Imperial College
London

Lecture 11

Echo Synthesizer & Challenges Explained

Peter Cheung
Department of Electrical & Electronic Engineering
Imperial College London

URL: www.ee.imperial.ac.uk/pcheung/teaching/E2_CAS/
E-mail: p.cheung@imperial.ac.uk

Lecture Objectives

- To revisit some of the issues that came up during the laboratory experiments
- To provide some guidelines on how to perform diagnosis when things don’t work
- To provide explanations on Lab 6
- To explain how the ADC works
- To explain some of the major modules used in the experiment
- To explain the idea of offset binary vs 2’s complement
- To explain the ALLPASS module and its use
- To explain how echo may be synthesized

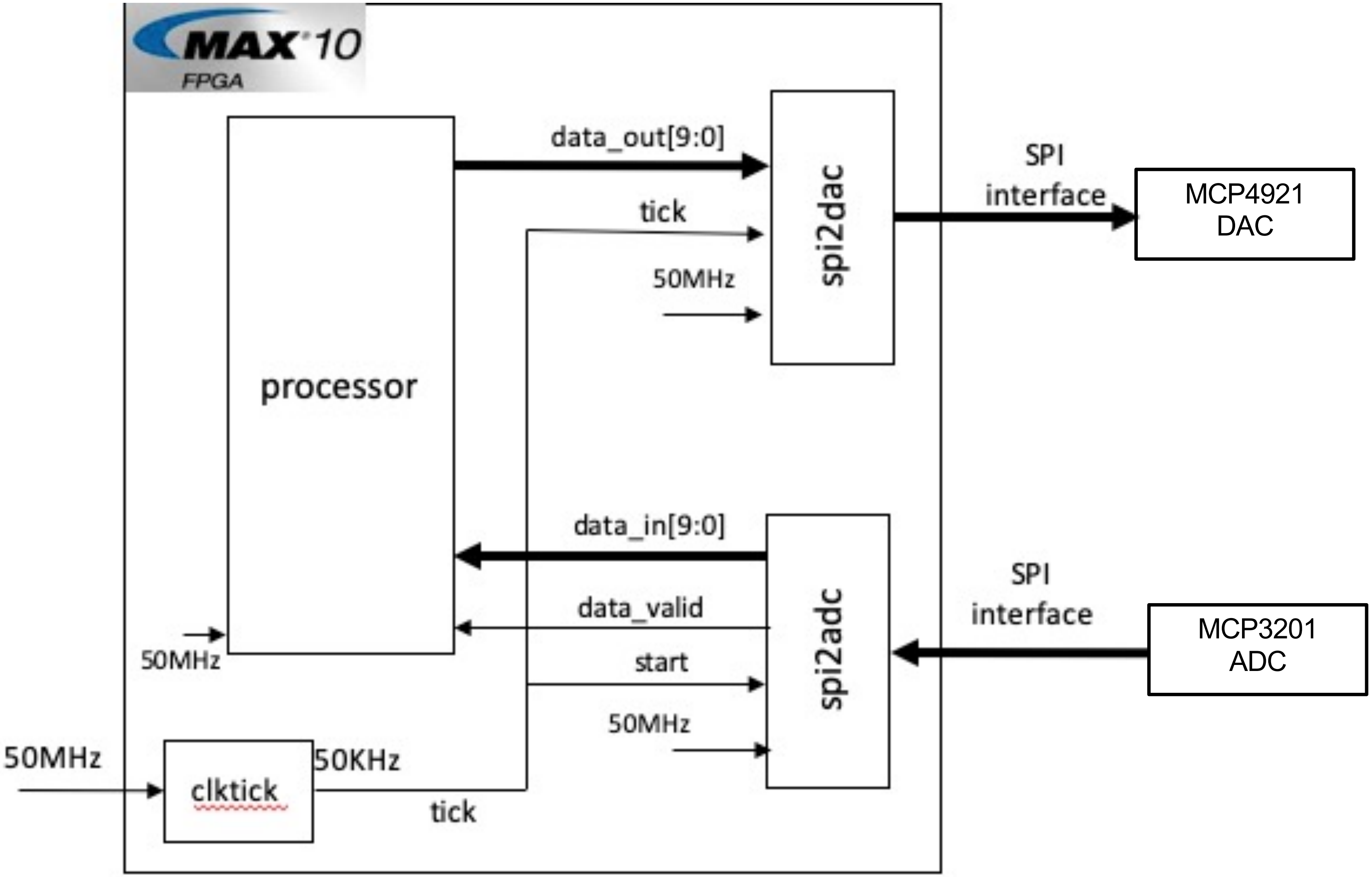
How to minimize problems?

1. Top level module name and file name (i.e. *.v) must match. This rule only applies to top-level module connected to physical pins.
2. Always check each .v file for syntax error with **Processing > Start > Analyze and Elaborate**
3. Make sure that you have included ONLY the files used in your design with **Project > Add/Remove files in Project**
4. Make sure that you have specify the correct top-level entity by first open the top-level module file, and click **Project > Set as Top-level Entity**
5. Always check for correctness of your design with **Processing > Start > Start Analysis and Synthesize**, and fix any errors
6. Check that you have assigned top-level ports to physical pins (done by editing the **<project_name>.qsf** file).
7. Check that you have specified your device to be 10M50DAF484C7G
8. Edit .qsf file to add pin assignment immediately after creating the project

