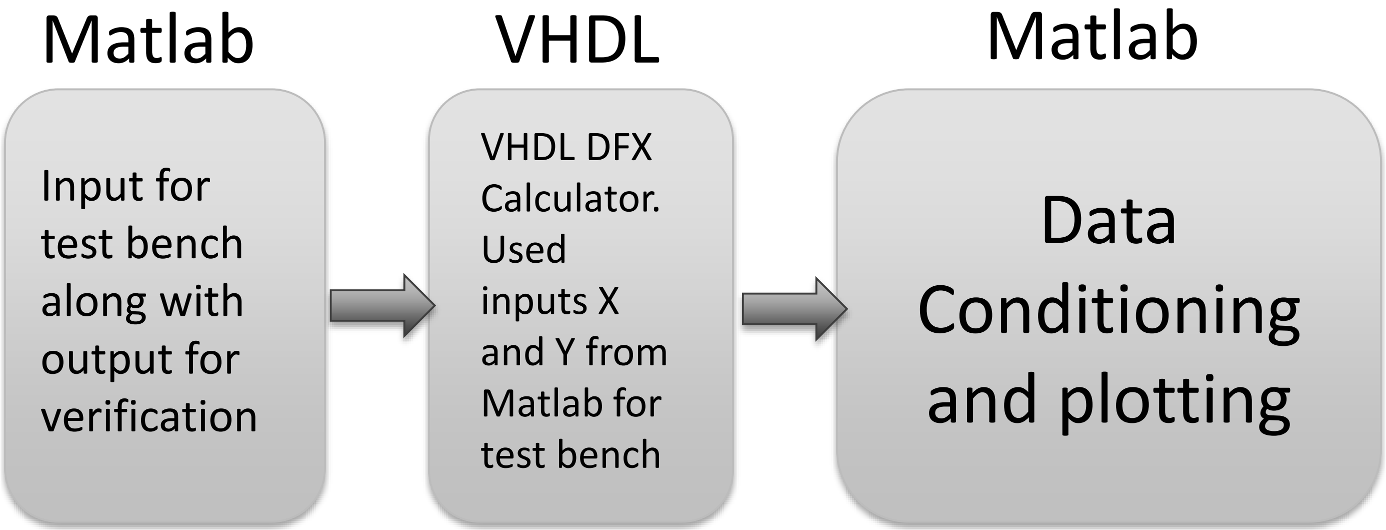
DFX Calculator in MATLAB(For test bench for VHDL)

- Random floating point number Generated using rand().
- Floating points converted to fixed point format using quantizer based on range.
- E.g.

  r = quantizer('fixed',[7,5]);

  q = quantizer('fixed',[7,3]);
- Fixed point numbers again converted into floating point format for calculations.
- Operands along with o/p and overflow data are converted into .txt file.

