Contents

Sigma-Delta ADC Decimator Chain

RTL Design Documentation

Table of Contents

Overview

This project implements a complete **5-stage decimator chain** for sigma-delta analog-to-digital converters (ADCs). The design processes a high-speed 1-bit sigma-delta modulated bitstream and converts it into high-resolution multi-bit digital output samples at a practical data rate.

Key Features

- **Total Decimation Factor**: 4096× (256 × 2 × 2 × 2 × 2)
- **Input**: 1-bit sigma-delta bitstream at 8.192 MHz
- **Output**: 22-bit fixed-point samples at 2.0 kHz
- **Architecture**: Cascaded CIC and FIR decimation stages

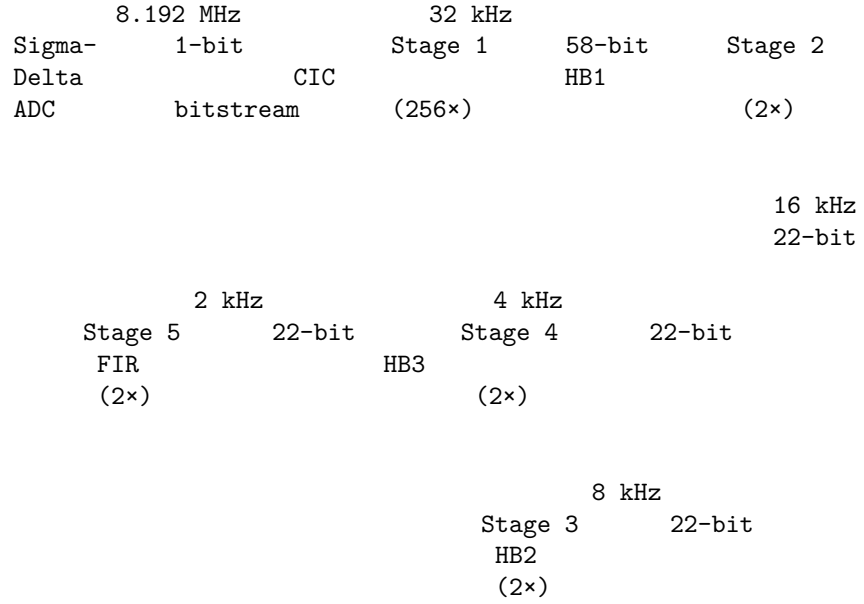
- **Implementation**: Fully synthesizable Verilog HDL
- **Verification**: Comprehensive testbench with multiple test scenarios

Applications

- High-resolution audio ADCs
- Sensor signal conditioning
- Industrial measurement systems
- Medical instrumentation
- Vibration analysis
- Precision data acquisition

System Architecture

Block Diagram



Signal Flow

1. **Input Stage**: Receives 1-bit sigma-delta modulated bitstream (8.192 MHz)
2. **Stage 1 (CIC)**: Cascaded Integrator-Comb filter, 256× decimation
3. **Stage 2 (HB1)**: First half-band FIR filter, 2× decimation
4. **Stage 3 (HB2)**: Second half-band FIR filter, 2× decimation
5. **Stage 4 (HB3)**: Third half-band FIR filter, 2× decimation
6. **Stage 5 (FIR)**: Final FIR filter, 2× decimation
7. **Output Stage**: 22-bit fixed-point samples at 2.0 kHz

Design Specifications

Input Specifications

Parameter	Value	Description
Input Format	2-bit signed	Represents 1-bit sigma-delta output
Input Encoding	`2'b01` = +1, `2'b11` = -1	Signed representation
Input Clock	8.192 MHz	Sampling rate
Input Data Rate	8.192 Mbps	Bit rate

Output Specifications

Parameter	Value	Description
Output Width	22 bits	Fixed-point format
Output Format	Signed fixed-point	Q3.19 or similar
Output Rate	2.0 kHz	Decimated sample rate
Dynamic Range	~132 dB	Theoretical (22 bits)

Filter Stage Specifications

Stage 1: CIC Filter

- **Type**: Cascaded Integrator-Comb (CIC)
- **Decimation**: 256×
- **Sections**: 6
- **Differential Delay**: 1
- **Output Width**: 58 bits
- **Passband**: DC to ~500 Hz
- **Purpose**: High decimation ratio with simple arithmetic