Common mistakes

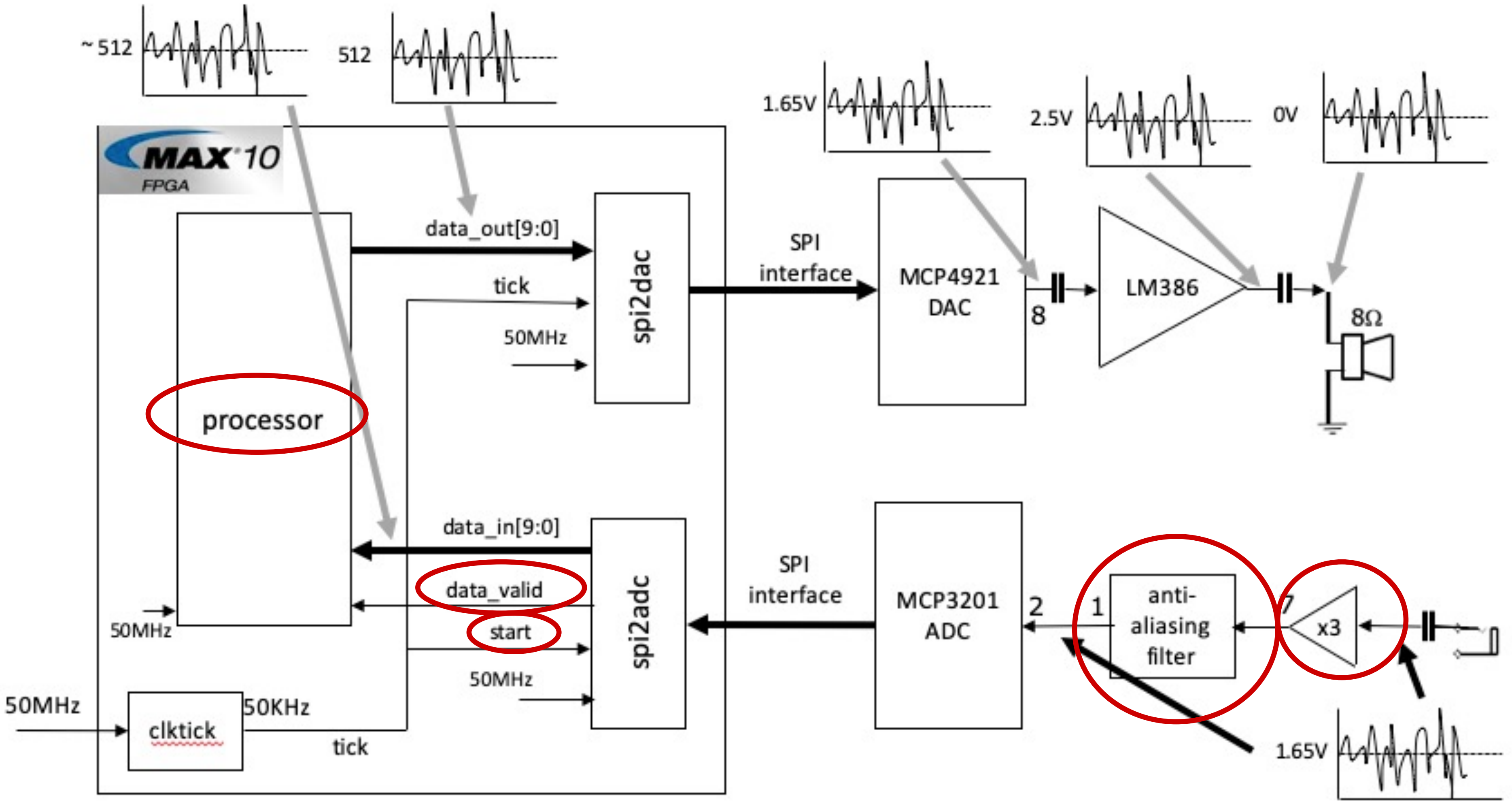
1. Bad organisation of design folder – missing versions, files, folder etc.
2. Wrong case for signal names (all names are case sensitive)
3. Wrong number or wrong order of signals when instantiating a module
4. Different number of bits used in signals at top-level and lower modules
5. Missing pin assignments or use the wrong pin names
6. You may use multiple always_ff @ (posedge/negedge clk) blocks in the SAME module, but must not do assignment to the same signal more than once