Verification Process Block Diagram



Matlab

Input for test bench along with output for verification

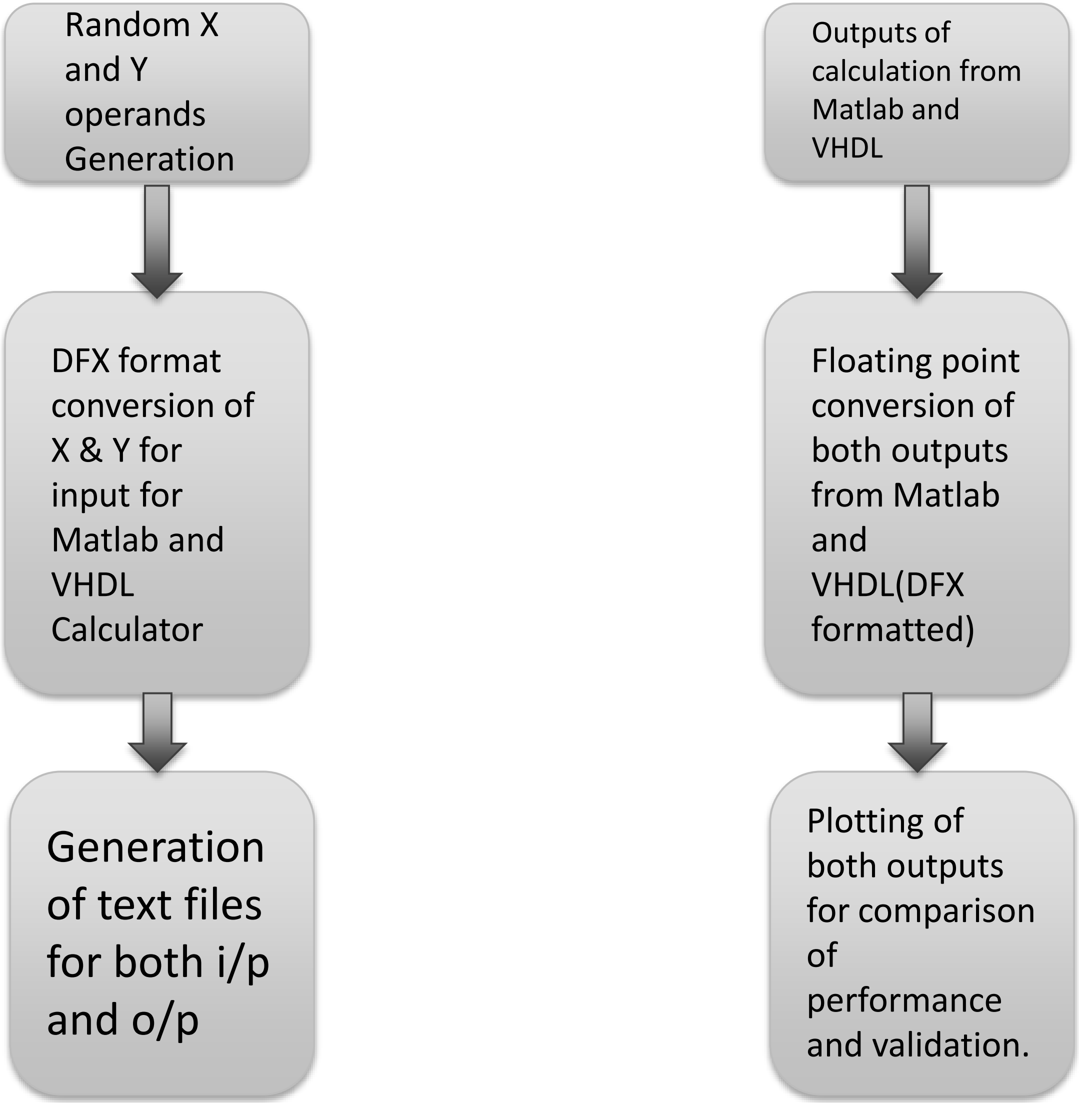
VHDL

VHDL DFX Calculator. Used inputs X and Y from Matlab for test bench

Matlab

Data Conditioning and plotting

Matlab Flow Chart



Random X and Y operands Generation

DFX format conversion of X & Y for input for Matlab and VHDL Calculator

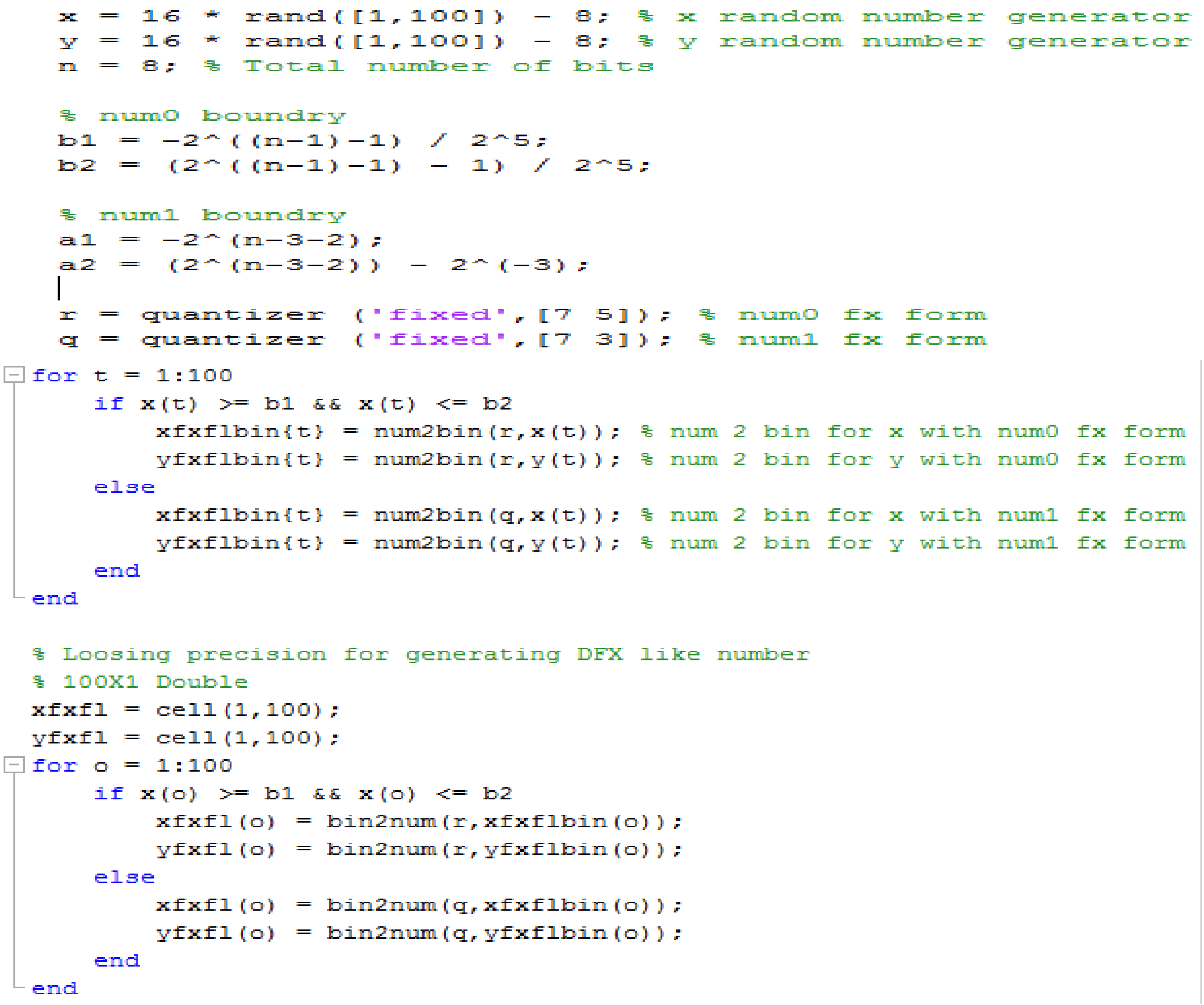
Generation of text files for both i/p and o/p

Outputs of calculation from Matlab and VHDL

Floating point conversion of both outputs from Matlab and VHDL(DFX formatted)

