Circular iterative CORDIC using Dual Fixed-Point Arithmetic

Oakland University

By:Daniyah Alaswad, Yazen Alali.

Overview


Introduction.


Motivation.


Procedure.


Results .

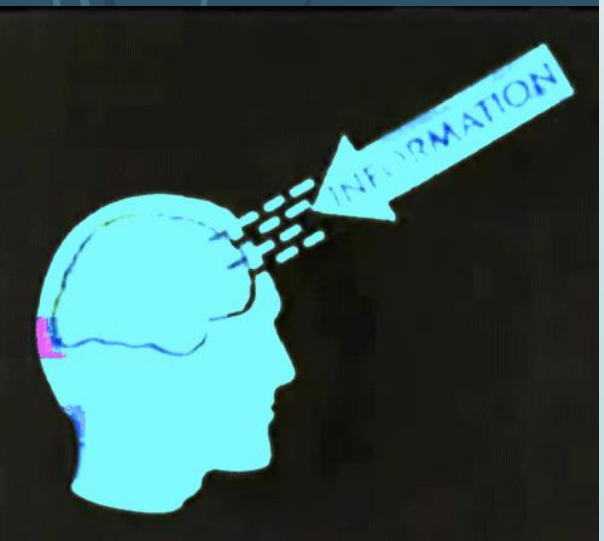

Challenges.


Conclusion.

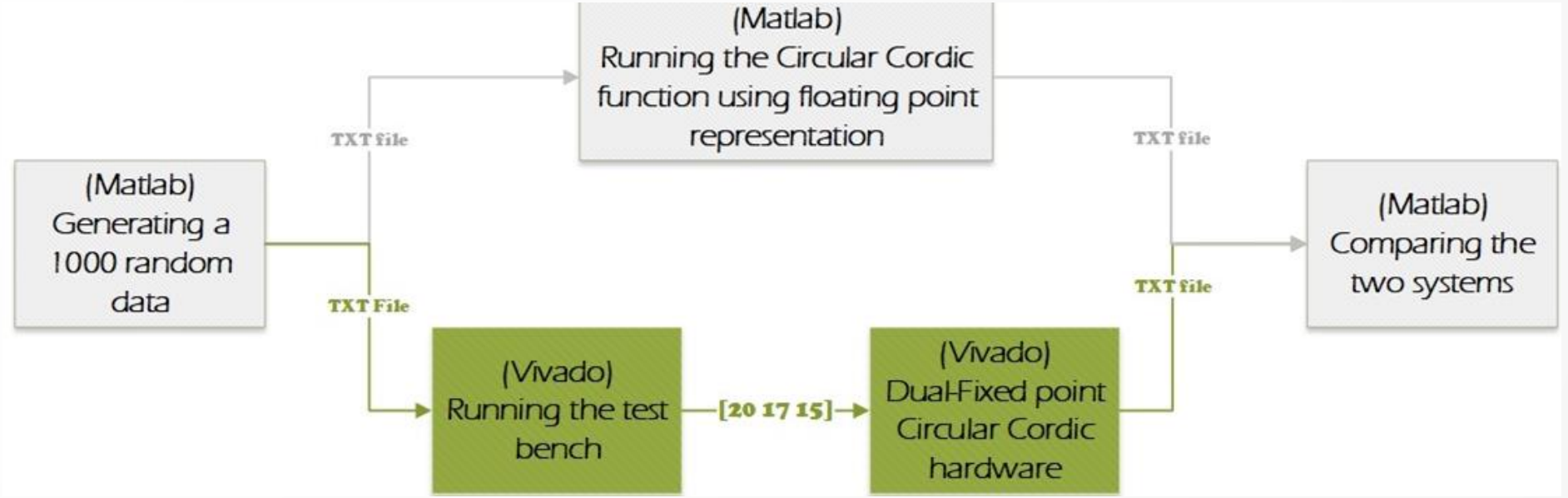
Introduction:

- All DSP tasks can be efficiently implemented using processing elements performing vector rotations .The Coordinate Rotation Digital Computer algorithm CORDIC offers the opportunity to calculate all the desired functions in a rather simple and elegant way.