Lab 6 Task 3 Explained

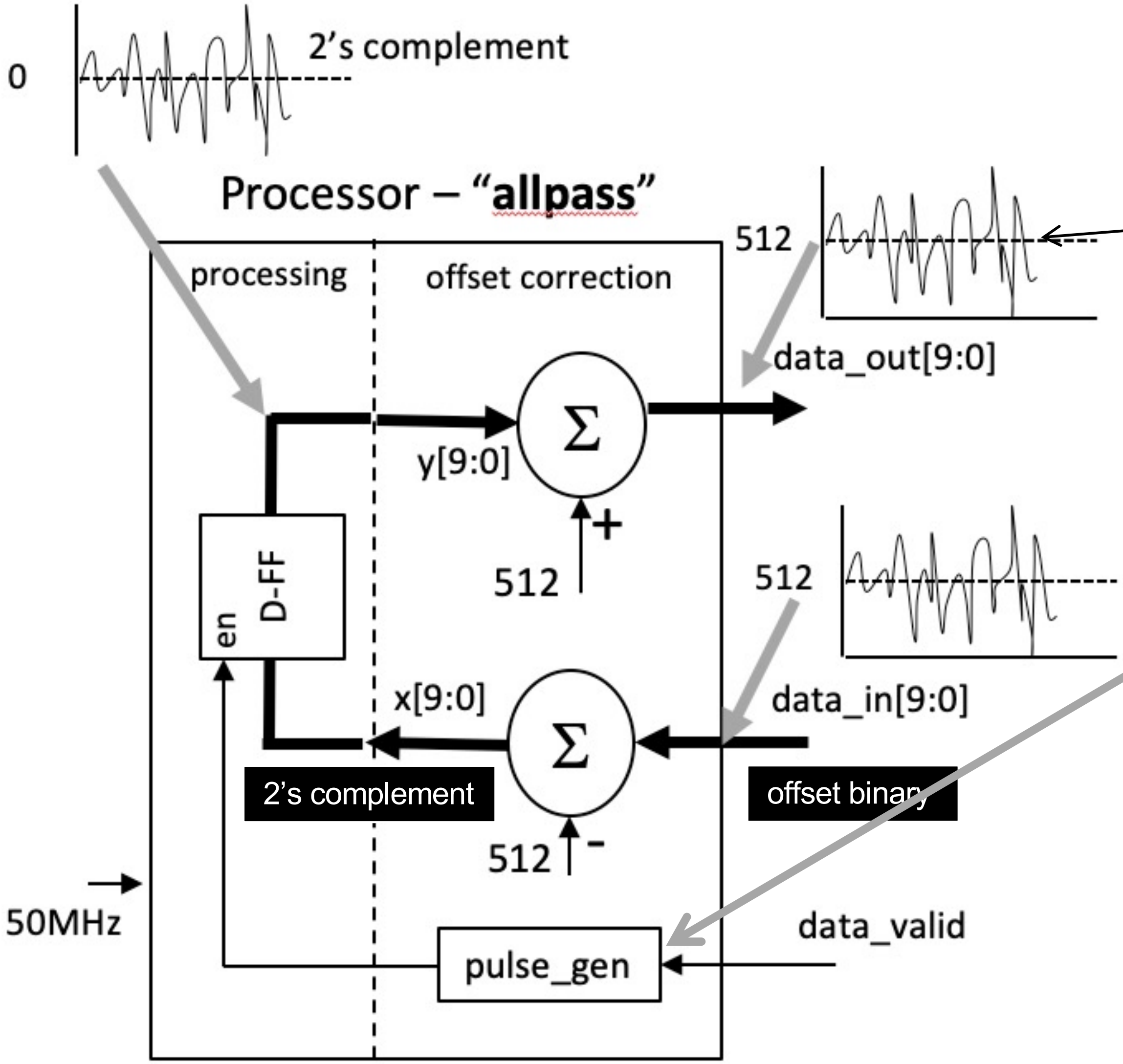


Combining analogue and digital systems

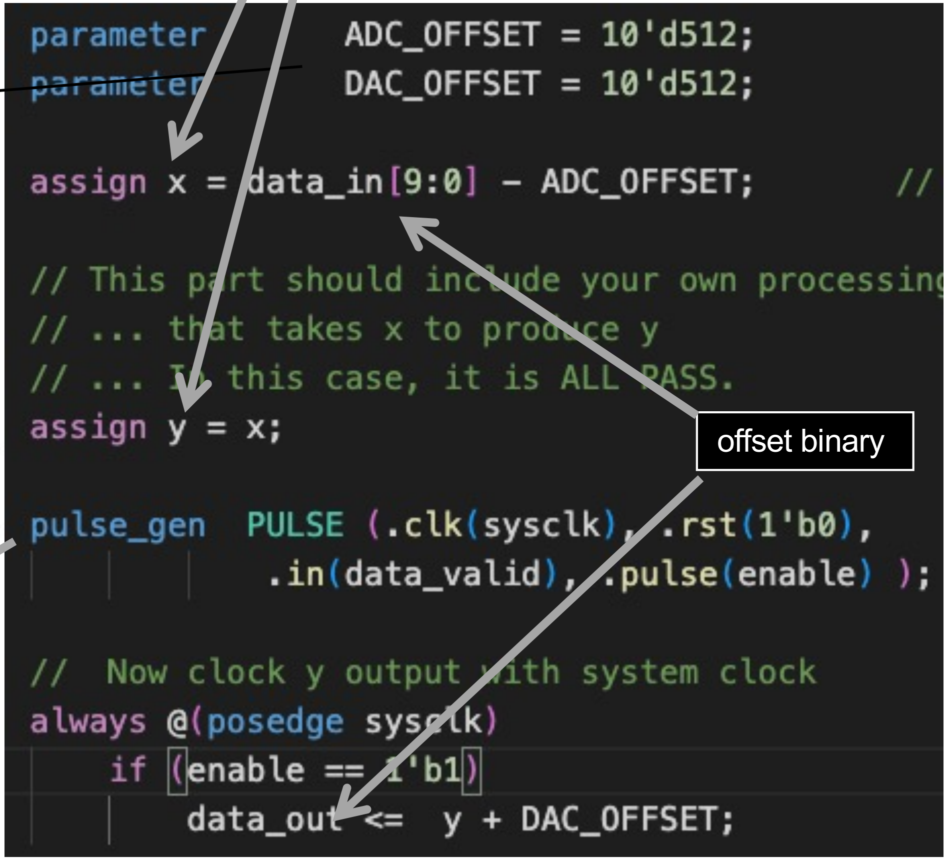
- X3 amplifier & anti-aliasing LP filter
- ADC produces a data_valid pulse at end of conversion



Lab 6 Task 3 – allpass.v (offset correction)