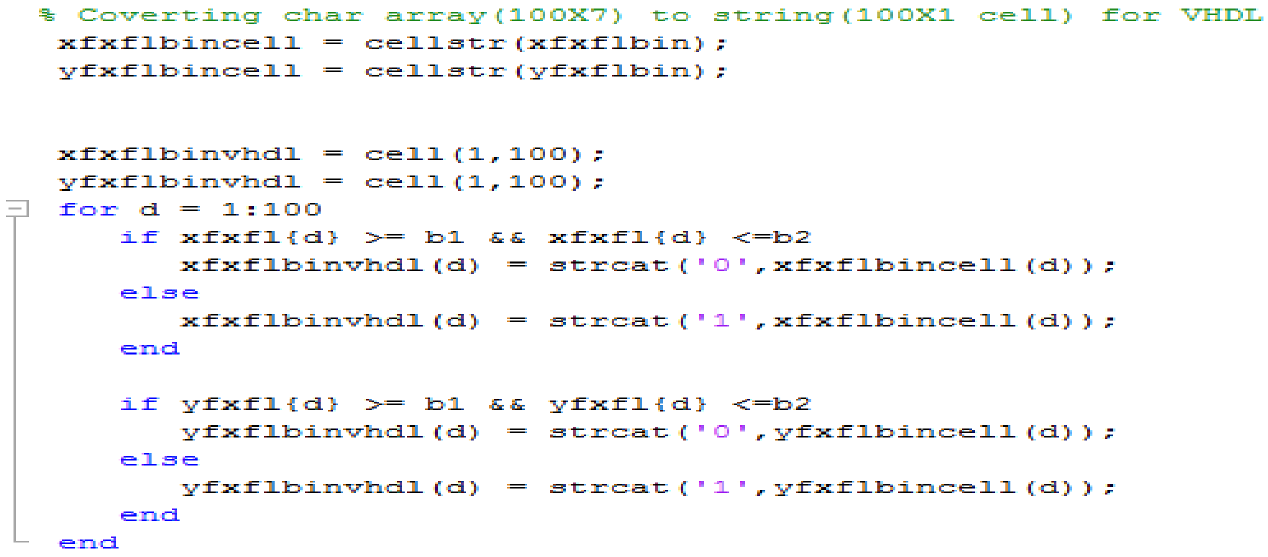
Plotting of both outputs for comparison of performance and validation.

Code Snippet

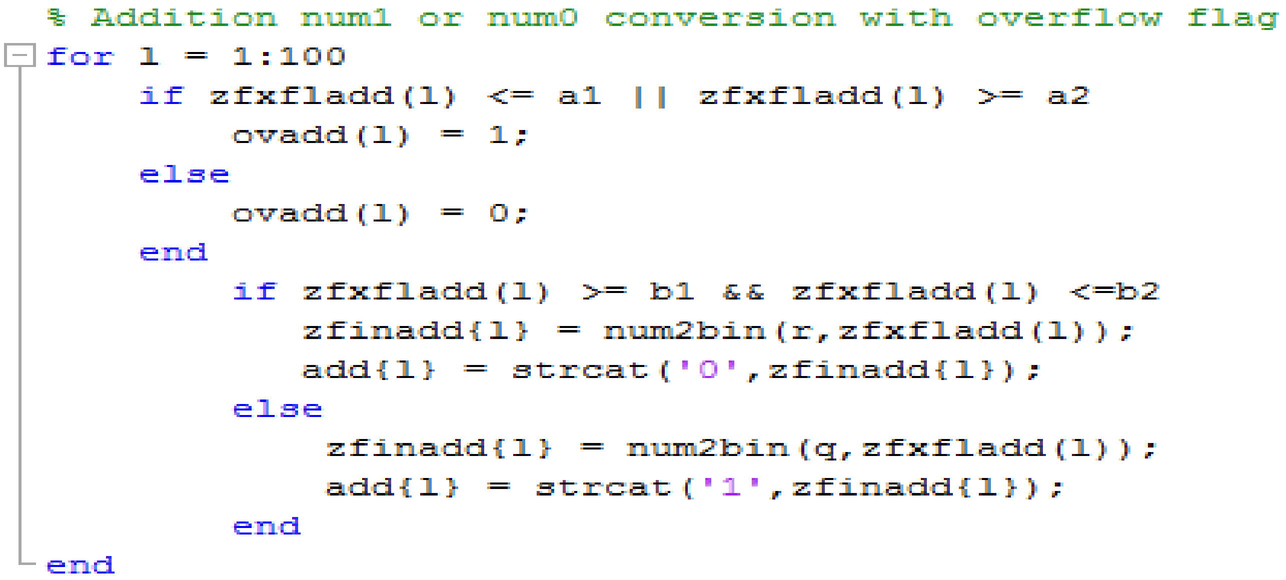


```
 x = 16 * rand([1,100]) - 8; % x random number generator
 y = 16 * rand([1,100]) - 8; % y random number generator
 n = 8; % Total number of bits

 % num0 boundry
 b1 = -2^((n-1)-1) / 2^5;
 b2 = (2^((n-1)-1) - 1) / 2^5;

 % num1 boundry
 a1 = -2^(n-3-2);
 a2 = (2^(n-3-2)) - 2^(-3);

 r = quantizer ('fixed',[7 5]); % num0 fx form
 q = quantizer ('fixed',[7 3]); % num1 fx form
for t = 1:100
    if x(t) >= b1 && x(t) <= b2
        xfxflbin{t} = num2bin(r,x(t)); % num 2 bin for x with num0 fx form
        yfxflbin{t} = num2bin(r,y(t)); % num 2 bin for y with num0 fx form
    else
        xfxflbin{t} = num2bin(q,x(t)); % num 2 bin for x with num1 fx form
        yfxflbin{t} = num2bin(q,y(t)); % num 2 bin for y with num1 fx form
    end
end

% Loosing precision for generating DFX like number
% 100X1 Double
xfxfl = cell(1,100);
yfxfl = cell(1,100);
for o = 1:100
    if x(o) >= b1 && x(o) <= b2
        xfxfl(o) = bin2num(r,xfxflbin(o));
        yfxfl(o) = bin2num(r,yfxflbin(o));
    else
        xfxfl(o) = bin2num(q,xfxflbin(o));
        yfxfl(o) = bin2num(q,yfxflbin(o));
    end
end
```

