README_RTL: Decimation Filter

Module Hierarchy and Simulation Procedures

Team 94

Contents

1 Project Overview **2**

2 Project File Inventory **2**

2.1 RTL & Testbench Files 2

2.2 Coefficient Files (Memory) 2

2.3 Input/Output Simulation Files 2

2.4 Generated Simulation Files 2

3 Module Hierarchy **2**

4 Architecture and Data Flow **3**

4.1 Stage 1: CIC Filter (`cic_filter.v`) 3

4.2 Stage 2: FIR Compensation Filter (`fir_filter.v`) 3

4.3 Stages 3 & 4: Halfband Filters (`halfband_filter.v`) 3

5 Simulation Steps **3**

5.1 Input File Configuration 3

5.2 Execution Commands 3

5.3 GTKWave Setup 4

1 Project Overview

This folder implements a multi-stage **Digital Decimation Filter** for a Delta-Sigma ADC. The system processes a high-speed, low-resolution modulator bit-stream to produce high-resolution, low-speed PCM output. The architecture achieves a total decimation factor of **128** (R_{total}) using a three-stage topology: **CIC (**$R = 16$**)** $\rightarrow$ **FIR (**$R = 2$**)** $\rightarrow$ **Halfband (**$2 \times 2 = 4$**)**.

2 Project File Inventory

The project folder contains the following source codes, memory files, and simulation data:

2.1 RTL & Testbench Files

- `decimation_filter.v`: Top-level module wiring the CIC, FIR, and Halfband stages together.
- `cic_filter.v`: Stage 1 implementation (Cascaded Integrator-Comb) for coarse decimation.
- `fir_filter.v`: Stage 2 implementation (FIR) for droop compensation.
- `halfband_filter.v`: Module for Stage 3 and Stage 4 decimation.
- `tb_decimation_filter.v`: Testbench for clock generation, file I/O, and verification.

2.2 Coefficient Files (Memory)

- `fir_coeffs.mem`: Hexadecimal coefficients for the Stage 2 FIR filter.
- `hb1_coeffs.mem`: Hexadecimal coefficients for the Stage 3 Halfband filter.
- `hb2_coeffs.mem`: Hexadecimal coefficients for the Stage 4 Halfband filter.

2.3 Input/Output Simulation Files

- `modulator_output1.txt`: Primary input bitstream generated by the Delta-Sigma Modulator.
- `modulator_output2.txt`: Alternative input dataset for testing different signal conditions.
- `modulator_output3.txt`: Third input dataset option.
- `output_filters.txt`: The filtered PCM output corresponding to `modulator_output1.txt`.

2.4 Generated Simulation Files

- `filters.out`: The compiled executable binary generated by Icarus Verilog from the source files.
- `wave_decimation_filter.vcd`: The Value Change Dump file containing signal transitions for waveform analysis.

  NOTE: Both files are for the modulator_output1.txt input file.

3 Module Hierarchy

The top-level module (`decimation_filter.v`) integrates the specific filter stages. The testbench drives this module with file I/O operations.

- `tb_decimation_filter.v`: Testbench responsible for Clock/Reset generation and File I/O.
  - `decimation_filter.v`: Top-level wrapper.
  - `cic_filter.v`: Stage 1 (Coarse Decimation).
  - `fir_filter.v`: Stage 2 (Droop Compensation).
  - `halfband_filter.v`: Stages 3 & 4 (Final filtering, instances *u_hb1, u_hb2*).

4 Architecture and Data Flow

4.1 Stage 1: CIC Filter (`cic_filter.v`)

The Cascaded Integrator-Comb (CIC) filter performs the initial coarse decimation ($R = 16, N = 15, M = 1$) without hardware multipliers.

- **Data Flow:** 5-bit Input $\rightarrow$ 65-bit Output.
- **Detail:** To prevent overflow, the internal width is $Width = Input + N \times \log_2(R \times M) = 65$ bits.

4.2 Stage 2: FIR Compensation Filter (`fir_filter.v`)

This stage corrects the frequency response droop caused by the CIC stage and provides further decimation ($R = 2$) using 26 taps.

- **Data Flow:** 65-bit Input $\rightarrow$ 50-bit Output (Truncated).
- **Detail:** Implements a parallel Multiply-Accumulate (MAC) architecture.

4.3 Stages 3 & 4: Halfband Filters (`halfband_filter.v`)

Two identical instances ($R = 2$ each, 7 Taps) are used to reach the final Nyquist rate.

- **Data Flow:** 50-bit Input $\rightarrow$ 50-bit Output.
- **Detail:** Uses a **Finite State Machine (FSM)** for serial operations to optimize area.

5 Simulation Steps

The project is verified using **Icarus Verilog v12.0** and analyzed with **GTKWave Analyzer v3.3.100** in Visual Studio Code.

5.1 Input File Configuration

The provided input files (`modulator_output1/2/3`) are derived from the upstream Modulator.

- **Default Setup:** The testbench is currently configured to read `modulator_output1.txt`.
- **Reference Output:** The file `output_filters.txt` included in the folder is the pre-generated output for this default input.
- **Changing Inputs:** To test with `modulator_output2` or `3`, open `tb_decimation_filter.v`, locate the file read command (around line 88), and update the filename:

```
    // Example change:
    input_file = $fopen("modulator_output2.txt", "r");
```

  Note: For custom inputs, refer to the README_MATLAB.

5.2 Execution Commands

1. Compilation

```
iverilog -o filters.out *.v
```

Explanation: The ‘iverilog‘ command compiles all Verilog source files (‘*.v‘) in the directory into a simulation executable named ‘filters.out‘.

2. Simulation

```
vvp .\filters.out
```

Explanation: The ‘vvp‘ (Verilog Virtual Processor) command runs the compiled executable. It prints the simulation progress, sample counts, and decimation results to the console.

3. Waveform Analysis

```
gtkwave .\wave_decimation_filter.vcd
```

Explanation: This command launches the GTKWave GUI to visualize the signal waveforms stored in the ‘.vcd‘ file generated during the simulation.

5.3 GTKWave Setup

1. **Hierarchy:** In the SST panel, navigate to `tb_decimation_filter` → `dut`.
2. **Signals:** Drag the following to the viewer:
   - **Input:** `clk`, `in_valid`, `in_data`.
   - **Stage 1:** `dut.cic_out_valid` (Observe rate drop by 16).
   - **Stage 2:** `dut.fir_out_valid` (Observe rate drop by 2).
   - **Output:** `out_valid`, `out_data` (Final Output).
3. **Verification:** Confirm that `out_valid` asserts once for every 128 input valid pulses.