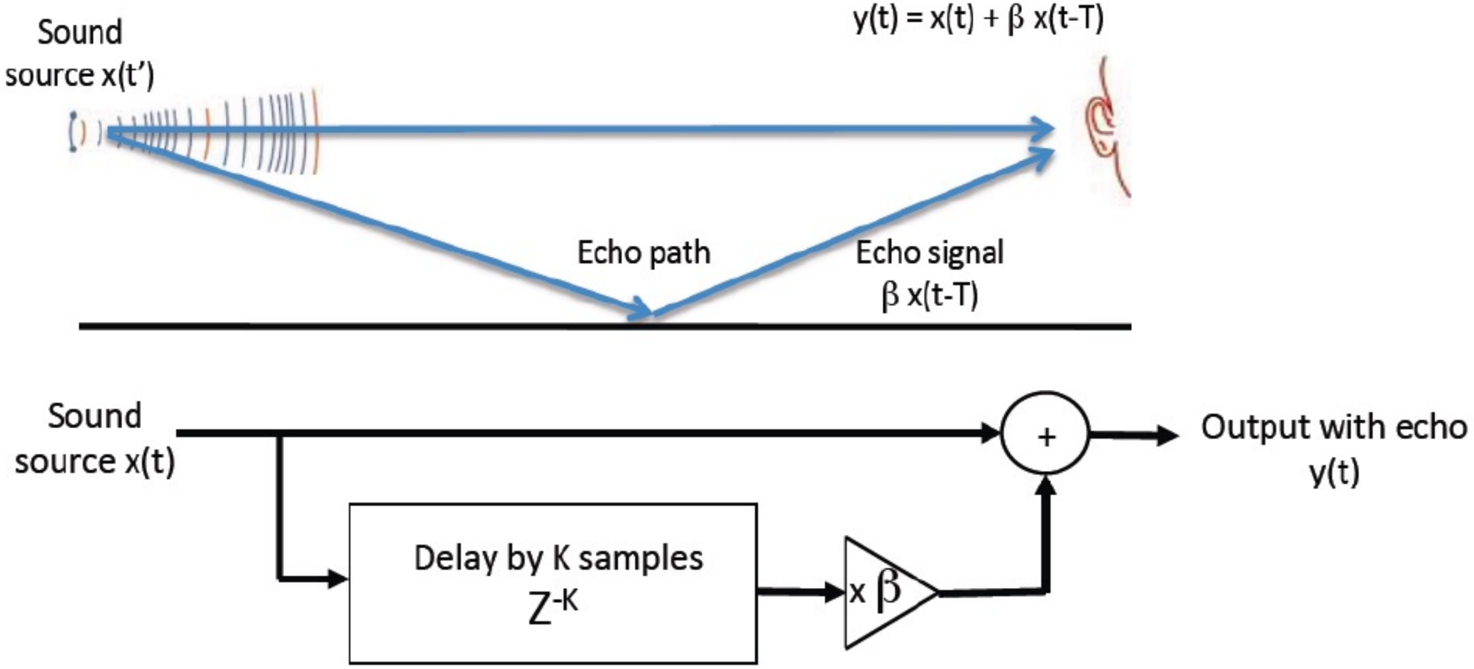
2's complement

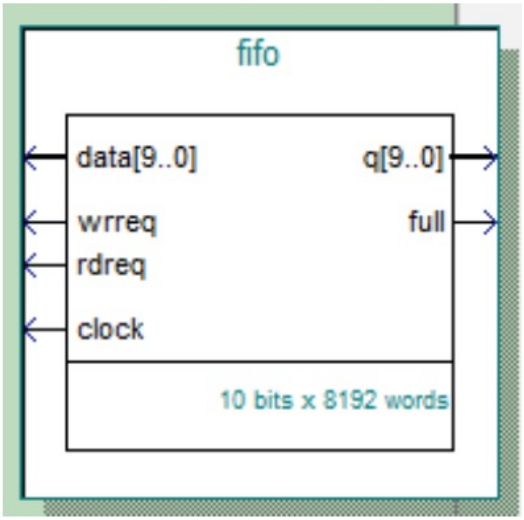


```
parameter       ADC_OFFSET = 10'd512;
parameter       DAC_OFFSET = 10'd512;

assign x = data_in[9:0] - ADC_OFFSET;        //

// This part should include your own processing
// ... that takes x to produce y
// ... In this case, it is ALL PASS.
assign y = x;

pulse_gen  PULSE (.clk(sysclk), .rst(1'b0),
                  .in(data_valid), .pulse(enable) );

//  Now clock y output with system clock
always @(posedge sysclk)
    if (enable == 1'b1)
        data_out <= y + DAC_OFFSET;
```

offset binary

Lab 6 Task 4 – single echo synthesizer

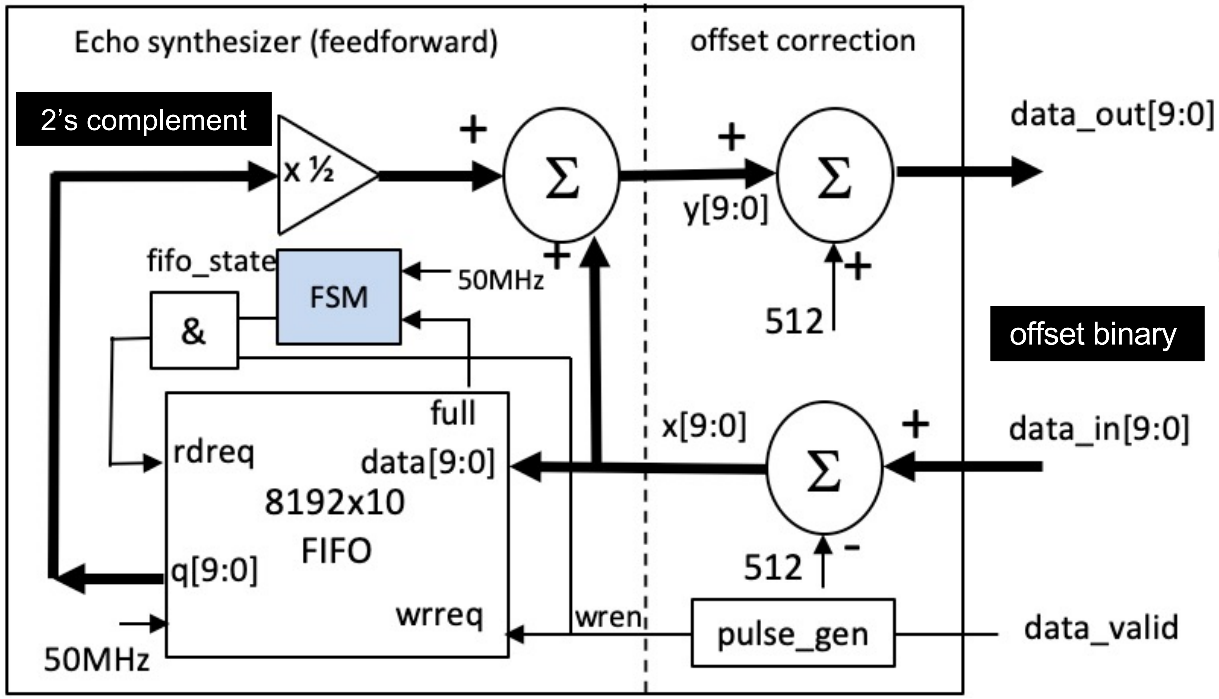


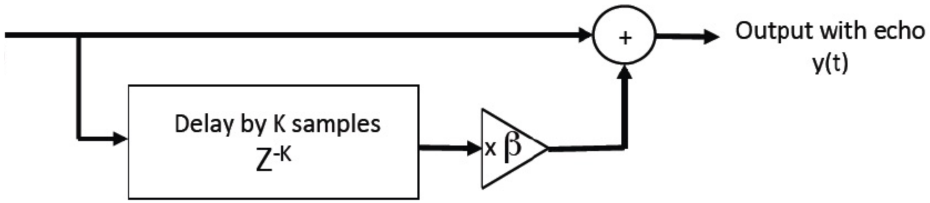


- Single echo of source signal
- Signal flow-graph is simple: a K samples delay block, a gain block and an adder
- Use First-in-First-out memory to store sample: need a status signal “full” to indicate FIFO full
- Sampling frequency = 50KHz, theref a 8192 word FIFO provides 0.1638 second delay