Code Snippet

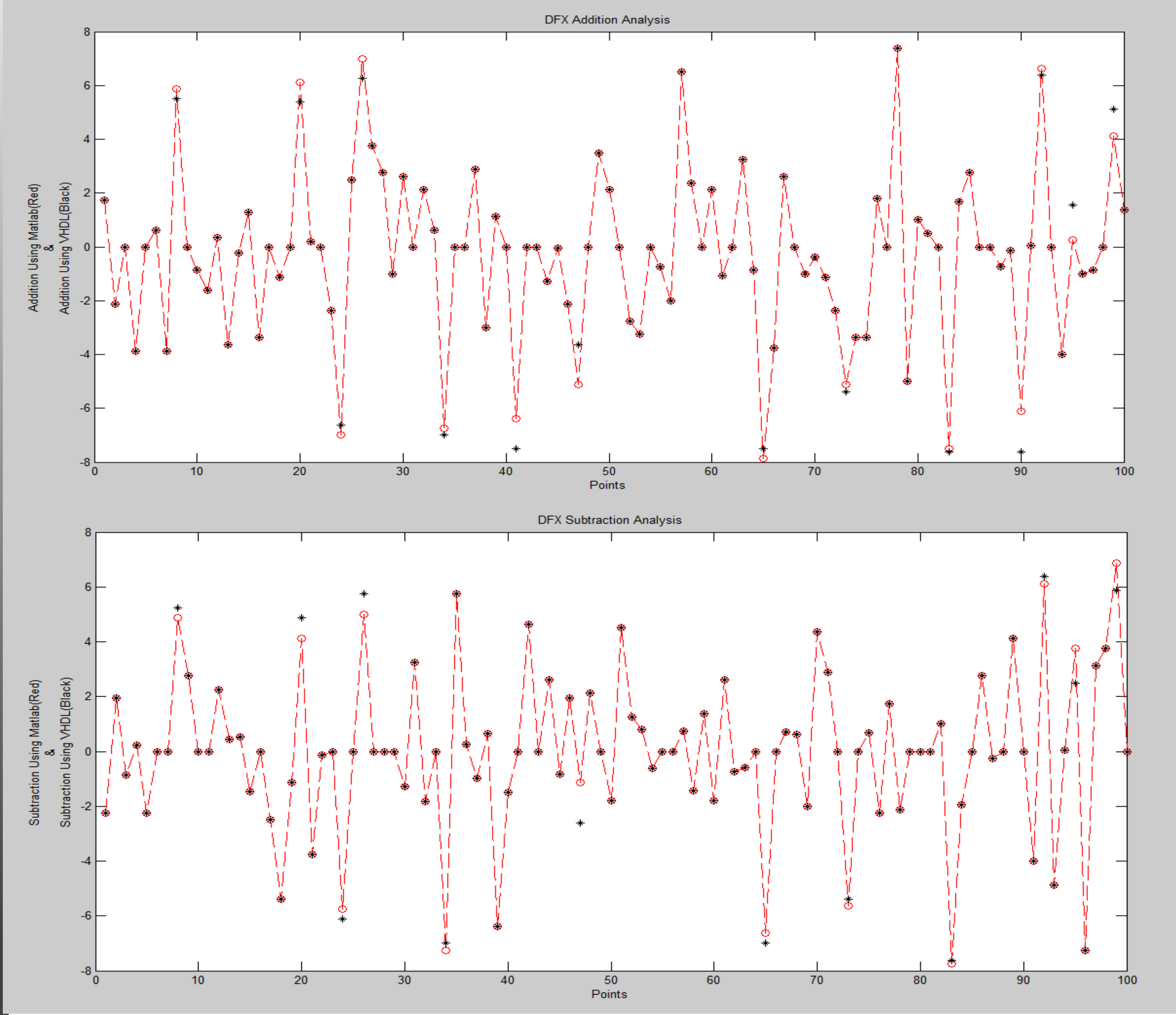


```
% Coverting char array(100X7) to string(100X1 cell) for VHDL
 xfxflbincell = cellstr(xfxflbin);
 yfxflbincell = cellstr(yfxflbin);


 xfxflbinvhdl = cell(1,100);
 yfxflbinvhdl = cell(1,100);
 for d = 1:100
     if xfxfl{d} >= b1 && xfxfl{d} <=b2
         xfxflbinvhdl(d) = strcat('0',xfxflbincell(d));
     else
         xfxflbinvhdl(d) = strcat('1',xfxflbincell(d));
     end

     if yfxfl{d} >= b1 && yfxfl{d} <=b2
         yfxflbinvhdl(d) = strcat('0',yfxflbincell(d));
     else
         yfxflbinvhdl(d) = strcat('1',yfxflbincell(d));
     end
 end
```