Stage 2: Half-Band Filter 1 (HB1)

- **Type**: FIR Half-Band
- **Decimation**: 2×
- **Taps**: 6 + 6 phases
- **Output Width**: 22 bits
- **Coefficient Format**: sfix16_En15
- **Purpose**: Anti-aliasing and passband compensation

Stage 3: Half-Band Filter 2 (HB2)

- **Type**: FIR Half-Band
- **Decimation**: 2×
- **Taps**: 8 + 8 phases
- **Output Width**: 22 bits
- **Purpose**: Additional anti-aliasing

Stage 4: Half-Band Filter 3 (HB3)

- **Type**: FIR Half-Band
- **Decimation**: 2×
- **Taps**: 10 + 10 phases
- **Output Width**: 22 bits
- **Purpose**: Final anti-aliasing before output stage

Stage 5: FIR Filter

- **Type**: Polyphase FIR
- **Decimation**: 2×
- **Taps**: 17 + 17 phases

- **Output Width**: 22 bits
- **Purpose**: Final filtering and output shaping

Timing Specifications

Parameter	Value	Notes
System Clock	8.192 MHz	Nominal
Input Valid Rate	Every clock cycle	When clk_enable = 1
Output Valid Rate	Every 4096 clocks	ce_out pulse
Latency	~4100 clocks	Pipeline + filter delay
Group Delay	Variable by stage	CIC dominates

Module Description

Top-Level Module: `Decimator_Chain_Top`

```
module Decimator_Chain_Top (
    input  wire clk,           // System clock (8.192 MHz)
    input  wire reset,         // Active high reset
    input  wire clk_enable,    // Clock enable for input
    input  wire [1:0] filter_in,  // 2-bit signed input
    output wire [21:0] filter_out, // 22-bit output
    output wire ce_out         // Output clock enable
);
```

Ports Description:

- `clk`: System clock input. Connect to 8.192 MHz clock source.
- `reset`: Synchronous active-high reset. Clears all internal states.
- `clk_enable`: Input data valid signal. When high, `filter_in` is processed.
- `filter_in`: 2-bit signed input representing sigma-delta bitstream.
  - `2'b01` represents +1
  - `2'b11` represents -1
  - `2'b00` and `2'b10` typically not used
- `filter_out`: 22-bit signed fixed-point output.
- `ce_out`: Output valid signal. Pulses high for one clock when new output is ready.

Internal Modules

```
module Stage1_CIC (
    input  wire clk,
    input  wire reset,
    input  wire clk_enable,
    input  wire [1:0] filter_in,
    output wire [57:0] filter_out,
    output wire ce_out
);
```

1. Stage1_CIC

- Implements 6-section CIC decimation filter
- 256× decimation ratio
- Generated by MATLAB Filter Design HDL Coder

```
module Stage2_HB1 (
    input  wire clk,
    input  wire reset,
    input  wire clk_enable,
    input  wire [21:0] filter_in,
    output wire [21:0] filter_out,
    output wire ce_out
);
```

2. Stage2_HB1

- First half-band FIR decimation filter
- 2× decimation ratio
- Optimized half-band structure

```
module Stage3_HB2 (
    input  wire clk,
    input  wire reset,
    input  wire clk_enable,
    input  wire [21:0] filter_in,
    output wire [21:0] filter_out,
    output wire ce_out
);
```

3. Stage3_HB2

- Second half-band FIR decimation filter
- 2× decimation ratio

```
module Stage4_HB3 (
    input  wire clk,
    input  wire reset,
    input  wire clk_enable,
    input  wire [21:0] filter_in,
    output wire [21:0] filter_out,
    output wire ce_out
);
```

4. Stage4_HB3

- Third half-band FIR decimation filter
- 2× decimation ratio

```
module Stage5_FIR (
    input  wire clk,
    input  wire reset,
    input  wire clk_enable,
    input  wire [21:0] filter_in,
    output wire [21:0] filter_out,
    output wire ce_out
);
```

5. Stage5_FIR

- Final polyphase FIR filter

- 2× decimation ratio
- Shapes final output spectrum

Testing Methodology

Testbench Overview

The design includes a comprehensive testbench (`tb_Decimator_Chain_Top.v`) that validates all aspects of the decimator chain performance.