Motivation:



- It combines the simplicity of a fixed-point system with the wider dynamic range offered by a floating point system.
- Using a single bit exponent which selects two different fixed- point representations, it allows dynamic scaling of signals throughout the system.
- Less Resources.
- Higher precision than FX.
- Not Widespread .
- Applied the knowledge acquired in class in designing the DFX CORDIC.



Procedure: Block diagram

MATLAB code:

- Random floating point number Generated using rand() for x , y, and z.
- We set boundaries for num0 and num1.
- Set the values to (-8,8) for x and y. (-pi/2, pi/2) for z.
- Floating points converted to fixed point format using quantizer based on range.
- convert the binary numbers to integers.
- Print files.

```
%% generate 1000 data such that the output won't go above 7.9999 and below -8
i = 1;
while i <= 1000
    x_test = 2^(n - p1 - 3) * rand - 2^(n - p1 - 4); % x random number generator
    y_test = 2^(n - p1 - 3) * rand - 2^(n - p1 - 4); % y random number generator
    z_test = pi * rand - (pi/2); % z random number generator

    [xx_0 , yy_0, zz_0] = cordic_circular_esp(x_test,y_test,z_test, 0,15);
    [xx_1 , yy_1, zz_1] = cordic_circular_esp(x_test,y_test,z_test, 1,15);

    if (xx_0(18)<7.9999 && xx_0(18)>-8) && (yy_0(18)<7.9999 && yy_0(18)>-8) && (zz_0(18)<7.9999 && zz_0(18)
        if (xx_1(18)<7.9999 && xx_1(18)>-8) && (yy_1(18)<7.9999 && yy_1(18)>-8) && (zz_1(18)<7.9999 && zz_1
            x(i) = x_test;
            y(i) = y_test;
            z(i) = z_test;
            i = i + 1;
        end
    end
end
```

MATLAB Comparison

- Take the output file from VHDL code.
- Convert it to binary.
- Take the input files.
- Perform calculation.
- Compare MATLAB output with VHDL output.
- Find error percentage.