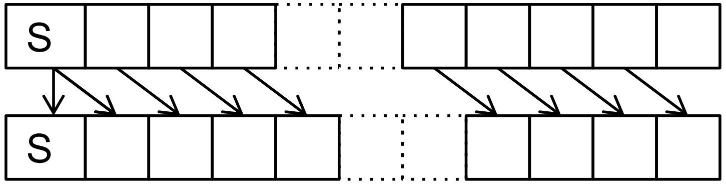
Lab 6 Task 4a – single echo synthesizer

Processor – simple echo



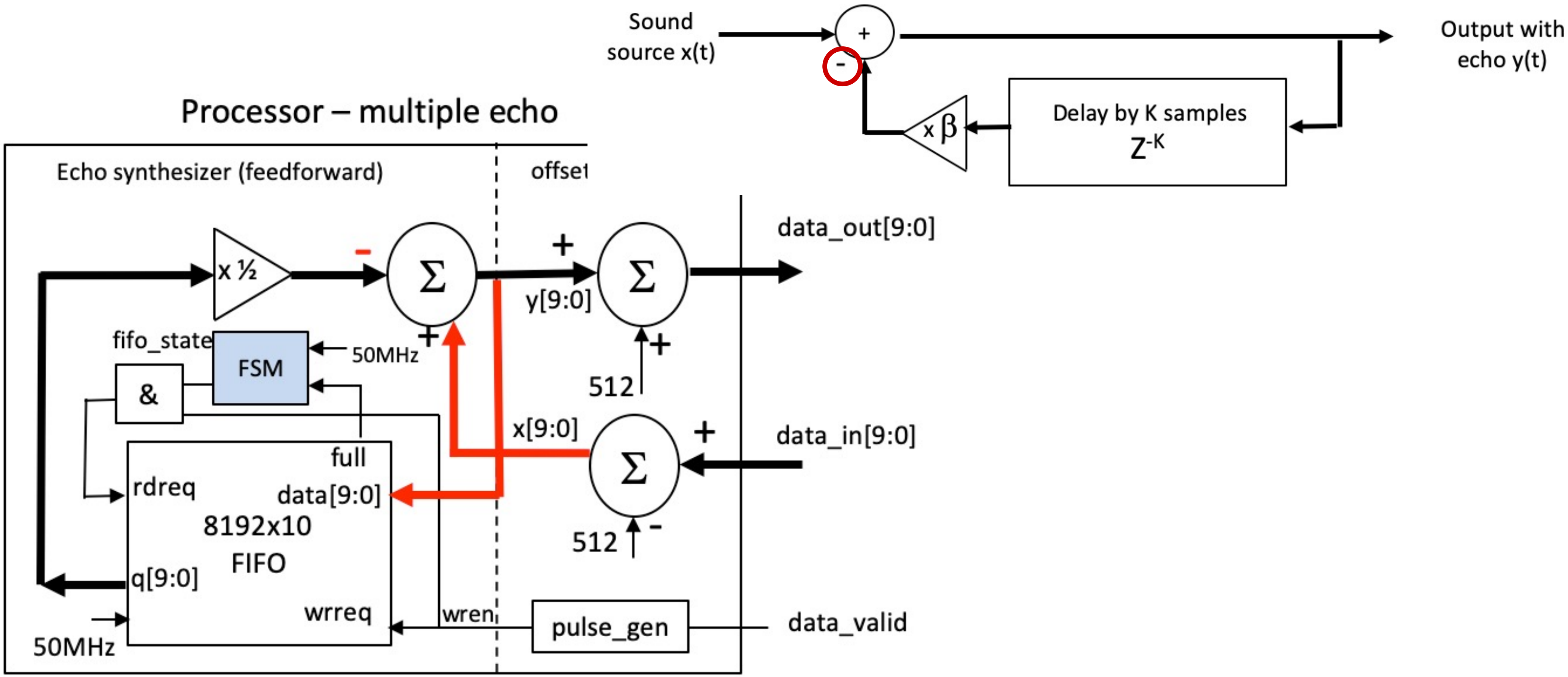


Right shift with SIGN EXTENSION



- Computation in 2's complement for signed integers
- x 0.5 = signed right-shift by 1-bit (sign-extension)
- Verilog: y[9:0] = x[9:0] + {q[9], q[9:1]};
- Additional signal to processor module: data_valid = a high pulse whenever there is a new data_in
- Need to fill to First-in-First-out memory before starting to read data off it – hence FSM

Lab 6 Task 4b – multiple echoes synthesizer

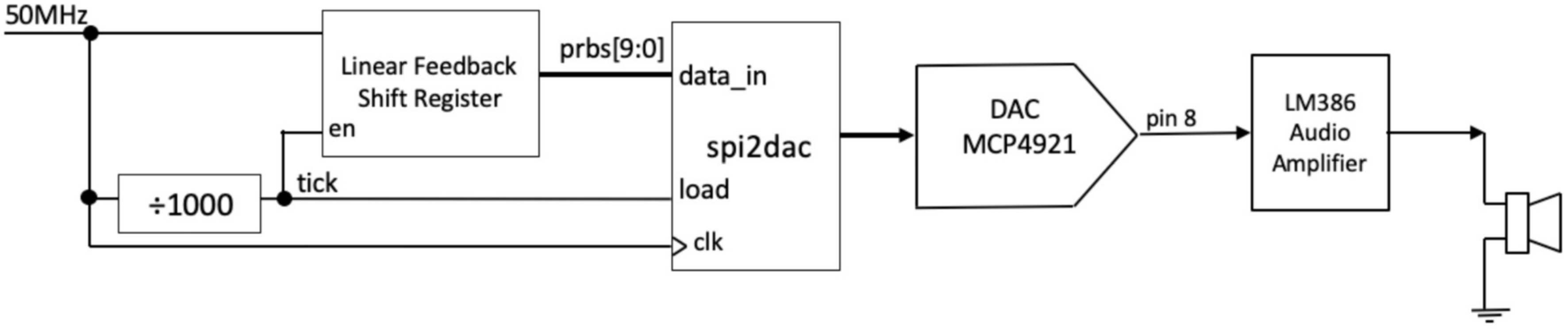


- Instead of feedforward only, this uses a feedback loop
- To avoid instability, you must SUBTRACT delayed echo signal instead of add
- FIFO now stores y[9:0] output, and NOT input

Challenges

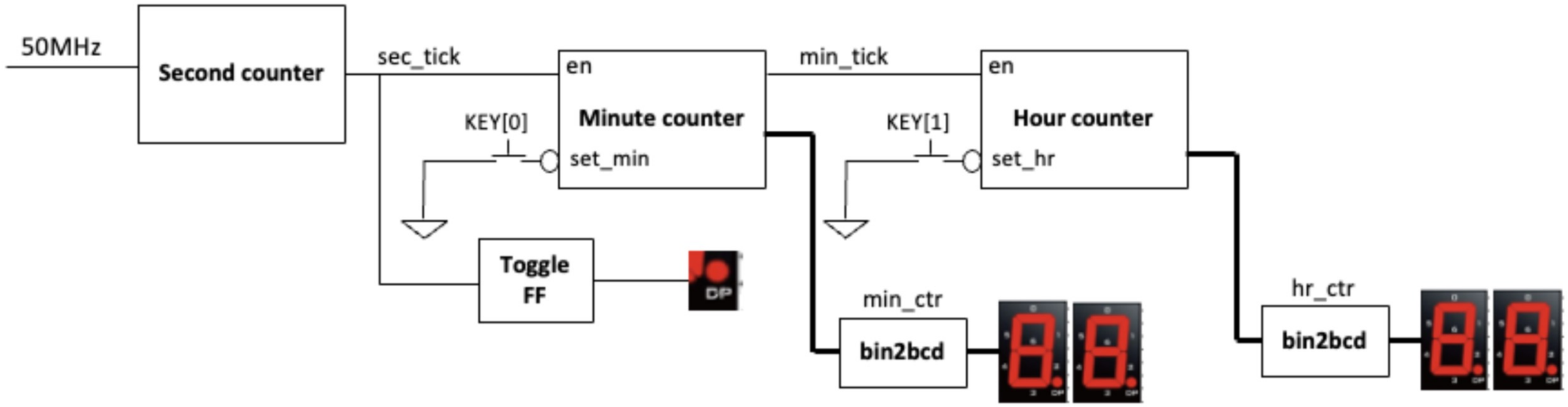
- Lab 1 – 6: Teaching you by holding your hands, with a few “test yourself” tasks
- Challenges: Open-ended problems to challenge you. Give you a chance to “showoff” what you have learned
- No time to do more than one or two. Welcome to do them all if you want.
- Final Lab Oral – asked evidence of successful challenges (videos)
- Not completing any challenges will limit your final lab oral grade to at best a B (fair to others)
- All challenges are ranked in levels of difficulties (1 to 4)