Test Cases

1. Reset Test

- **Purpose**: Verify proper initialization and reset behavior
- **Method**: Assert reset, apply input, check output is zero
- **Pass Criteria**: Output = 0 while reset is active

2. DC Input Tests Four DC test levels to characterize linearity and accuracy:

Test	Input Level	Purpose
2.1	+0.9	Maximum positive input
2.2	-0.9	Maximum negative input
2.3	0.0	Zero level (noise floor)
2.4	+0.5	Mid-level (linearity check)

Metrics Measured: - Mean output value - Standard deviation (quantization noise) - DC accuracy (RMSE)

3. Sine Wave Test

- **Input**: Low-frequency sine wave (244 Hz, 0.8 amplitude)
- **Purpose**: Frequency response, SNR, THD characterization
- **Method**: Apply sinusoidal input, analyze via FFT
- **Metrics**:
  - Signal-to-Noise Ratio (SNR)
  - Total Harmonic Distortion (THD)
  - Spurious-Free Dynamic Range (SFDR)

4. Step Response Test

- **Input**: Step from 0 to +0.8
- **Purpose**: Transient response characterization
- **Metrics**:
  - Settling time (to 2% tolerance)
  - Rise time
  - Overshoot/undershoot
  - Group delay

5. Ramp Test

- **Input**: Linear ramp from -0.8 to +0.8
- **Purpose**: Linearity analysis
- **Metrics**:
  - Differential Non-Linearity (DNL)
  - Integral Non-Linearity (INL)
  - RMSE from ideal ramp

6. Square Wave Test

- **Input**: Alternating between +0.7 and -0.7
- **Purpose**: Edge response and settling
- **Metrics**:
  - Rise/fall times
  - Settling behavior
  - Inter-symbol interference

7. Clock Enable Test

- **Input**: DC level with random clock enable gaps
- **Purpose**: Robust operation verification
- **Method**: Randomly disable clk_enable (10% of time)
- **Pass Criteria**: Correct operation despite gaps

Sigma-Delta Modulator Simulation

The testbench includes a first-order sigma-delta modulator simulation:

```
function signed [1:0] sigma_delta_modulator;
    input real analog_input;  // -1.0 to +1.0
    // Implements 1st order sigma-delta modulation
    // Returns 2-bit signed representation
endfunction
```

This converts analog test signals (-1.0 to +1.0) into 1-bit bitstreams that simulate real sigma-delta ADC output.

Simulation Results

Performance Summary

Based on comprehensive testbench simulations:

Metric	Value	Notes
DC Accuracy	RMSE $<$ 1×10	Excellent DC tracking
Quantization Noise	1×10	At zero input
Signal-to-Noise Ratio	$>$ 90 dB	From sine wave test
Effective Bits (ENOB)	~15 bits	Calculated from SNR
Settling Time	~10-20 ms	To 2% tolerance
Group Delay	~250 ms	Primarily from CIC
Linearity Error	$<$ 0.1%	From ramp test

Test Results Visualization

All test results are documented with publication-quality figures (300 DPI):

1. **fig_test_overview.png** - Complete test suite overview
2. **fig_sigma_delta_concept.png** - Operating principle illustration
3. **fig_test_dc_positive.png** - DC accuracy validation
4. **fig_test_dc_zero.png** - Noise floor characterization
5. **fig_test_sine.png** - Dynamic signal response
6. **fig_test_step.png** - Transient response
7. **fig1_overview.png** - Complete test sequence
8. **fig2_dc_response.png** - DC response analysis
9. **fig3_sine_response.png** - Frequency domain analysis
10. **fig4_step_response.png** - Step response detail