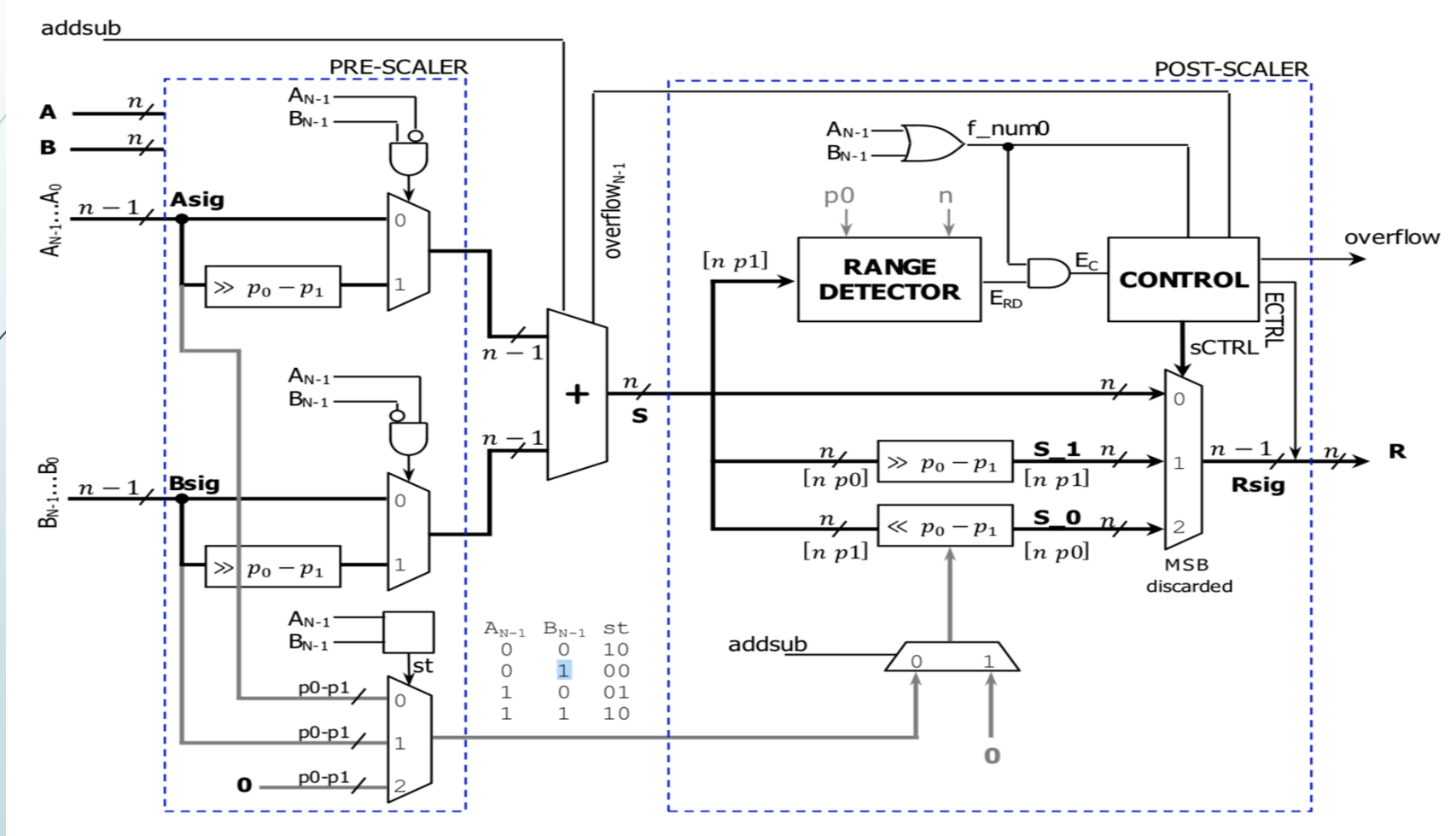
```
Run the cordic function in matlab

zeros(length(x_unsigned(:,1)),1);          % declare the
zeros(length(y_unsigned(:,1)),1);          % declare the
zeros(length(z_unsigned(:,1)),1);          % declare the

i = 1:length(z_unsigned(:,1))
if i > 500
    mode = 1;
else
    mode = 0;
end
[xx, yy, zz] = cordic_circular_esp(x_DFX(i),y_DFX(i),
x(i) = xx(18);
y(i) = yy(18);
z(i) = zz(18);
```

Challenges:

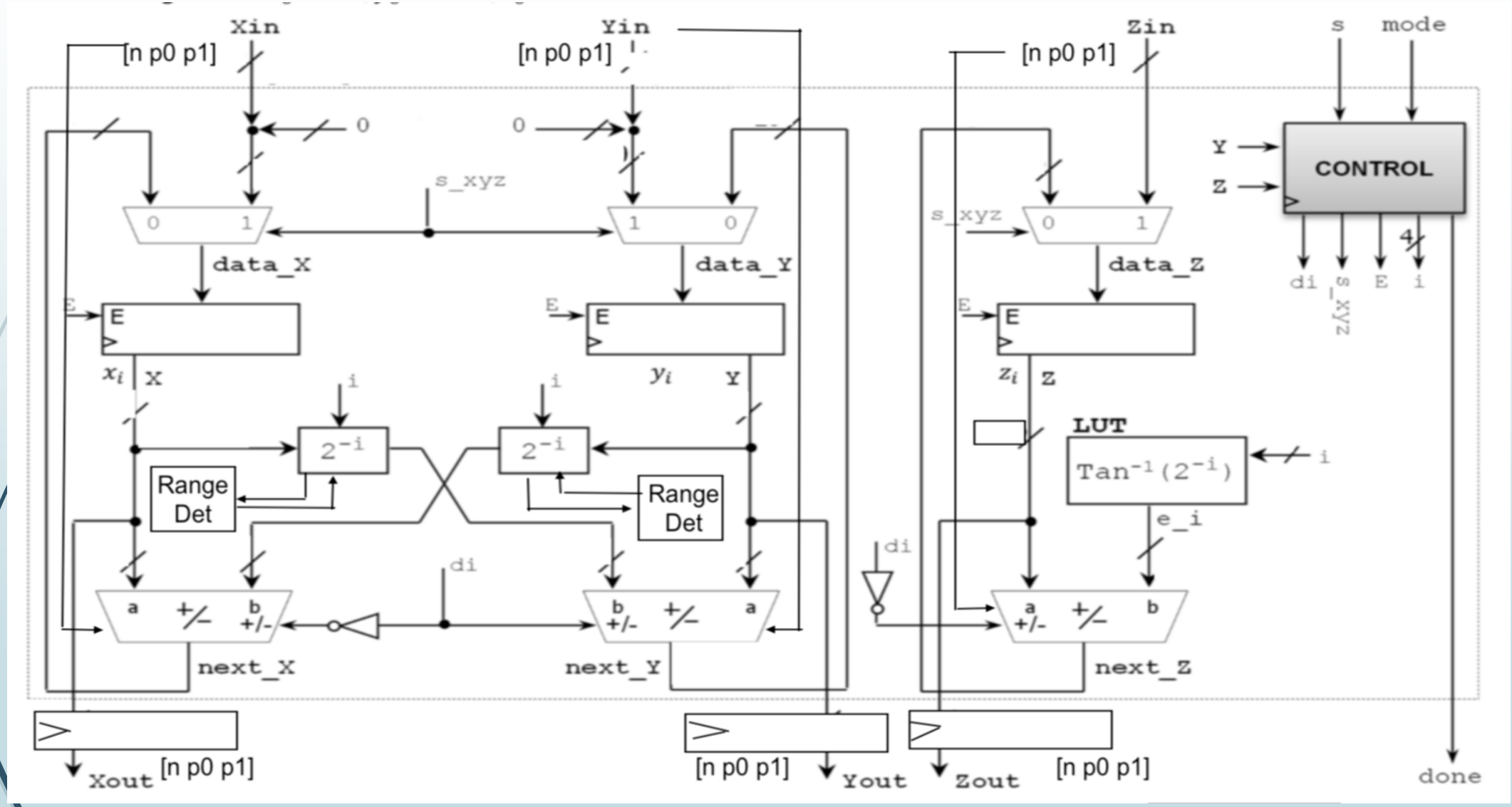
- MATLAB.
- Testbench.
- num0, num1.
- AXI Lite.

VHDL Implementation.

VHDL (DFX ADD/SUB) design



VHDL (CORDIC Design)