Challenge 1 – Noise Generator (level 1)



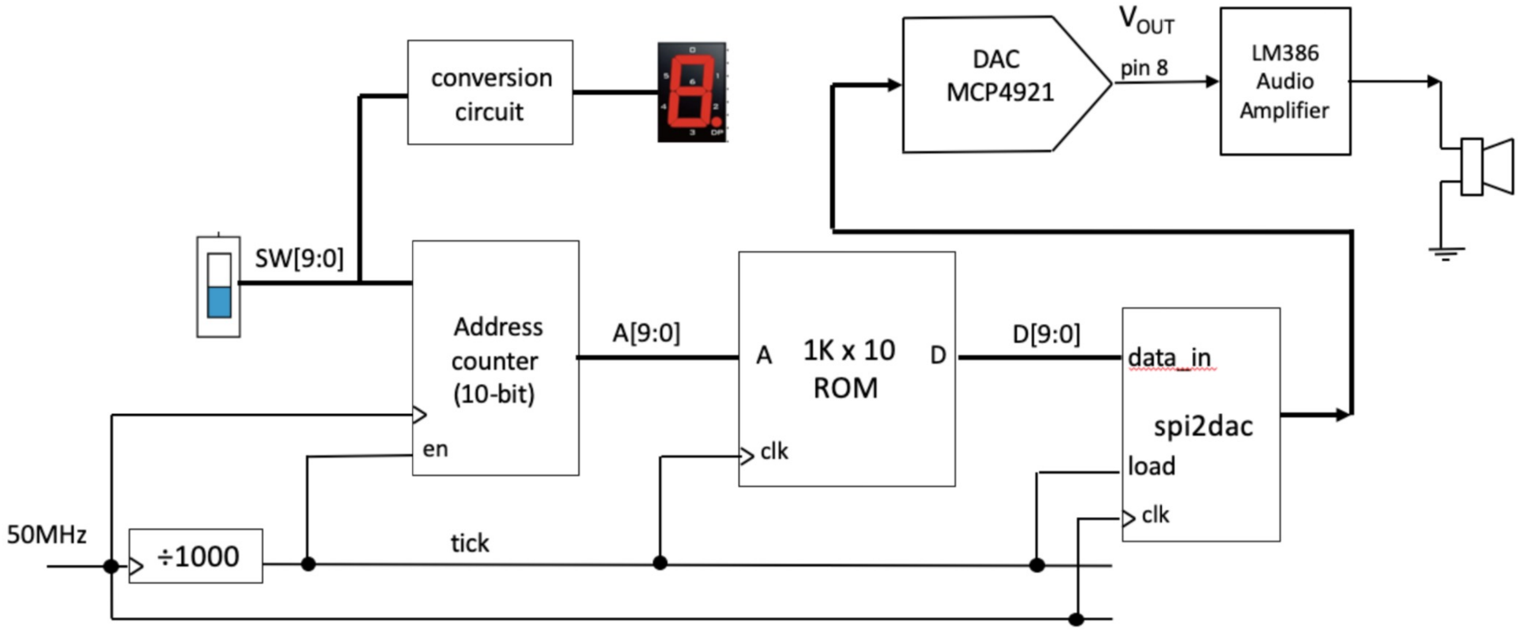
- Hint: If you use the 9-bit PRBS from Lab 5, remember that the DAC is 10-bits. Beware of the mismatch.
- Better to use a 10-bit PRBS implementing a primitive polynomial for 10-bits from the table provided in the lecture notes.

Challenge 2 – A real-time clock (level 2)