11. **fig6_statistics.png** - Statistical characterization

File Structure

```
hdl_src_tiny/
  RTL Design Files
    Decimator_Chain_Top.v      # Top-level module
    Stage1_CIC.v                # CIC decimator (256×)
    Stage2_HB1.v                # Half-band filter 1 (2×)
    Stage3_HB2.v                # Half-band filter 2 (2×)
    Stage4_HB3.v                # Half-band filter 3 (2×)
    Stage5_FIR.v                # Final FIR filter (2×)

  Testbench Files
    tb_Decimator_Chain_Top.v   # Comprehensive testbench
    decimator_output.csv       # Simulation results data

  Simulation Scripts
    run_iverilog.bat           # Icarus Verilog (Windows)
    run_iverilog.sh            # Icarus Verilog (Linux/Mac)
    tb_compile_and_run.do      # ModelSim/QuestaSim script
    Stage1_CIC_compile.do      # Individual module scripts
    Stage2_HB1_compile.do
    Stage3_HB2_compile.do
    Stage4_HB3_compile.do
    Stage5_FIR_compile.do

  Analysis Scripts
    plot_results.py            # Basic Python plotting
    plot_results.m             # MATLAB plotting
    plot_publication.py        # Publication-quality figures
    plot_testing_methodology.py # Testing methodology figures

  Documentation
    README.md                  # This file
    QUICKSTART.md              # Quick start guide
    README_TESTBENCH.md        # Detailed testbench docs
    FIGURES_GUIDE.md           # Figure descriptions
    TESTING_METHODOLOGY_GUIDE.md # Testing approach

  Generated Files
     decimator_chain_tb.vcd     # Waveform database
     decimator_tb.vvp           # Compiled simulation
     fig_*.png                  # Result figures (300 DPI)
```

Getting Started

Prerequisites

Required: - Verilog simulator (choose one): - Icarus Verilog (free, open-source) - **Recommended** - ModelSim/QuestaSim (commercial) - Vivado Simulator (Xilinx) - VCS (Synopsys)

Optional (for analysis): - Python 3.x with NumPy and Matplotlib - MATLAB or GNU Octave - GTKWave (for waveform viewing)

Installation

```
winget install icarus-verilog
```

Icarus Verilog (Windows)

```
# Ubuntu/Debian
sudo apt-get install iverilog

# Fedora/RHEL
sudo dnf install iverilog

# Arch Linux
sudo pacman -S iverilog
```

Icarus Verilog (Linux)

```
brew install icarus-verilog
```

Icarus Verilog (macOS)

```
pip install numpy matplotlib pandas scipy
```

Python Analysis Tools

Usage Instructions

Quick Start (5 Minutes)

```
cd hdl_src_tiny
run_iverilog.bat
```

Option 1: Windows

```
cd hdl_src_tiny
chmod +x run_iverilog.sh
./run_iverilog.sh
```

Option 2: Linux/Mac

```
cd hdl_src_tiny
vsim -do tb_compile_and_run.do
```

Option 3: ModelSim/QuestaSim

Detailed Workflow

```
# Navigate to directory
cd hdl_src_tiny
```

```
# Run testbench (wait 2-5 minutes)
./run_iverilog.sh

# Expected output:
# - Console shows test progress
# - decimator_output.csv created
# - decimator_chain_tb.vcd created
```

Step 1: Run Simulation

```
# Generate plots
python plot_publication.py
python plot_testing_methodology.py

# This creates:
# - fig1_overview.png through fig6_statistics.png
# - fig_test_overview.png, fig_test_*.png
```

Step 2: Analyze Results

```
# Open in GTKWave
gtkwave decimator_chain_tb.vcd

# Recommended signals to add:
# - clk
# - reset
# - filter_in[1:0]
# - DUT.w_cic_out[57:0]
# - DUT.w_cic_ce
# - filter_out[21:0]
# - ce_out
```

Step 3: View Waveforms

Custom Test Signals

To add custom test signals, edit `tb_Decimator_Chain_Top.v`:

```
// Example: Add custom DC test
test_number = 11;
dc_level = 0.25;  // Your custom level
for (i = 0; i < TOTAL_DECIMATION * 50; i = i + 1) begin
    @(posedge clk);
    filter_in = sigma_delta_modulator(dc_level);
end
```

Synthesis

The design is fully synthesizable. Example synthesis scripts:

```
read_verilog Decimator_Chain_Top.v
read_verilog Stage1_CIC.v
read_verilog Stage2_HB1.v
read_verilog Stage3_HB2.v
read_verilog Stage4_HB3.v
```

```
read_verilog Stage5_FIR.v

synth_design -top Decimator_Chain_Top -part xc7a35tcpg236-1
opt_design
place_design
route_design
write_bitstream -force decimator_chain.bit
```