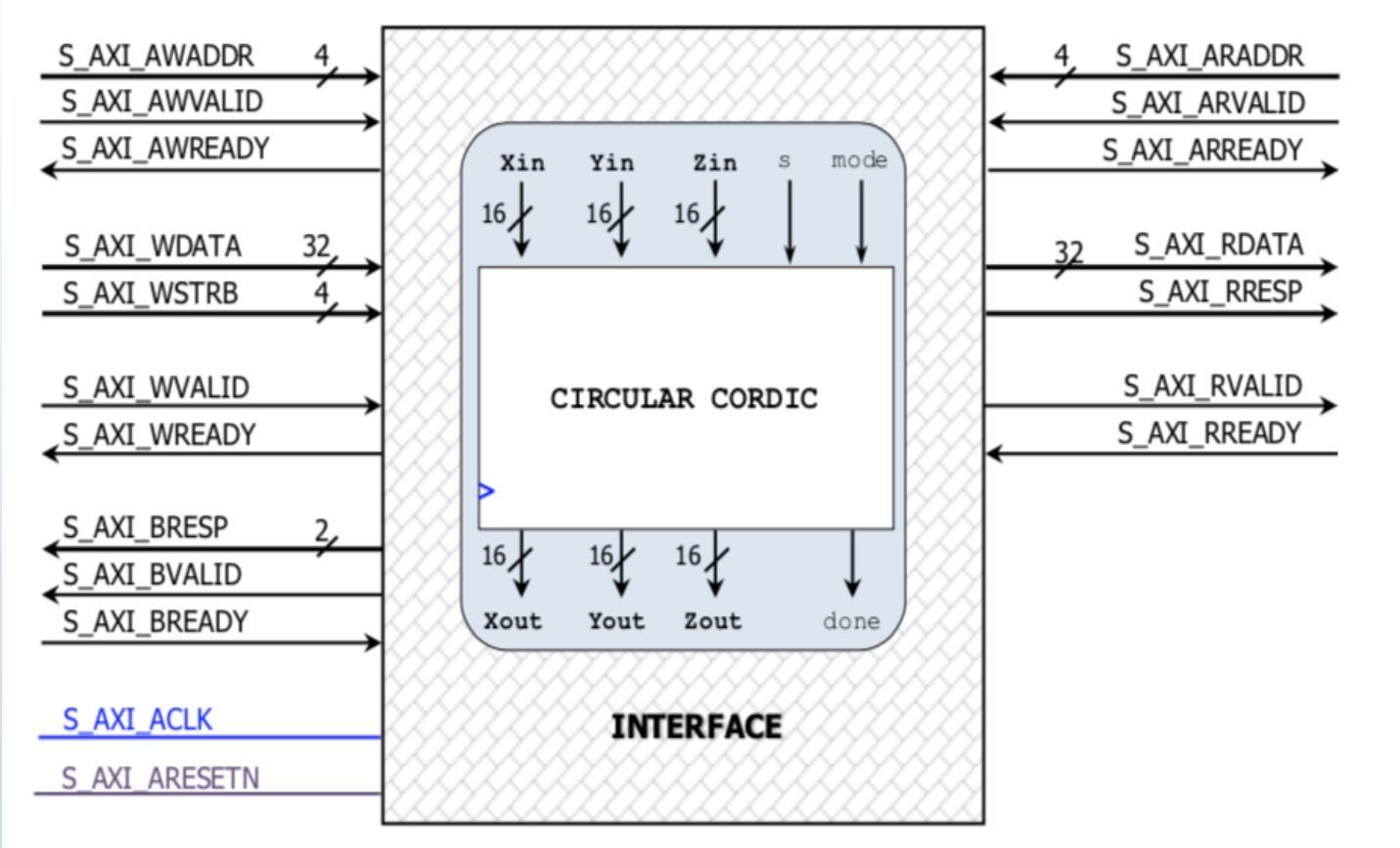
Code:

