

Tiny Tap $\Delta\Sigma$ ADC Decimator Chain (5-Stage)

1 Overview

This project implements a 5-stage, area-optimized digital decimation chain for a 1-bit Delta–Sigma Modulator (DSM) and also generates Verilog HDL for each stage plus a simple top-level wrapper.

Target specification:

- Output sample rate: 2 kHz
- Oversampling ratio (OSR): 4096 \Rightarrow modulator rate = 8.192 MHz
- Goal: > 16 ENOB with minimal logic / multipliers
- Architecture:

$$\text{CIC}(256) \rightarrow \text{HB}(2) \rightarrow \text{HB}(2) \rightarrow \text{HB}(2) \rightarrow \text{FIR}(2)$$

The MATLAB script is a complete flow from DSM simulation \rightarrow filtering \rightarrow ENOB estimation \rightarrow HDL generation.

2 System Architecture

Overall signal path:

```
Analogue Input (sine @ ~50 Hz)
|
4th-Order Modulator (1-bit)
|   Fs = 8.192 MHz
v
1-bit Bitstream (\pm1)
|
v
[Stage 1] CIC Decimator    R = 256
|   (no multipliers)
v
[Stage 2] Halfband FIR    R = 2
|
v
[Stage 3] Halfband FIR    R = 2
|
v
[Stage 4] Halfband FIR    R = 2
|
v
[Stage 5] Final FIR LPF   R = 2
|
v
Output @ 2 kHz
```

Total decimation factor is

$$256 \times 2 \times 2 \times 2 \times 2 = 4096 = \text{OSR}.$$

The last FIR is deliberately made small (around 27 taps) by relaxing its transition band and letting the three halfband filters provide most of the out-of-band attenuation.

3 Software Requirements

You need the following in MATLAB:

1. **Delta–Sigma Toolbox** (Richard Schreier)
 - Functions used: `synthesizeNTF`, `realizeNTF`, `stuffABCD`, `simulateDSM`.
 - Add the toolbox folder to the MATLAB path before running.
2. **DSP System Toolbox**
 - Objects: `dsp.CICDecimator`, `dsp.FIRDecimator`.
 - Filter design: `fdesign.decimator`, `design`.
3. **Fixed-Point Designer**
 - Fixed-point types: `fi`, `nnumerictype`.
4. **HDL Coder**
 - HDL generation using `generatehdl`.

Any reasonably recent MATLAB version that supports the above functions should work.

4 Files

Typical minimal setup:

- `tiny_tap_dsadc.m`: the main MATLAB script.
- `hdl_src_tiny/` (created automatically):
 - `Stage1_CIC.v`
 - `Stage2_HB1.v`
 - `Stage3_HB2.v`
 - `Stage4_HB3.v`
 - `Stage5_FIR.v`
 - `Decimator_Chain_Top.v` (top-level wrapper)

5 How to Run

1. Open MATLAB.
2. Add required folders to the MATLAB path:
 - The folder containing `tiny_tap_dsadc.m`.
 - The folder containing the Delta–Sigma Toolbox.
3. Check that the required toolboxes (DSP System Toolbox, Fixed-Point Designer, HDL Coder) are installed and licensed.
4. Run the script:

```
tiny_tap_dsadc % or whatever you named the .m file
```

5. Expected behaviour:
 - The command window prints:

- Stage information (tap counts, effective multipliers).
- Final SNDR and ENOB.
- A MATLAB figure shows the output PSD.
- A folder `hdl_src_tiny/` is created with Verilog files.

6 Script Flow

6.1 System Specifications

Key parameters in the script:

- `Fs_out = 2e3`: final output sample rate (2 kHz).
 - `OSR = 4096`: oversampling ratio.
 - `Fs_in = Fs_out * OSR`: modulator sampling rate (8.192 MHz).
 - `DSM_Order = 4`: 4th-order DSM is synthesised for these specs.
- The output band is approximately 0–1 kHz (Nyquist frequency at 2 kHz).

6.2 Input Signal and DSM Simulation

A near-50 Hz sinusoidal input is generated:

- `Target_Fin = 50` (Hz).
- `Amp = 0.5` (full-scale = 1).

The tone frequency is quantised to an FFT bin to obtain a cleaner SNDR measurement. The Delta–Sigma Toolbox is used as follows:

- `ntf = synthesizeNTF(DSM_Order, OSR, 1)`.
- `[a, g, b, c] = realizeNTF(ntf)`.
- `ABCD = stuffABCD(a, g, b, c)`.
- `v_bitstream = simulateDSM(u, ABCD)`.
- The 0/1 bitstream is converted to bipolar ±1 using `v_bipolar_all = 2*double(v_bitstream) - 1`.

6.3 Chunking and Padding

To avoid very large intermediate arrays, the DSM output is processed in chunks:

- `ChunkSize = 100 * OSR`: samples per chunk at the modulator rate.
- If the length of the DSM output is not an exact multiple of `ChunkSize`, the script pads with zeros.
- The number of chunks is given by `NumChunks = length(v_bipolar_all) / ChunkSize`.

The steady-state FFT window is not affected by padding because only the central part of the output is used for spectral analysis.