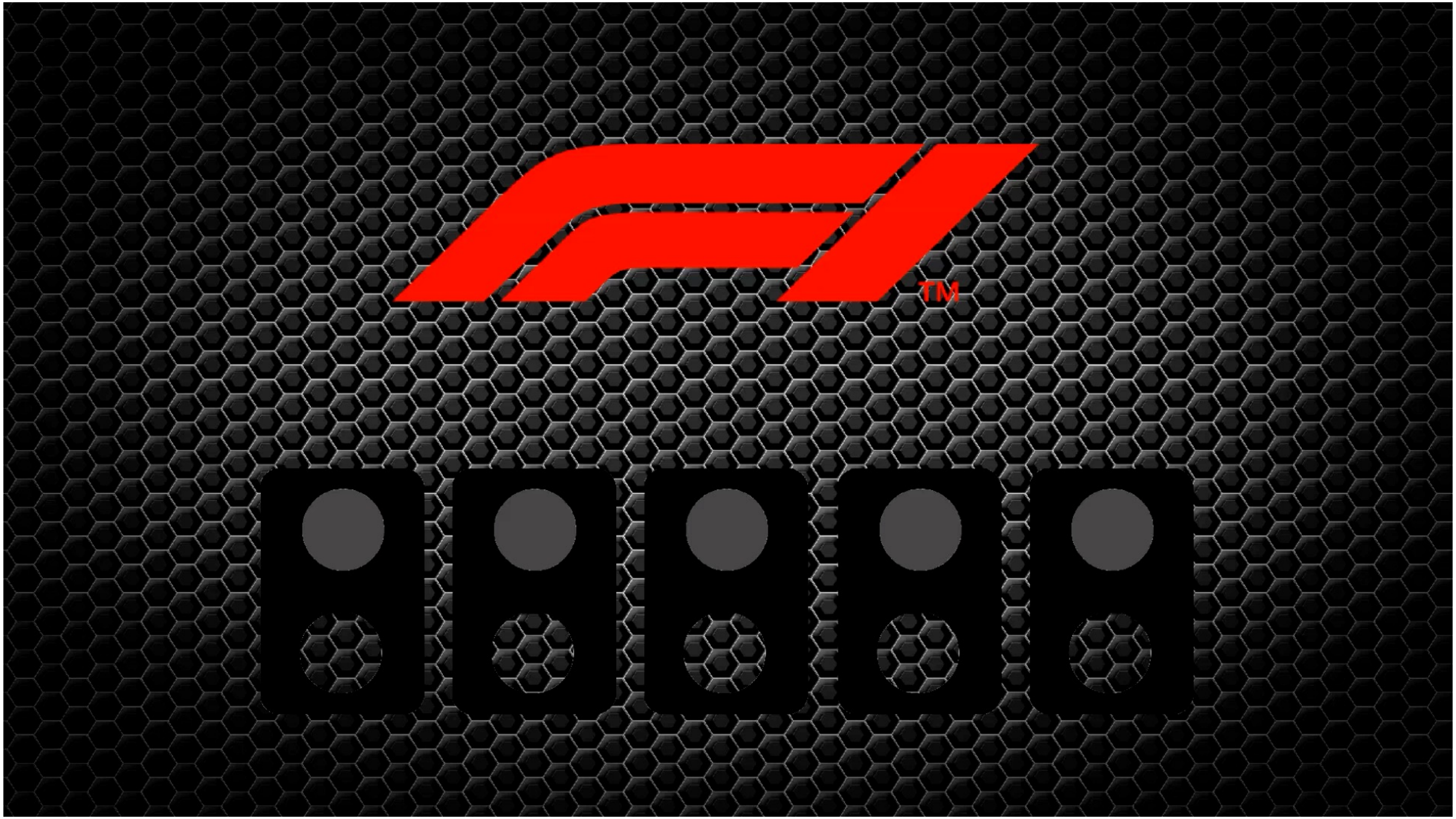
- This is a simple challenge because you only use counters and a few extra modules.
- The tricky part is to have a way of setting minutes and hours.

Challenge 3 – Variable sinewave generator (level 2 or 3)

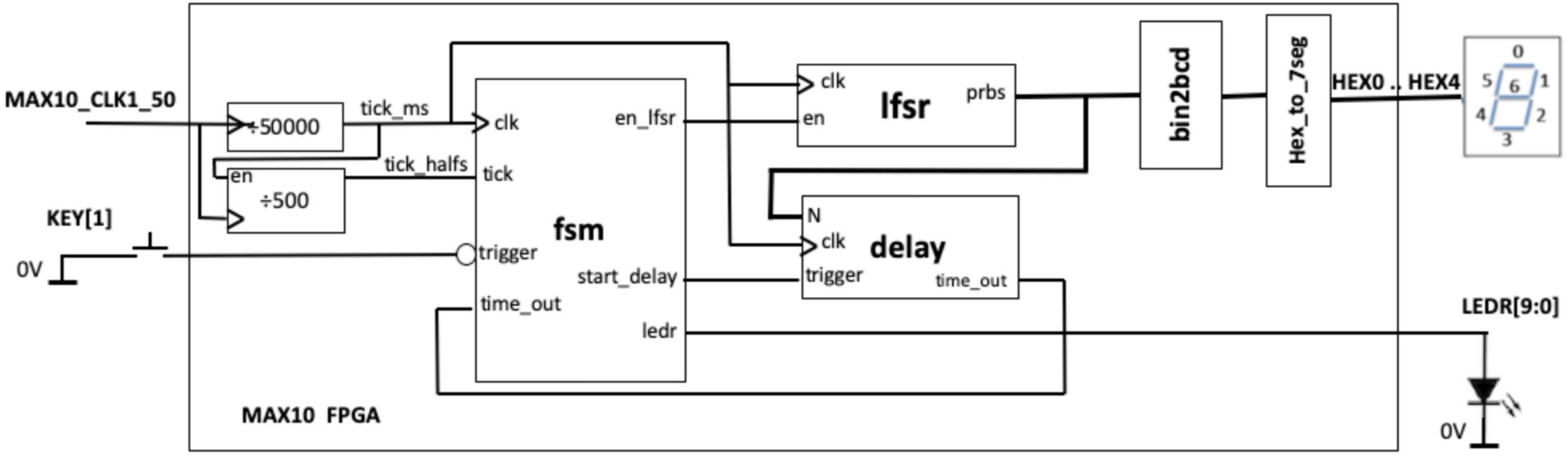


- This challenge teaches you to design with a ROM as well. The level 3 attainment will involve the translation of SW[9:0] to the actual frequency of the sinewave produced.

Challenge 4 – Formula 1 starting light (level 4)

