ECE M216A Final Project – Fall 2025

Hardware Realization of a MASH-111 ΔΣ Modulator

Deadline: Sunday Dec 7th 2025 11:59PM

Motivation:

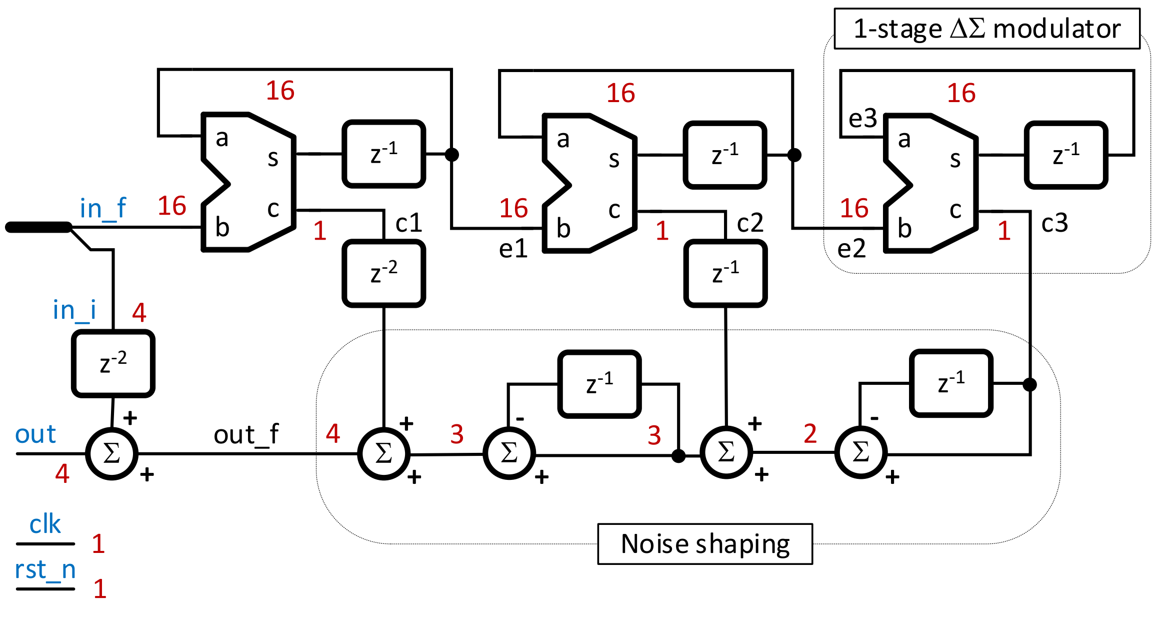
ΔΣ modulators are widely used in data converters and phased-locked loops (PLLs). Particularly, in a PLL there is often a need to divide the frequency by a fractional number. While integer dividers can be readily implemented with flipflops in feedback loops, fractional division is often challenging, and is very commonly done using a digital ΔΣ modulator to control the modulus (divide ratio) of an integer divider. Varying the integer divide ratio periodically leads to a fractional division on average over time, as intended.

Project Statement:

Implement a MASH-111 (Multi-stAge-noise-SHaping) ΔΣ modulator in 16nm CMOS operating at 500MHz clock rate, with the lowest possible area and power consumption.

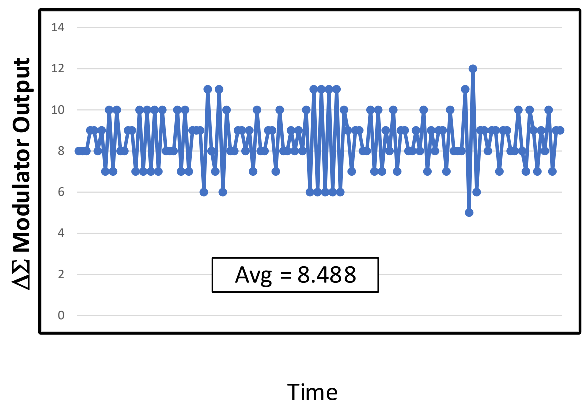
ΔΣ Modulator Basic Functionality:

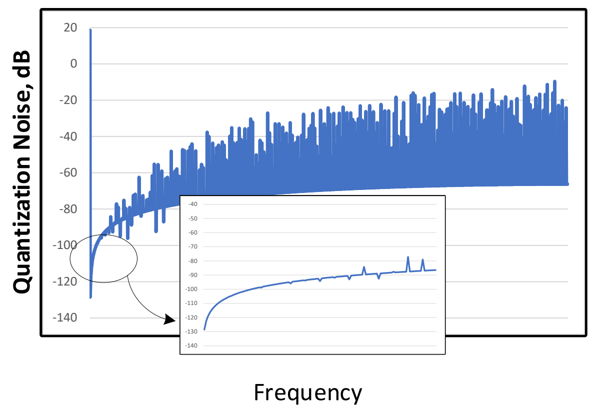
The block diagram of the 3rd-order ΔΣ modulator is shown below. As the name suggests, the modulator consists of a series of delay cells (Δ) and adders (Σ). Each delay cell (e.g. z^{-2} block responsible for *two cycles delay*) is simply realized with D flipflops.



Each of the 16-bit wide adders placed in a feedback loop, creates an *accumulator* which can be thought of an integrator, with a transfer function of $\frac{z^{-1}}{1-z^{-1}}$. The carry bit of each adder effectively acts like a *1-bit quantizer*, while the sum output represents the *error signal* fed into the next accumulator (e1 and e2). The first accumulator takes the factional input (in_f). A MASH-111 architecture as the name suggests, comprise of 3 1-bit stages cascaded.