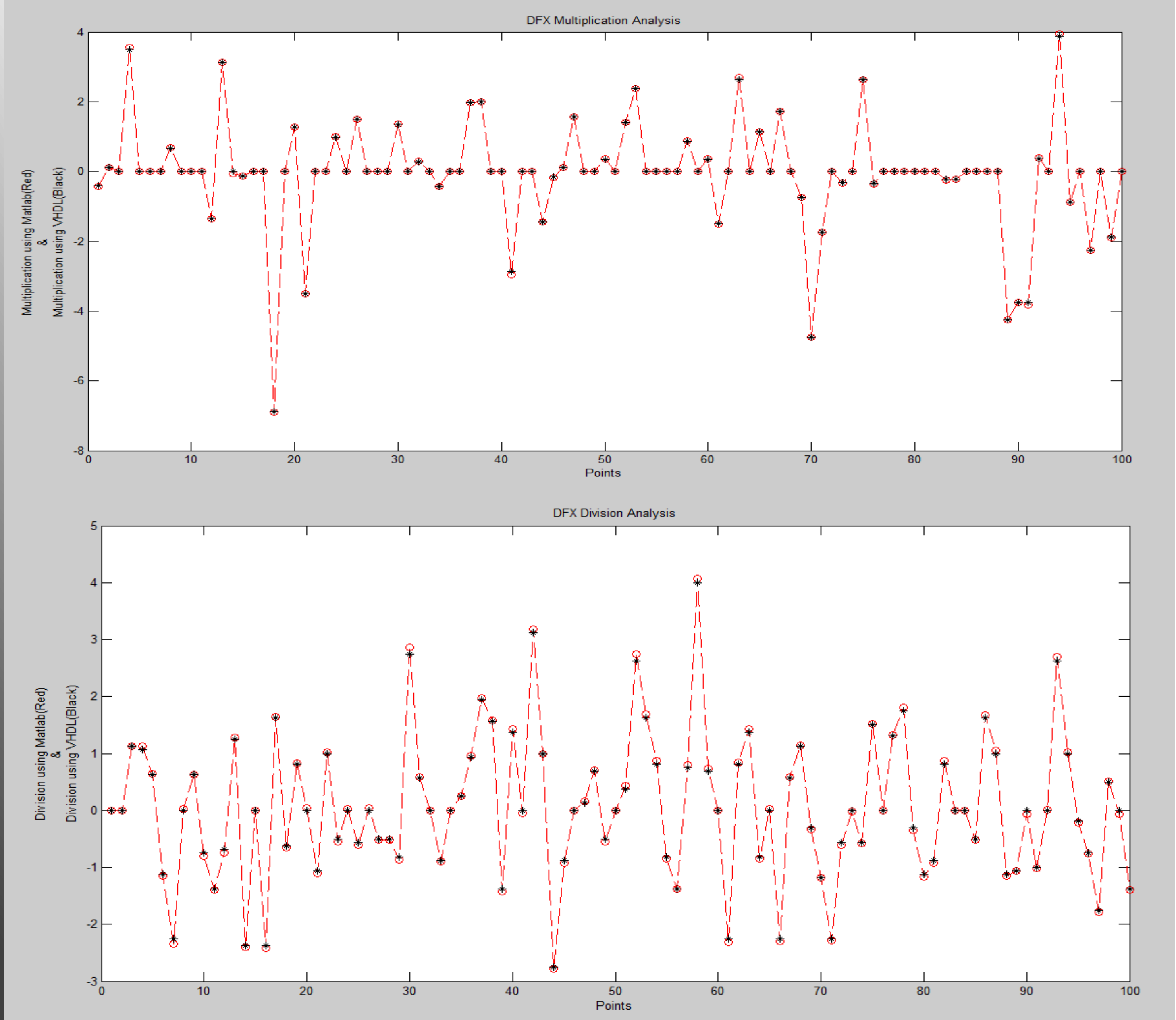
```
% Addition num1 or num0 conversion with overflow flag
for l = 1:100
    if zfxfladd(l) <= a1 || zfxfladd(l) >= a2
        ovadd(l) = 1;
    else
        ovadd(l) = 0;
    end
        if zfxfladd(l) >= b1 && zfxfladd(l) <=b2
           zfinadd{l} = num2bin(r,zfxfladd(l));
           add{l} = strcat('0',zfinadd{l});
        else
            zfinadd{l} = num2bin(q,zfxfladd(l));
            add{l} = strcat('1',zfinadd{l});
        end
end
```