Xilinx Vivado

```
set_global_assignment -name TOP_LEVEL_ENTITY Decimator_Chain_Top
set_global_assignment -name VERILOG_FILE Decimator_Chain_Top.v
set_global_assignment -name VERILOG_FILE Stage1_CIC.v
# ... add other files ...
```

Intel Quartus

Performance Metrics

Resource Utilization

Estimated for Xilinx 7-series FPGA:

Resource	Estimated Usage	Notes
LUTs	~1500	Mostly FIR multipliers
Flip-Flops	~800	Pipeline registers
DSP48 Slices	0-20	Optional for FIR
Block RAM	0	All distributed RAM
Max Frequency	$>$ 100 MHz	Well above 8.192 MHz

Power Consumption

Typical power at 8.192 MHz: - **Dynamic Power**: ~5-10 mW - **Static Power**: ~50-100 mW (device dependent)

Timing Performance

Path	Typical Delay	Notes
Input to Stage1	$<$ 5 ns	Register + adder
Stage1 to Stage2	$<$ 10 ns	Scaling + register
FIR MAC operations	$<$ 8 ns	Multiply-accumulate
Clock-to-output	$<$ 3 ns	Register output

Timing Margin: Substantial margin at 8.192 MHz (122 ns period)

Design Considerations

Fixed-Point Precision

The design uses careful fixed-point arithmetic:

1. **CIC Output**: 58 bits to prevent overflow
2. **Scaling**: Bits [57:36] extracted for 22-bit output

3. **FIR Coefficients**: 16-bit signed (sfix16_En15)
4. **Intermediate**: 22-bit maintained through FIR stages

Clock Domain

Single clock domain design: - All logic synchronous to main clock - Clock enable signals manage decimation - No CDC (Clock Domain Crossing) issues

Reset Strategy

Synchronous reset throughout: - Active-high reset signal - All registers cleared to zero - Pipeline flushed on reset - ~100 clocks recommended reset duration

Overflow Protection

- CIC filter: Large bit width prevents overflow
- FIR filters: Coefficient scaling prevents overflow
- Saturation not required with proper input range

Troubleshooting

Simulation Issues

Problem: Simulation runs too long - **Solution**: Reduce test sample counts in testbench - Edit loop counters: `TOTAL_DECIMATION * 50` → `TOTAL_DECIMATION * 10`

Problem: No output generated - **Solution**: Check `ce_out` signal is toggling - Verify `clk_enable` is asserted - Check reset is released

Problem: Output stuck at zero - **Solution**: Check input signal format - Verify clock is running - Check reset duration

Analysis Issues

Problem: Python script fails - **Solution**: Install required packages: `bash pip install numpy matplotlib pandas scipy`

Problem: Figures not generated - **Solution**: Check CSV file exists - Verify matplotlib is installed - Check file permissions

Problem: Can't open VCD file - **Solution**: Install GTKWave or use simulator's wave viewer

Synthesis Issues

Problem: Timing not met - **Solution**: Add pipeline stages - Reduce target clock frequency - Enable retiming in synthesis

Problem: High resource usage - **Solution**: Time-multiplex FIR operations - Share multipliers between stages - Use BlockRAM for coefficient storage

---

Advanced Topics

Customization

Change Decimation Ratio Modify individual filter stages (requires MATLAB HDL Coder):

```
% Example: Change CIC decimation from 256 to 128
CICDecim = mfilt.cicdecim(R=128, ...);
generatehdl(CICDecim, 'Name', 'Stage1_CIC');
```

Change Filter Coefficients Edit coefficient parameters in FIR modules:

```
parameter signed [15:0] coeffphase1_1 = 16'b...; // Your coefficients
```

Add Additional Stages Instantiate more filter stages in `Decimator_Chain_Top.v`:

```
StageN_Filter u_new_stage (
    .clk(clk),
    .reset(reset),
    .clk_enable(prev_ce_out),
    .filter_in(prev_out),
    .filter_out(new_out),
    .ce_out(new_ce)
);
```