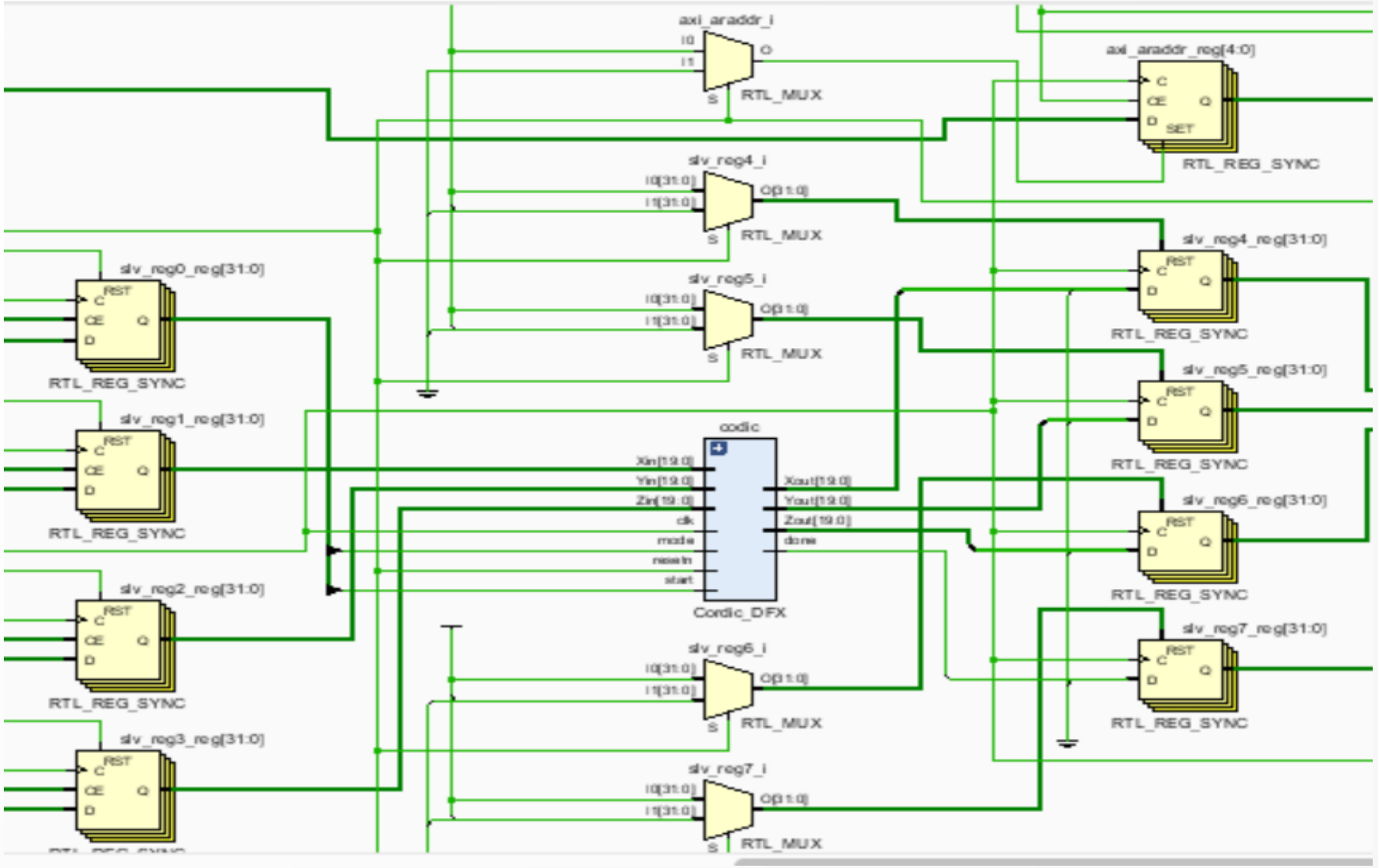
```
-- Calculating X and Y
if X_num = '0' and Y_num = '1' then
    X <= (Shift_right(X,P0 - P1)) + (Shift_right((Y),i));
    Y <= Y - (Shift_right(X,P0 - P1 + i));
    X_num <= '1';
elsif X_num = '1' and Y_num = '0' then
    X <= X + (Shift_right(Y,P0 - P1 + i));
    Y <= (Shift_right(Y,P0 - P1)) - (Shift_right(X,i));
    Y_num <= '1';
else
    X <= X + (Shift_right((Y),i));
    Y <= Y - (Shift_right((X),i));
end if;
```

```
-- Checking the X num0 and num1
   if X_num = '0' then
       if X >= Boundary_value0_max or X < Boundary_value0_min then
           X <= (Shift_right(X,P0 - P1));
           X_num <= '1';
       end if;
   else
       if X < Boundary_value1_max and X >= Boundary_value1_min then
           X <= (Shift_left(X,P0 - P1));
           X_num <= '0';
       end if;
   end if;
```

AXI Lite:

- We used 8 registers.
- We used 20 bits for input and outputs.




axi_araddr_i
I0
I1
O
S
RTL_MUX
axi_araddr_reg[4:0]
C
CE
D
Q
SET
RTL_REG_SYNC
slv_reg4_i
I0[31:0]
I1[31:0]
O[31:0]
slv_reg4_reg[31:0]
RST
slv_reg5_i
slv_reg5_reg[31:0]
slv_reg0_reg[31:0]
slv_reg1_reg[31:0]
slv_reg2_reg[31:0]
slv_reg3_reg[31:0]
codic
Xin[19:0]
Yin[19:0]
Zin[19:0]
clk
mode
nreset
start
Xout[19:0]
Yout[19:0]
Zout[19:0]
done
Cordic_DFX
slv_reg6_i
slv_reg6_reg[31:0]
slv_reg7_i
slv_reg7_reg[31:0]

