M.EEC041 - Digital Systems Design

2021/2022

Laboratory project 1 - V1.0
22 March 2022

1. Introduction

This project consists in implementing a digital circuit to calculate the polar coordinates of a vector, given its rectangular coordinates. The function implemented by the system can thus be summarized as:

$$\textit{Given } \boldsymbol{X} \textit{ and } \boldsymbol{Y}\textit{, calculate } \boldsymbol{M} = \sqrt{X^2 + Y^2} \textit{ and } \boldsymbol{A} = atan(^{Y}/_{X})$$

The $\boldsymbol{X}$ and $\boldsymbol{Y}$ inputs are fixed-point signed numbers (two's complement), represented with 16 integer bits and 16 fractional bits. The modulus $\boldsymbol{M}$ is positive fixed-point with 16 integer bits and 16 fractional bits and the angle $\boldsymbol{A}$ is signed in two's complement, with 8 bits for the integer part and 24 bits for the fractional part. The valid input ranges for $\boldsymbol{X}$ and $\boldsymbol{Y}$ is explained below.

The circuit implements the CORDIC algorithm (in translation mode) as a clocked synchronous sequential digital circuit and must be built as a synthesizable Verilog module, representing exactly the logic diagram shown in figure 1. The description of the CORDIC algorithm is beyond the scope of this work, although it will be studied in detail by the end of the semester. The circuit computes the two results $\boldsymbol{M}$ and $\boldsymbol{A}$ by successive approximations in 32 iterations, using a total of 33 clock cycles. To execute a computation the signal **enable** must be set to 1 during 33 clock cycles and **start** must be set to 1 only in the first clock cycle to load the $\boldsymbol{X}$ and $\boldsymbol{Y}$ operands. The testbench provided in **./src/verilog-tb/rec2pol_tb.v** already includes a Verilog task to activate these control signals in the appropriate sequence.

The Verilog interface is:

```
module rec2pol(
           input clock,
           input reset,  // synchronous reset, active high
           input enable, // set and keep high to enable iteration
           input start,  // set to 1 for one clock to start
           input  signed [31:0] x,    // X component, 16Q16
           input  signed [31:0] y,    // Y component, 16Q16
           output signed [31:0] mod,  // Modulus, 16Q16
           output signed [31:0] angle // Angle in degrees, 8Q24
             );
```

(Note: **nQm**: **n+m** word, **n** bits for the integer part and **m** bits for the fractional part)