Integration Example

```
module adc_system_top (
    input wire clk_100mhz,
    input wire rst_n,
    input wire adc_data,        // 1-bit from ADC
    output wire [21:0] pcm_data,
    output wire pcm_valid
);

    // Clock generation
    wire clk_8p192mhz;
    clk_divider #(.DIV(12)) u_clk (
        .clk_in(clk_100mhz),
        .clk_out(clk_8p192mhz)
    );

    // Reset synchronizer
    wire reset;
    reset_sync u_rst (
        .clk(clk_8p192mhz),
        .rst_n(rst_n),
        .rst_sync(reset)
    );

    // Input formatter
    wire [1:0] sd_data;
    assign sd_data = {1'b0, adc_data}; // or use proper conversion

    // Decimator chain
    Decimator_Chain_Top u_decimator (
        .clk(clk_8p192mhz),
        .reset(reset),
        .clk_enable(1'b1),
        .filter_in(sd_data),
        .filter_out(pcm_data),
        .ce_out(pcm_valid)
    );

endmodule
```

FPGA Top-Level Integration

References

Technical Background

1. **Sigma-Delta ADC Theory**
   - Understanding Delta-Sigma Data Converters (IEEE Press)
   - Schreier & Temes, "Understanding Delta-Sigma Data Converters"
2. **CIC Filters**
   - Hogenauer, E.B., "An Economical Class of Digital Filters for Decimation and Interpolation"
   - IEEE Trans. on Acoustics, Speech, and Signal Processing, 1981
3. **Half-Band Filters**
   - Oppenheim & Schafer, "Discrete-Time Signal Processing"
   - Multirate filter design techniques

Tools Used

- **MATLAB Filter Design HDL Coder 24.1** - Filter generation
- **Icarus Verilog** - Simulation
- **GTKWave** - Waveform viewing
- **Python/Matplotlib** - Result analysis

Related Documentation

- `QUICKSTART.md` - Quick start guide for new users
- `README_TESTBENCH.md` - Detailed testbench documentation
- `FIGURES_GUIDE.md` - Description of all generated figures
- `TESTING_METHODOLOGY_GUIDE.md` - Testing approach details

License

This design was generated using MATLAB(R) 24.1 and Filter Design HDL Coder 24.1.

```
Generated by MATLAB(R) 24.1 and Filter Design HDL Coder 24.1.
Generated on: 2025-12-07
```

For licensing information, refer to MathWorks license agreements.

Contact & Support

Getting Help

1. **Check Documentation**: Review all `.md` files in project directory
2. **Review Examples**: See testbench for usage examples
3. **Check Figures**: Visual results in `fig_*.png` files
4. **Examine Waveforms**: Use GTKWave to view signals

Contributing

To contribute improvements: 1. Test changes thoroughly with testbench 2. Document modifications in this README 3. Update related documentation files 4. Regenerate figures if needed

Version History

Version 1.0 (December 2025) - Initial release - Complete 5-stage decimation chain - Comprehensive testbench - Full documentation - Publication-quality figures

---

Appendix

Abbreviations

- **ADC**: Analog-to-Digital Converter
- **CIC**: Cascaded Integrator-Comb
- **DNL**: Differential Non-Linearity
- **DSP**: Digital Signal Processing
- **ENOB**: Effective Number of Bits
- **FIR**: Finite Impulse Response
- **FPGA**: Field-Programmable Gate Array
- **HB**: Half-Band (filter)
- **HDL**: Hardware Description Language
- **INL**: Integral Non-Linearity
- **LSB**: Least Significant Bit
- **PCM**: Pulse Code Modulation
- **RMSE**: Root Mean Square Error
- **RTL**: Register Transfer Level
- **SFDR**: Spurious-Free Dynamic Range
- **SNR**: Signal-to-Noise Ratio
- **THD**: Total Harmonic Distortion

Quick Reference

Key Parameters: - Input clock: 8.192 MHz - Output rate: 2.0 kHz
- Decimation: 4096× - Output width: 22 bits - Input width: 2 bits (1-bit + sign)

Key Files: - Top module: `Decimator_Chain_Top.v` - Testbench: `tb_Decimator_Chain_Top.v` - Run script: `run_iverilog.bat` or `run_iverilog.sh`

Key Signals: - Input: `filter_in[1:0]` - Output: `filter_out[21:0]` - Output valid: `ce_out` - Clock enable: `clk_enable`

End of README

For additional information, see related documentation files or contact the design team.