SDK

- Create a C code that perform the operation to test it. We tested all modes.
- This is done also to verify that the Dual-Fixed point Cordic hardware is correctly running on a real board.

system.hdf | system.mss | helloworld.c

```
// slv_reg1(19 to 0) = Xin
// slv_reg2(19 to 0) = Yin
// slv_reg3(19 to 0) = Zin
MYIP_CORDIC_mWriteReg(XPAR_MYIP_CORDIC_0_S00_AXI_BASEADDR, MYIP_CORDIC_S00_AXI_SLV_REG1_OFFSET, 0x00034065);
MYIP_CORDIC_mWriteReg(XPAR_MYIP_CORDIC_0_S00_AXI_BASEADDR, MYIP_CORDIC_S00_AXI_SLV_REG2_OFFSET, 0x00072437);
MYIP_CORDIC_mWriteReg(XPAR_MYIP_CORDIC_0_S00_AXI_BASEADDR, MYIP_CORDIC_S00_AXI_SLV_REG3_OFFSET, 0x000503bb);

// Start bit0, mode bit4(rotation)
MYIP_CORDIC_mWriteReg(XPAR_MYIP_CORDIC_0_S00_AXI_BASEADDR, MYIP_CORDIC_S00_AXI_SLV_REG0_OFFSET, 0x00000000);
// Start bit0(1), mode bit4(rotation)
MYIP_CORDIC_mWriteReg(XPAR_MYIP_CORDIC_0_S00_AXI_BASEADDR, MYIP_CORDIC_S00_AXI_SLV_REG0_OFFSET, 0x00000001);
// Start bit0(0), mode bit4(rotation)
MYIP_CORDIC_mWriteReg(XPAR_MYIP_CORDIC_0_S00_AXI_BASEADDR, MYIP_CORDIC_S00_AXI_SLV_REG0_OFFSET, 0x00000000);
```

Problems | Tasks | Console | Properties | SDK Terminal

Connected to: Serial (COM11, 115200, 0, 8)

```
X = 60438

Y = D4699

Z = 7FFF9

test2, mode = 0

X = DE38A

Y = 952BA

Z = 7FFF9
```

Send | Clear

SDK Log

```
15:55:48 INFO    : Context for processor 'ps7_cortexa9_0' is
15:55:49 INFO    : Processor reset is completed for 'ps7_cort
15:55:49 INFO    : Context for processor 'ps7_cortexa9_0' is
15:55:49 INFO    : The application 'C:/Users/dhalaswa/Desktop
15:55:49 INFO    : 'configparams force-mem-access 0' command
15:55:49 INFO    : ----------------XSDB Script--------------
connect -url tcp:127.0.0.1:3121
source C:/Users/dhalaswa/Desktop/recproject/jj/project_1/pro
targets -set -nocase -filter {name =~"APU*" && jtag_cable_na
loadhw -hw C:/Users/dhalaswa/Desktop/recproject/jj/project_1
configparams force-mem-access 1
targets -set -nocase -filter {name =~"APU*" && jtag_cable_na
stop
ps7_init
ps7_post_config
targets -set -nocase -filter {name =~ "ARM*#0" && jtag_cable
rst -processor
targets -set -nocase -filter {name =~ "ARM*#0" && jtag cable
```

Results:

- These results are taken from 1000 data tested values.
- Maximum error of all X, Y and Z values after the comparison between our hardware and the CORDIC function in MATLAB.

Output Values	X	Y	Z
Max error in %	0.026%	0.022%	0.0085%

Conclusion:

- Apply the knowledge of DFX arithmetic in coding.
- It is obvious that DFX does almost give accurate results compared with floating numbers with much less operation time.
- Apply the knowledge of different interfaces such as AXI Bus and IP.

Future work:

- Increase the the number of data.
- Test more dual fixed format to find out the most suitable one.
- Do a partial configuration that changes hardware format as needed.

Any Questions?