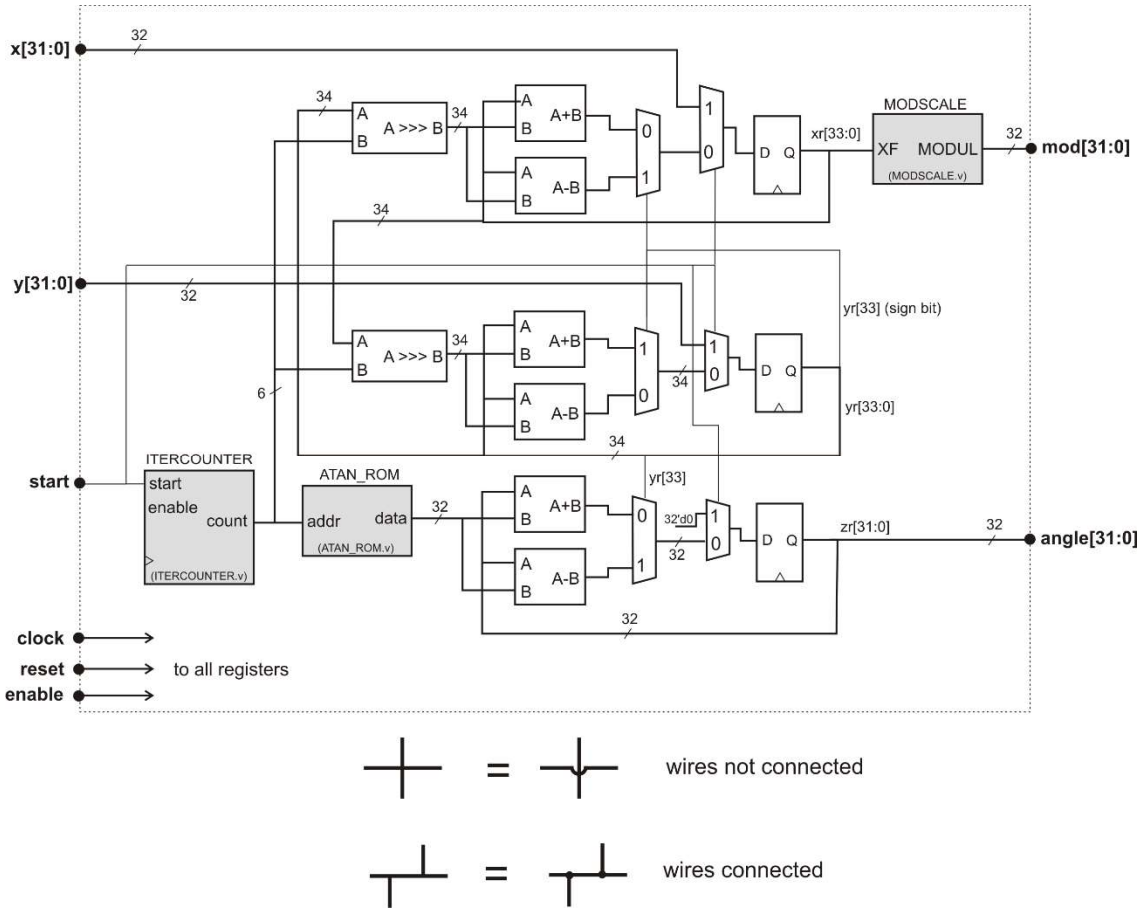

Figure 1 – RTL diagram of the sequential polar coordinate calculator.

2. Implementation

Download and install is your work area the zip archive with the laboratory design kit. The contents of these directories are:

`./doc`	contains documentation related to the project, as this guide.
`./matlab`	Matlab/Octave scripts (CORDIC algorithm and generator of the datafiles located in ./simdata.
`./sim`	create here the QuestaSim simulation project. DO NOT use a subdirectory or the simulation will not run properly.
`./simdata`	datafiles used for simulation, with the data to initialize the **ATAN_ROM** memory.
`./src/verilog-rtl`	the Verilog files for implementation (or the synthesizable code).
`./src/verilog-tb`	the Verilog files for building the testbench.
`./src/verilog-IPs`	the provided IP cores (the gray blocks in the schematic).

The system must be implemented as a behavioral Verilog synthesizable module using a single clock signal for all registers, active in the positive (or rising) edge and a global synchronous reset, active high. The activation of the reset signal must set all the registers to zero.

The 3 shaded blocks shown in the block diagram are provided as IP cores ("*Intelectual Property*") in folder **./src/Verilog-IPs**. Module **ATAN_ROM** implements a ROM (Read-Only Memory) pre-loaded with the values of $\arctan(2^{-i})$, required for the CORDICV algorithm. Module **ITERCOUNTER** implements a 6-bit binary counter and generates the address to the **ATAN_ROM**. Module **MODSCALE** performs a multiplication by a constant to adjust the final result generated by the CORDIC algorithm.

All the other blocks should be modeled by combinational processes using the **assign** or **always @*** statements (the adders, subtractors, multiplexers and shifters) and using sequential processes coded with the **always @(posedge…)** statement (the 3 registers). All registers and the block **ITERCOUNTER** (sequential) must be enabled by the input **enable**.

3. Verification

To verify your Verilog model you must adapt the testbench given and implement a convenient verification program to maximize the functional coverage of your code. The task **execcordic** already included in the testbench implements the correct sequence of signals **start** and **enable** to perform a rectangular to polar conversion, including a text output statement (**$display()**) to print the results in fractional decimal. Improve the testbench in order to automate the verification procedure.

4. Additional developments

4.1 The current implementation only performs correctly for vectors in the 1st and 4th quadrants (X positive and output angle in the range [-90º , +90º]). Without modifying the initial implementation, create a new module that allows the rectangular to polar conversion along the 4 quadrants. The new module should instantiate the **rec2pol** module and implement the additional functionality, using at most one additional clock cycle.

4.2 The present implementation requires the external circuit to keep the enable signal high while the iterative process is running (during exactly 33 clock cycles). Make a new version of the **ITERCOUNTER** block that generates internally the signal **enable** for itself and for the 3 registers and outputs a new signal **ready** meaning the system is able to initiate a new computation. While the computation is begin performed, **ready** should be set to zero. This new module should receive only the **start** pulse, set to 1 during a single clock period. Additionally, if a computation is running (**ready==0**) the input **start** must be ignored.