6.4 Filter Chain Definition

The filter chain is stored in a cell array `Filters{1..5}`:

1. Stage 1 – CIC decimator:
 - `Filters{1} = dsp.CICDecimator(256, 1, 6)`.
 - Decimation factor $R = 256$, differential delay = 1, number of sections $N = 6$.
 - No multipliers; large internal gain $256^6 \approx 2^{48}$.
2. Stage 2 – Halfband 1:
 - Decimation by 2, relaxed transition width ($TW = 0.15$), about 10 taps.
 - Designed using `fdesign.decimator` and `design` with the '`halfband`' option.
3. Stage 3 – Halfband 2:
 - Decimation by 2, slightly tighter transition width ($TW = 0.1$).

4. Stage 4 – Halfband 3:
 - Decimation by 2, even tighter transition width ($TW = 0.08$).
5. Stage 5 – Final FIR:
 - Decimation by 2, low-pass FIR.
 - Passband edge = 0.35 (normalised), stopband edge = 0.65.
 - Stopband attenuation ≈ 96 dB.
 - About 27 taps (reduced from ~ 49 by relaxing the transition band).

Halfband filters exploit symmetry, so the effective number of multipliers is roughly half the tap count.

6.5 Fixed-Point Configuration

The script uses fixed-point arithmetic to reflect realistic hardware behaviour.

- CIC stage:
 - Internal section wordlength: 58 bits.
 - Output is integer (`OutputFractionLength = 0`).
 - CIC gain handled via `CIC_Gain_Bits = 48`.
- FIR stages:
 - Coefficients: Q1.15 (16-bit).
 - Products: 38-bit; accumulators: 54-bit.
 - Output: Q4.18 with `DATA_WIDTH = 22` and `FRAC_WIDTH = 18`.

This configuration comfortably supports around 16 ENOB without overflow.

6.6 Processing Loop

For each input chunk:

1. Convert the input to a 2-bit signed fixed-point type:

```
sig = fi(chunk_in, 1, 2, 0);
```

2. Pass the signal through all five stages:

```
for k = 1:NumStages
    sig = step(Filters{k}, sig);
    if k == 1
        val_norm = double(sig) * (2^(-CIC_Gain_Bits)) * 0.85;
        sig = fi(val_norm, 1, DATA_WIDTH, FRAC_WIDTH);
    end
end
```

3. Collect the decimated samples in `y_all`.

A MATLAB waitbar periodically shows progress during the loop.

6.7 SNDR and ENOB Calculation

The script estimates SNDR and ENOB as follows:

- Discard transient samples (`N_transient`).
- Apply a Hann window and compute the FFT of the remaining samples.
- Find the main signal bin and sum power in a small band of bins around it.

- Sum the remaining power (excluding a few low-frequency bins) as noise+distortion.

ENOB is computed from SNDR using the standard relation:

$$\text{ENOB} = \frac{\text{SNDR[dB]} - 1.76}{6.02}.$$

The final SNDR and ENOB values are printed in the command window.

6.8 Plot

The script produces a MATLAB figure:

- The one-sided power spectral density (PSD) of the decimated output at 2 kHz.
- The signal bins are highlighted to visually separate the tone from the noise floor.

7 HDL Generation

At the end of the script, Verilog HDL is generated for each stage and for a top-level wrapper.

7.1 Stage HDL

The script uses `generatehdl` for each filter:

- Stage1_CIC
- Stage2_HB1
- Stage3_HB2
- Stage4_HB3
- Stage5_FIR

Testbench generation is disabled to keep the output directory simpler, and coefficient multipliers are encoded using CSD for area savings.

7.2 Top-Level Wrapper

The wrapper `Decimator_Chain_Top.v` has the following interface:

```
module Decimator_Chain_Top (
    input wire clk, reset, clk_enable,
    input wire [1:0] filter_in,
    output wire [21:0] filter_out,
    output wire ce_out
);
    ...
endmodule
```

Key points:

- `filter_in[1:0]` is the (scaled) digital input corresponding to the DSM bitstream (represented as 2-bit fixed-point in the script).
- `filter_out[21:0]` is the Q4.18 output at 2 kHz.
- `ce_out` is asserted when a new output sample is valid.
- Internal clock-enable signals chain from one stage to the next.
- CIC scaling in hardware is implemented by taking the upper bits, e.g. `w_cic_scaled = w_cic_out[57:36]`.

This wrapper can be instantiated directly in an FPGA or ASIC design to interface the 1-bit DSM front-end with the decimation chain.

8 Customization Guide

Common customisation options:

1. Output rate and OSR

- Adjust `Fs_out` and `OSR` in the script.
- Re-synthesise the NTF and re-check CIC gain and filter responses.

2. Modulator order

- Change `DSM_Order`; higher order improves noise shaping but can affect stability.

3. Input tone and amplitude

- Change `Target_Fin` and `Amp` to test ENOB/SFDR across the band.

4. Filter specifications

- Modify `N` and `TW` for halfbands, and passband/stopband edges for the final FIR, to trade off area vs. performance.

5. Fixed-point wordlengths

- Adjust `DATA_WIDTH`, `FRAC_WIDTH`, CIC wordlengths and accumulator sizes for your target technology and ENOB requirement.

9 Notes and Tips

- If ENOB drops below the target:
 - Increase final FIR stopband attenuation.
 - Tighten halfband transitions (more taps).
 - Increase accumulator precision.
- If MATLAB reports unknown functions:
 - Check that the Delta-Sigma Toolbox is on the path.
 - Verify that DSP System Toolbox, Fixed-Point Designer and HDL Coder are installed.
- If HDL generation fails:
 - Confirm HDL Coder licence.
 - Ensure that the filter System objects are supported by `generatehdl`.