Plot for DFX 8_5_3

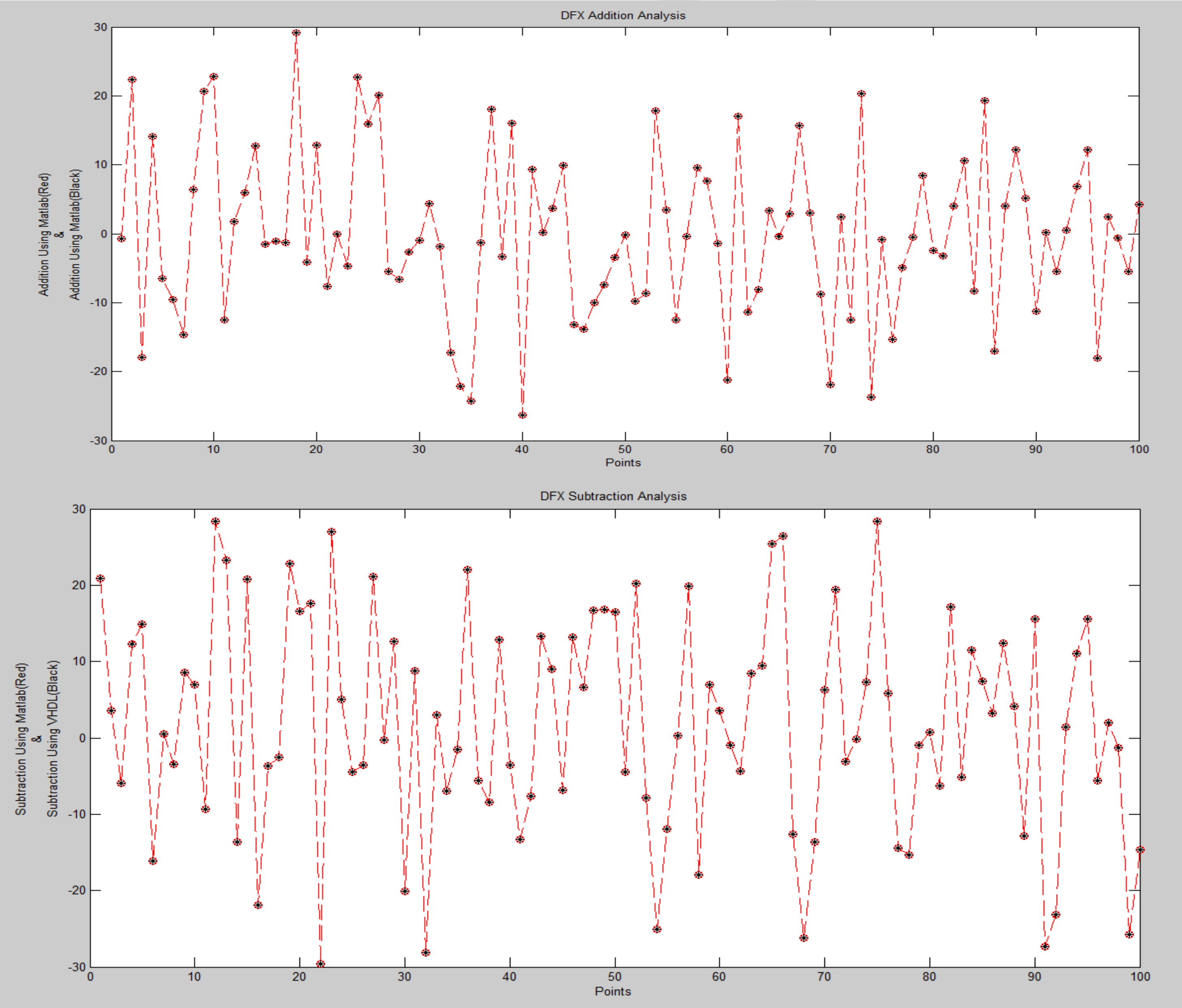


OAKLAND UNIVERSITY™

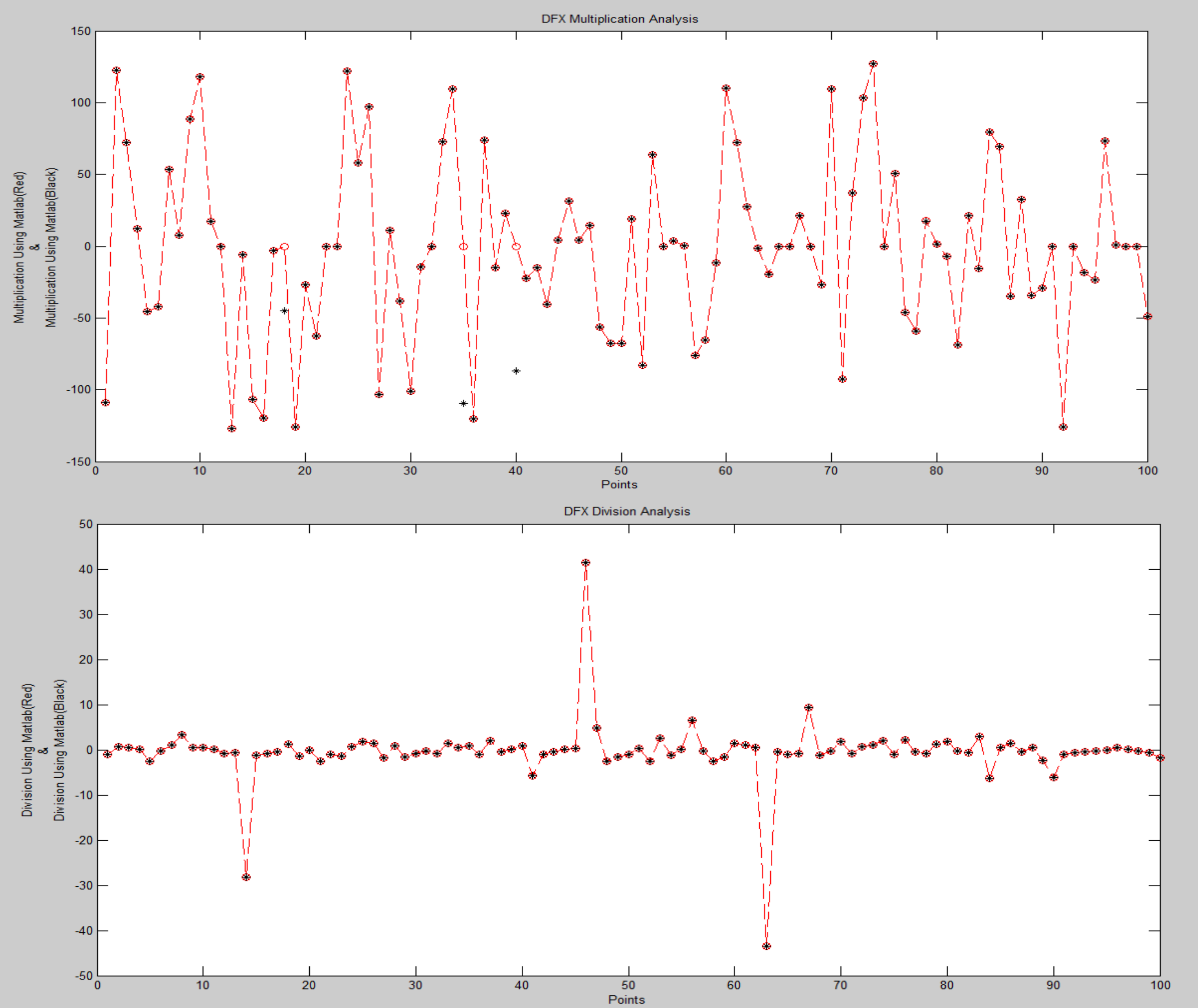
Plot for DFX 8_5_3



Plot for DFX 16_10_17



Plot for DFX 16_10_7



OAKLAND UNIVERSITY™

Relative error in both models w.r.t Matlab model.

