Preliminary Design: LFM Signal Generator

Alex • 05.23.2018

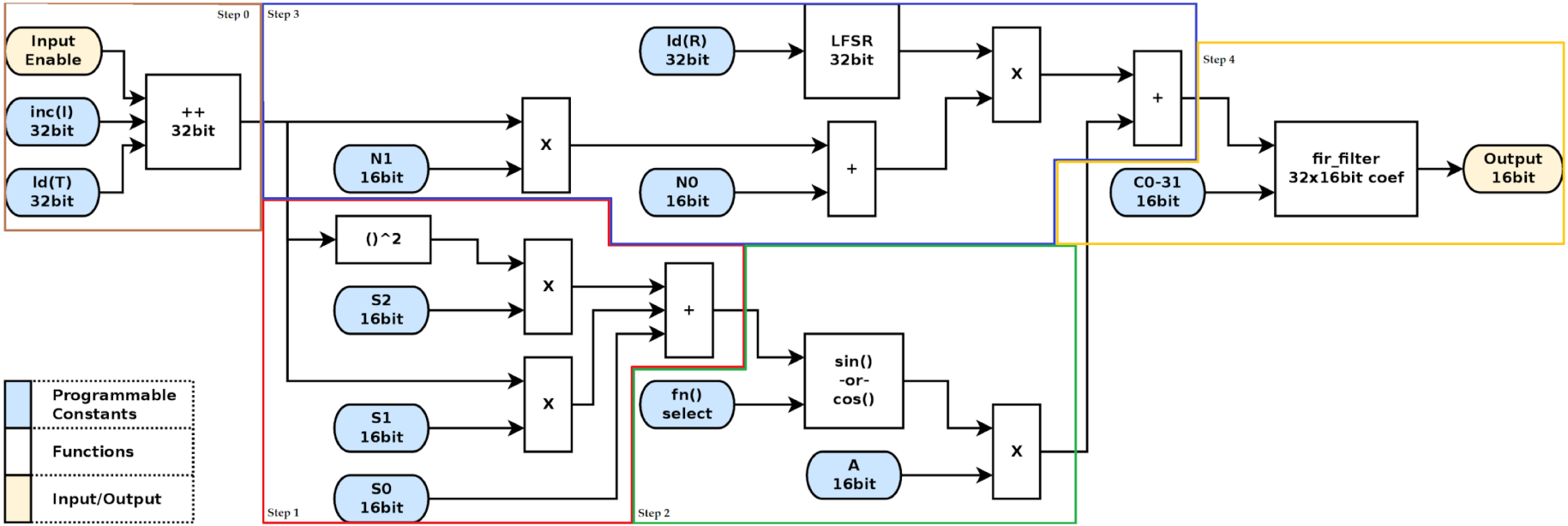
Overview

Goal: Generate a Chirp signal

Want to create a signal modeled by:

$y = \sin(s_2 * t^2 + s_1 * t + s_0)$

Block Diagram

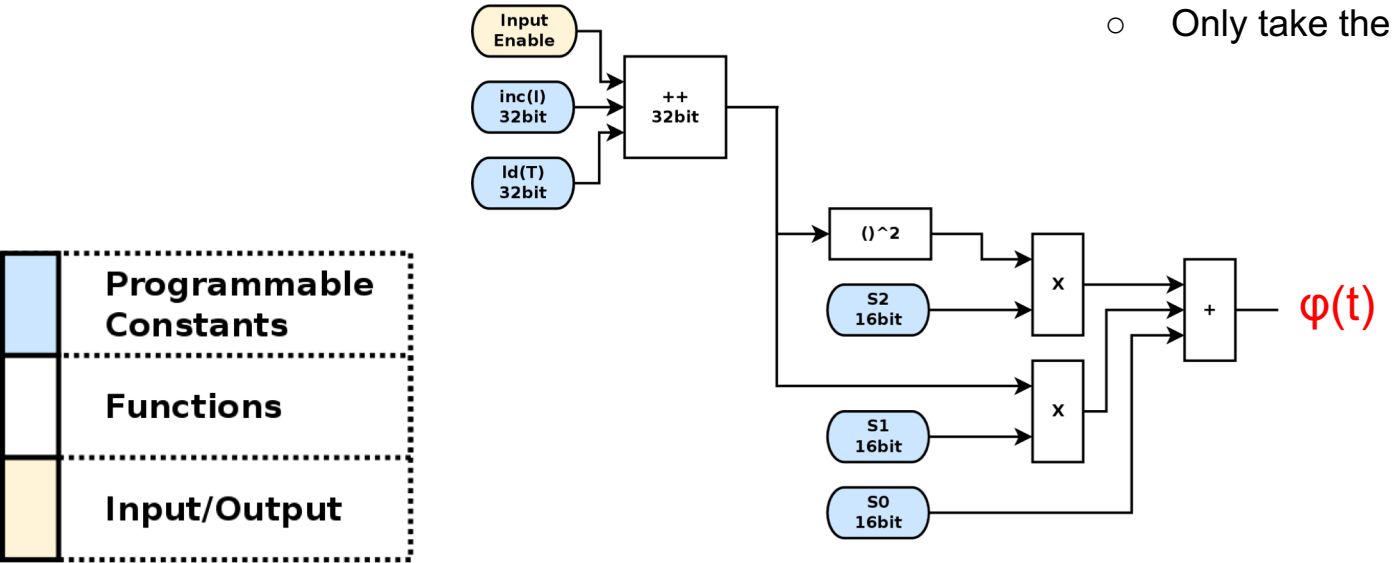


Step 0: Incrementing t

- **Initialize two integers**
  - “count” -- Gets value from ld(T)
  - “inc” -- Gets value from inc(I)

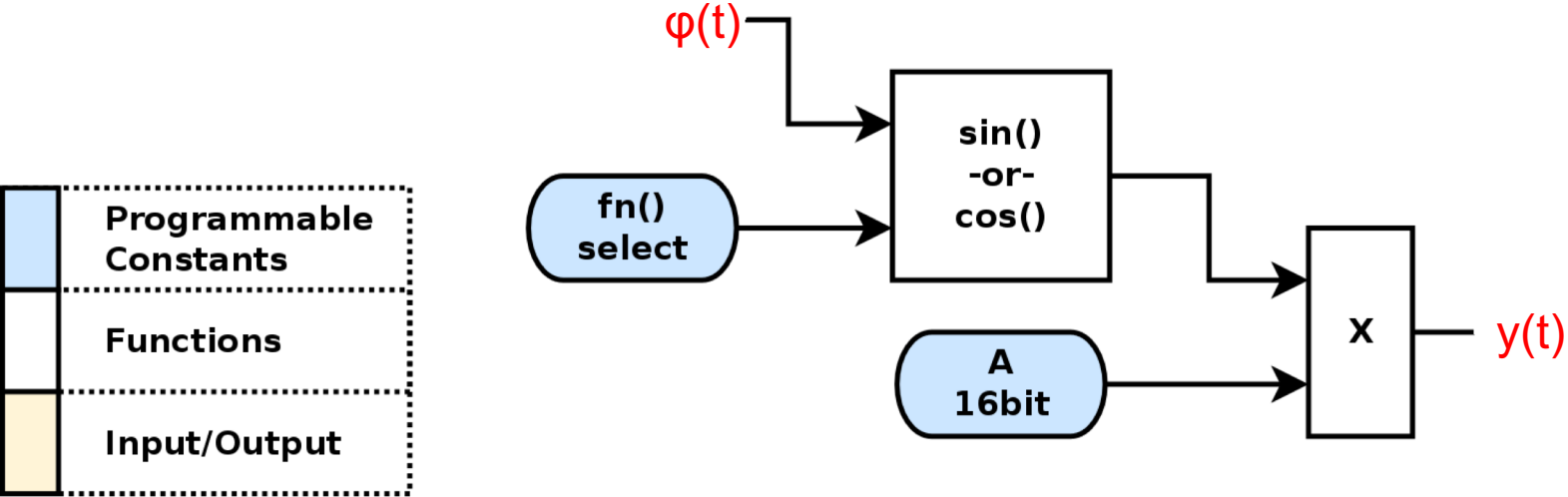
Step 1: Set Up φ(t)

- **$\varphi(t) = s_2 \times t^2 + s_2 \times t + s_0$**
- **Bit Lengths**
  - $s_2 \times t^2$ requires 80 bits
  - φ(t) needs 81 bits total
  - Only take the 32 MSB after summation