The 3 1-bit quantizers' outputs (c1, c2, and c3) effectively create the intended variation of the divide ratio such that on average the desired fraction is created. The input consists of a 4-bit integer (in_i or N), and a 16-bit fractional part (in_f or f), representing the number $N.f = in\_i + \frac{in\_f}{2^{16}-1}$. For instance, $in\_i = 4d'8$, and $in\_f = 16d'32000$ is equivalent to a divide ratio of 8.488 averaged over many cycles. A sample of the modulator output over 128 clock cycles is shown below on the left. The divider accordingly divides the frequency by anywhere from 5 to 12, such that on average it will be 8.488. The fractional output (out_f) can vary between -3 and +4, which is then added to the integer part (8 in this example) to create the final 4-bit output (out).





Due to the crude quantization (a 16-bit in_f reduces to a 3-bit $\{c3, c2, c1\}$), even though the intended frequency is created, large amount of jitter is produced as well. To circumvent that, the quantizers are fed into a series of differentiators (transfer function of $1 - z^{-1}$) and adders (as shown on the bottom of the block diagram) to *shape the noise*. As such, the quantization noise which is inherently white, is shaped such that most of its energy is concentrated at high frequencies, which can be subsequently filtered through the PLL. The FFT of the output in dB from 0 to $\frac{f_{clk}}{2}$ is shown on the right. The spike at DC represents the average divide ratio (8.488 or 18.576dB in this example), while the noise shaping is clearly evident. In a well-designed PLL, the bandwidth (along with the order and clock frequency of the ΔΣ modulator) are chosen such that the modulator contributes negligible jitter.

For various parts of the modulator, the corresponding bit widths are shown in red. As always, care must be taken to avoid overflow or erroneous arithmetic by proper bit/sign extension.

Note how the delays are balanced throughout the chain.

<u>Helpful tips:</u>

A few hints and useful discussion points:

- Following the math for the noise shaper, the final fractional output (out_f) can vary between -3 and +4 (c1, c2, and c3 are the carry signals and as such, are either 0 or 1). Despite being 8 levels, this requires 4 bits in a binary 2's complement format. Attention must be paid to bit extension, for instance zeros can be placed into MSBs as needed along the path. Alternatively, sign extension may be may done for the carry signals (e.g. $\{2\{c3\}\}$ for the 3rd accumulator carry output), which will now make them vary between 0 (00) and -1 (11). Accordingly, the fractional output will vary now between -4 and +3, which will be 3 bits, but must be *subtracted* from the integer input to balance the sign. This leads to a cleaner and more efficient (why?) design.

- The integer input must be restricted to 3 to 11 (why?). This is often not an issue in a practical PLL.
- Can you identify the critical path(s) in the design? If the clock period is T_{clk}, what are the constraints on the flipflops and adders timing?
- The fact that the fractional input is 16 bits arises from the need for a certain required frequency resolution for a given application. What are the trade-offs of having a better resolution (more bits) for the input? Do all the accumulators need to have the same resolution (or width)?

For more information, take a look at:

MASH-111 ΔΣ modulators: https://ieeexplore.ieee.org/document/4799178

Original fractional PLL paper: https://ieeexplore.ieee.org/document/229400

Top-Level Verilog Module:

The top-level wrapper **M216A_TopModule.v** containing the required I/O

The I/O signals are shown in blue, and are as follows:

Inputs:

in_i	4-bit input representing the integer part (ranges from 3 to 11)
in_f	16-bit input representing the fractional part (ranges from 0 to 65535)
clk	1-bit clock, this is the system master clock
rst_n	1-bit reset, if *low*, clear all the occupied widths to 0

Output:

out	4-bit output

Design Constraints:

- Clock frequency is 500MHz.
- Minimize area and energy (energy = power / clock frequency).

Project Testbench:

A testbench, **M216A_Testbench.v**, must be created and submitted along with top module and other files. The test bench will simply apply the clock and a DC input (e.g. , $in\_i = 4d'8$, and $in\_f = 16d'32000$) and observes the correct output is produced. Your final submitted code will be additionally tested with our test bench as well to ensure that the correct algorithm is implemented.

Project Deliverables:

- Design and performance summary document entitled: **Group_N.pdf** (1-page document, 12-pt font, 1-inch margins)
    - N indicates your group number
    - Include the names of all group members at the beginning of this document
    - List the following specs: Area, Power, Hold Time slack
    - Provide a top-level block diagram of your design and its key elements
    - List key features of your design
- All files below should be packed into one ZIP file (file name: **Group_N.zip**)
    - N indicates your group number
    - Top-Level Verilog Module (**M216A_TopModule.v**)
    - Your test bench (**M216A_Testbench.v**)
    - Your synthesis script (**Group_N.tcl**)
    - Synthesized Area and Power Reports (Reference: Lab2)
        - Files: **Group_N.Area, Group_N.Power**
    - Synthesized Timing Report (Slack) for your maximum clock frequency
        - Make sure you do not have any hold time violations
        - Files: **Group_N.TimingSetup, Group_N.TimingHold**

Project Submission:

Upload the following files to canvas by **11:59PM on Dec 7th, 2025**. All the files should be uploaded to bruinlearn before the deadline to be considered as a valid submission.

Project Timeline:

Week 6	-Understand the algorithm and clarify all the queries -Students to declare project teams by Nov 8th, 11:59PM (ProjectSignUp- TBD)
Week 7-8	-Implement the given algorithm in Verilog -Verify the functionality against your testbench
Weeks 8-9	-Optimize the design -Minimize area and power
Week 10	-Inspect for timing violations -Fix any setup and hold time violations and generate the timing reports