Operation	Relative Error
Addition	-0.0529
Subtraction	1.3037
Multiplication	0.0198
Division	-2.9183e-04

Resources used in VHDL

LUT used in Operation	8_5_3	16_10_7	24_15_10	32_20_15
Addition & Subtraction	31	64	76	101
Multiplication	73	187	302	353
Division	84	190	346	443

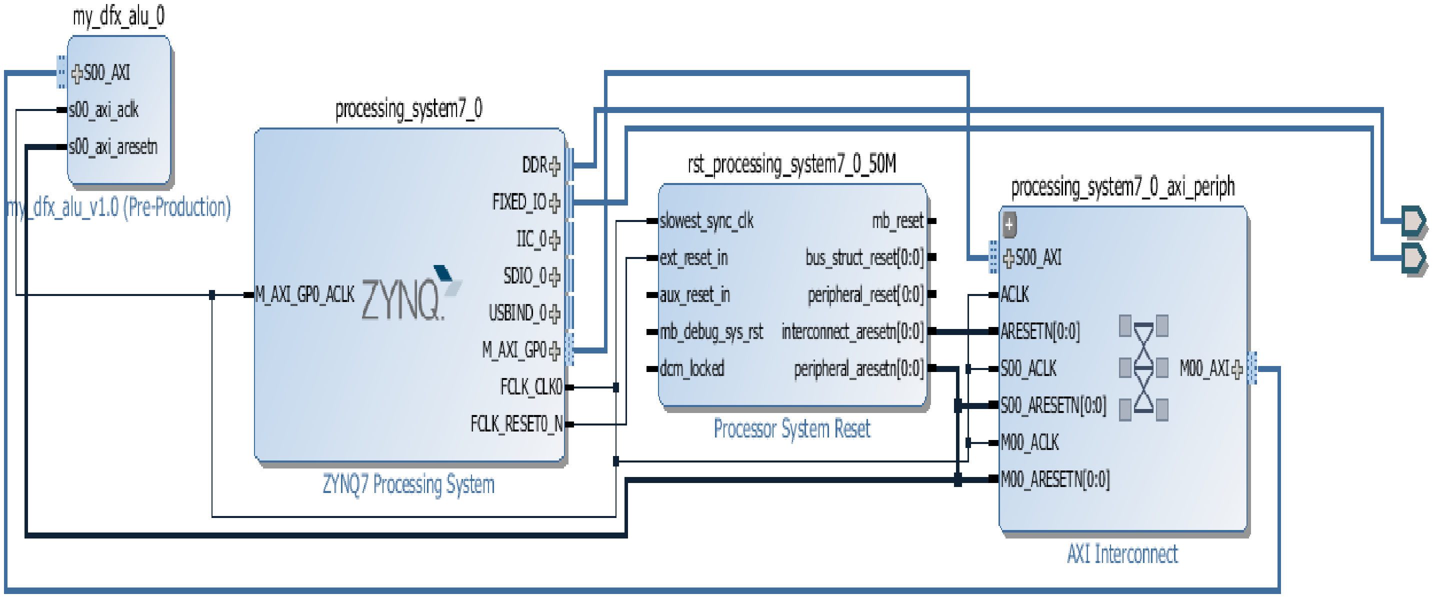
Time required for 1 operation in VHDL

Operation	Time
Addition & Subtraction	Combinational
Multiplication 8_5_3	8 Clocks
Division 8_5_3	16 Clock