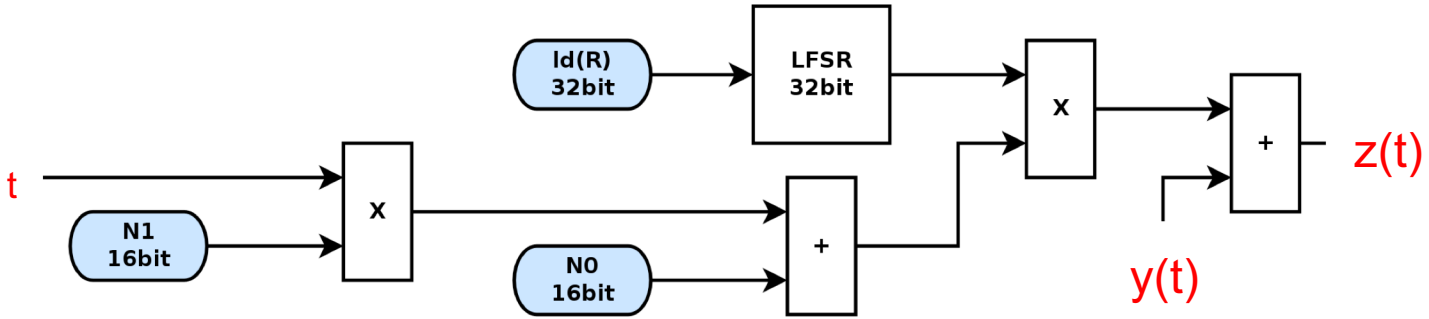
Step 2: Perform y(t) = A * sin(φ(t)) or cos(φ(t))

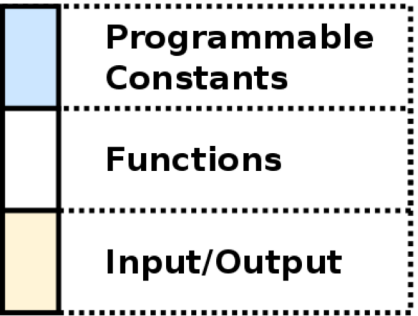
- **Implement using CORDIC**
  - fn() select determines sin() or cos() at build
    - CORDIC does support switching after build, however
  - Use TCL script to generate CORDIC core for sin or cos
  - Latency of CORDIC is calculated in Vivado



Step 3: Noise

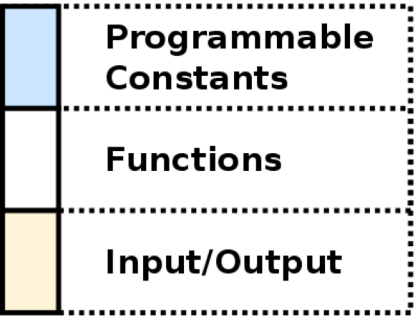
- **Generate Noise (N)**
  - Use t, N1, N0, and Adam's LFSR
- **Create z(t) = y(t) + N**

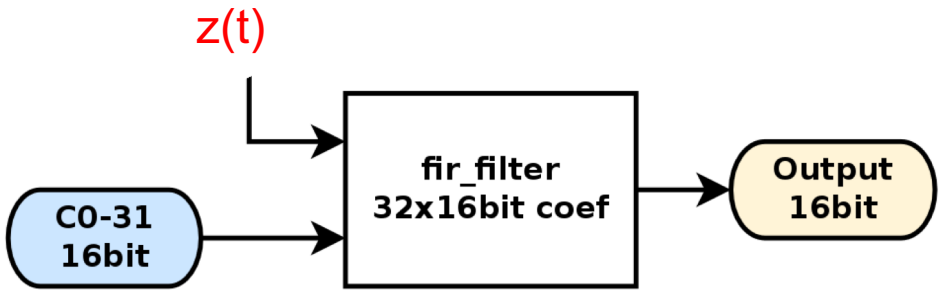




Step 4: Filtering

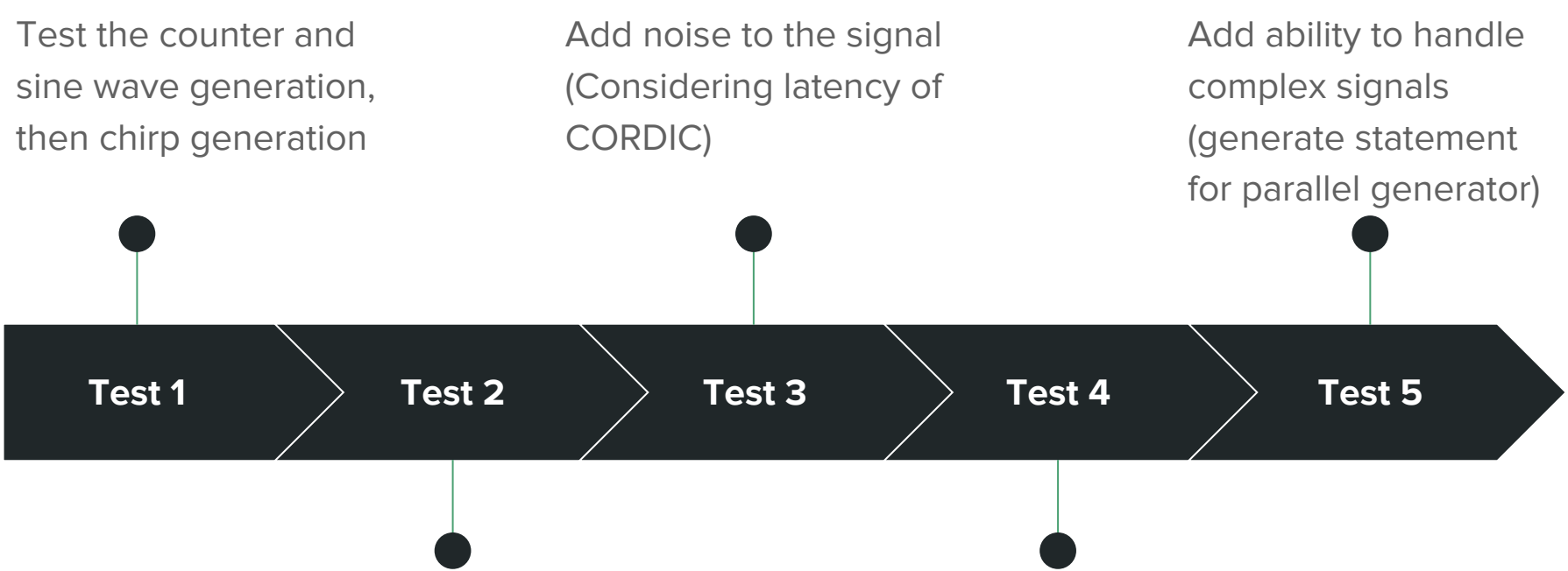
- **Use a FIR filter to remove noise**
  - Use FINS to include the FIR Filter IP





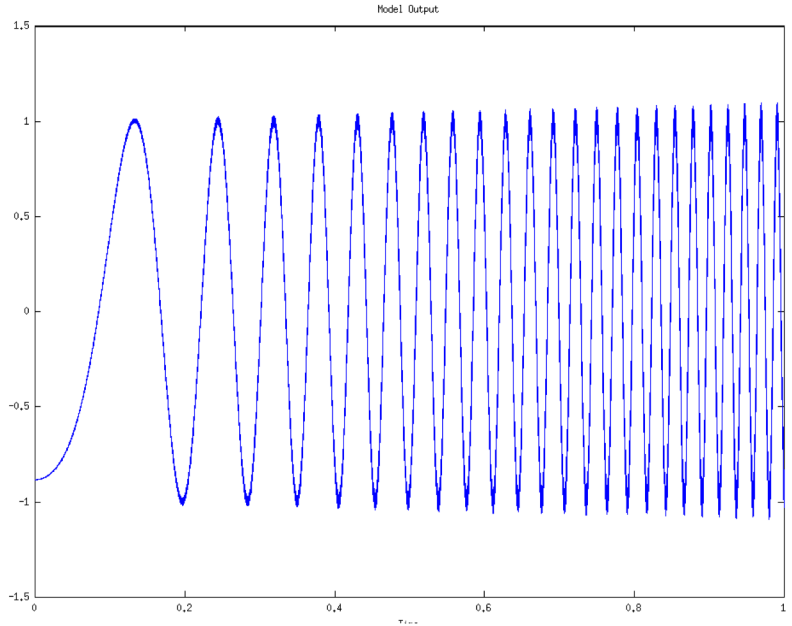
Bit Widths

- Will need to scale after multiplications
  - Trying not to exceed 32-bits throughout the process
  - Use the FINS.json to grab scaling factors and decide fractional bits
- FIR Filter input width
  - Parameter in JSON file
  - Test with input width = output width


Test the counter and sine wave generation, then chirp generation
Add noise to the signal (Considering latency of CORDIC)
Add ability to handle complex signals (generate statement for parallel generator)
Test 1
Test 2
Test 3
Test 4
Test 5
Try Chirp generation
Add the FIR filter, testing lowpass, highpass, and bandpass

Model Results

Starting at t=0



Starting at t=0.5