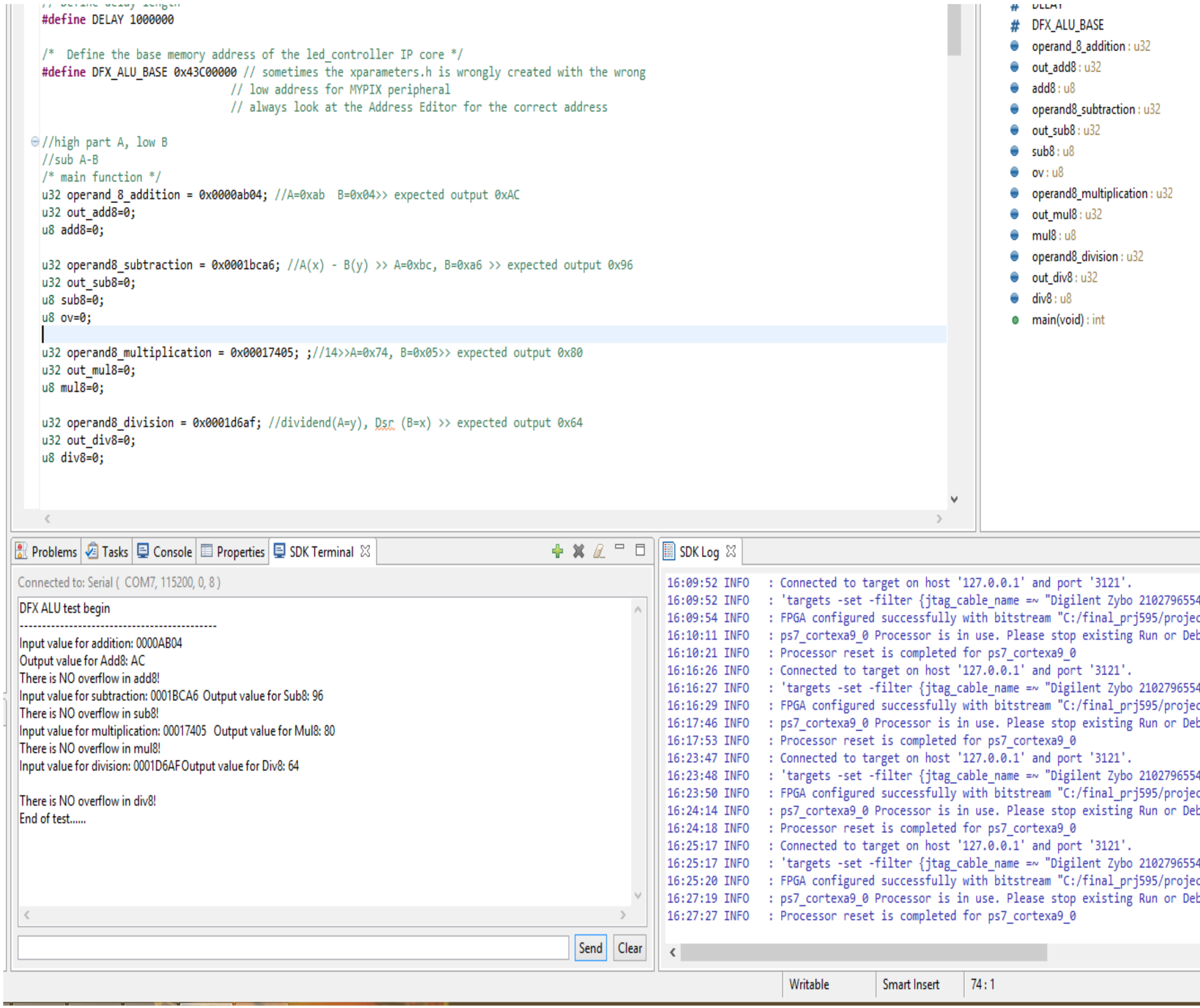
Challenges Faced

- VHDL
- MATLAB

DFX IP Package



DFX Calculator Demo



```
// Define delay length
#define DELAY 1000000

/*  Define the base memory address of the led_controller IP core */
#define DFX_ALU_BASE 0x43C00000 // sometimes the xparameters.h is wrongly created with the wrong
                              // low address for MYPIX peripheral
                              // always look at the Address Editor for the correct address

//high part A, low B
//sub A-B
/* main function */
u32 operand_8_addition = 0x0000ab04; //A=0xab  B=0x04>> expected output 0xAC
u32 out_add8=0;
u8 add8=0;

u32 operand8_subtraction = 0x0001bca6; //A(x) - B(y) >> A=0xbc, B=0xa6 >> expected output 0x96
u32 out_sub8=0;
u8 sub8=0;
u8 ov=0;

u32 operand8_multiplication = 0x00017405; ;//14>>A=0x74, B=0x05>> expected output 0x80
u32 out_mul8=0;
u8 mul8=0;

u32 operand8_division = 0x0001d6af; //dividend(A=y), Dsr (B=x) >> expected output 0x64
u32 out_div8=0;
u8 div8=0;
```

```
# DELAY
# DFX_ALU_BASE
● operand_8_addition : u32
● out_add8 : u32
● add8 : u8
● operand8_subtraction : u32
● out_sub8 : u32
● sub8 : u8
● ov : u8
● operand8_multiplication : u32
● out_mul8 : u32
● mul8 : u8
● operand8_division : u32
● out_div8 : u32
● div8 : u8
● main(void) : int
```

Connected to: Serial (COM7, 115200, 0, 8)

```
DFX ALU test begin
-----------------------------------------
Input value for addition: 0000AB04
Output value for Add8: AC
There is NO overflow in add8!
Input value for subtraction: 0001BCA6 Output value for Sub8: 96
There is NO overflow in sub8!
Input value for multiplication: 00017405  Output value for Mul8: 80
There is NO overflow in mul8!
Input value for division: 0001D6AF Output value for Div8: 64

There is NO overflow in div8!
End of test......
```

```
16:09:52 INFO   : Connected to target on host '127.0.0.1' and port '3121'.
16:09:52 INFO   : 'targets -set -filter {jtag_cable_name =~ "Digilent Zybo 2102796554
16:09:54 INFO   : FPGA configured successfully with bitstream "C:/final_prj595/projec
16:10:11 INFO   : ps7_cortexa9_0 Processor is in use. Please stop existing Run or Deb
16:10:21 INFO   : Processor reset is completed for ps7_cortexa9_0
16:16:26 INFO   : Connected to target on host '127.0.0.1' and port '3121'.
16:16:27 INFO   : 'targets -set -filter {jtag_cable_name =~ "Digilent Zybo 2102796554
16:16:29 INFO   : FPGA configured successfully with bitstream "C:/final_prj595/projec
16:17:46 INFO   : ps7_cortexa9_0 Processor is in use. Please stop existing Run or Deb
16:17:53 INFO   : Processor reset is completed for ps7_cortexa9_0
16:23:47 INFO   : Connected to target on host '127.0.0.1' and port '3121'.
16:23:48 INFO   : 'targets -set -filter {jtag_cable_name =~ "Digilent Zybo 2102796554
16:23:50 INFO   : FPGA configured successfully with bitstream "C:/final_prj595/projec
16:24:14 INFO   : ps7_cortexa9_0 Processor is in use. Please stop existing Run or Deb
16:24:18 INFO   : Processor reset is completed for ps7_cortexa9_0
16:25:17 INFO   : Connected to target on host '127.0.0.1' and port '3121'.
16:25:17 INFO   : 'targets -set -filter {jtag_cable_name =~ "Digilent Zybo 2102796554
16:25:20 INFO   : FPGA configured successfully with bitstream "C:/final_prj595/projec
16:27:19 INFO   : ps7_cortexa9_0 Processor is in use. Please stop existing Run or Deb
16:27:27 INFO   : Processor reset is completed for ps7_cortexa9_0
```

OAKLAND UNIVERSITY™

Thank you!
Questions?