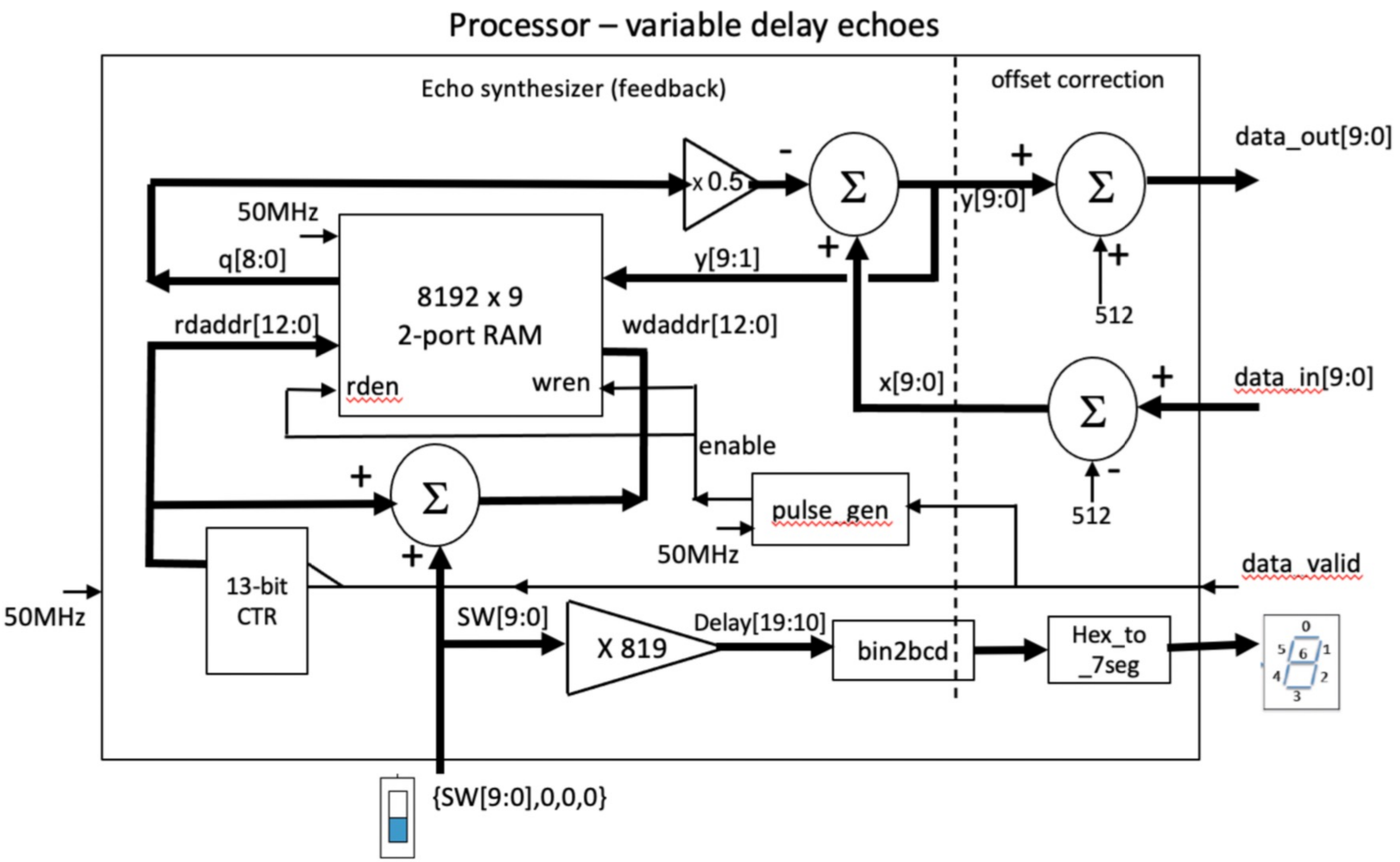


Challenge 4 – Formula 1 starting light (level 4)



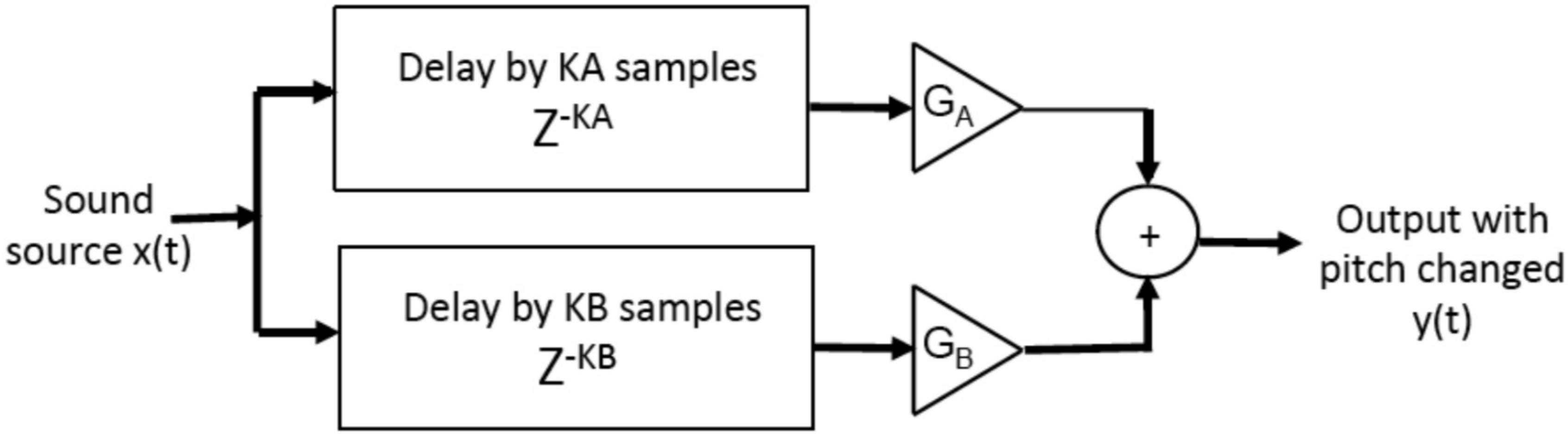
- This challenge is level 4 because it really tests every aspects of the digital part of the module. It requires LFSR, FSM, counters, delay module, shift registers etc.
- Stretch goal allows you test your reaction time (in ms)

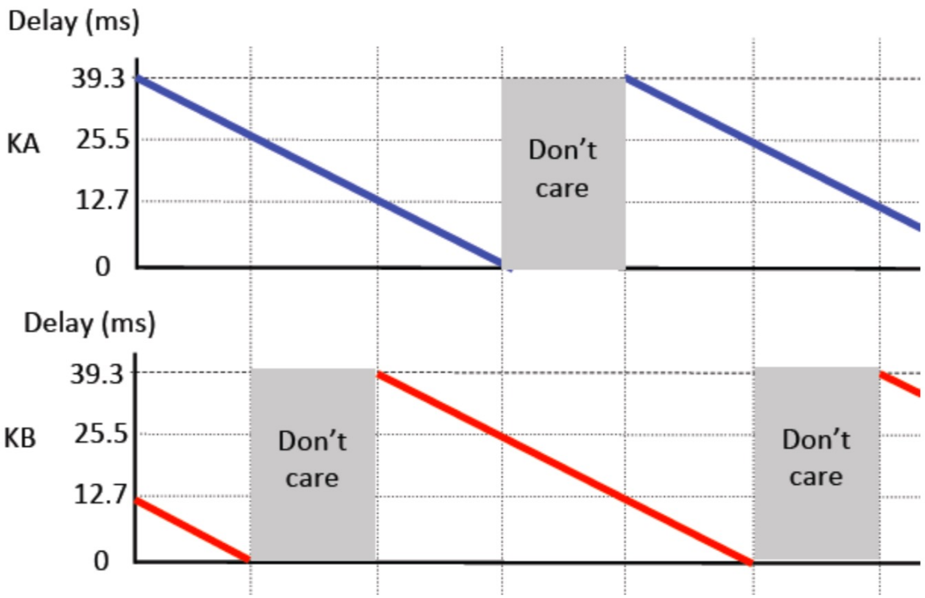
Challenge 5 – Variable Delay Echo Synthesizer (level 4)

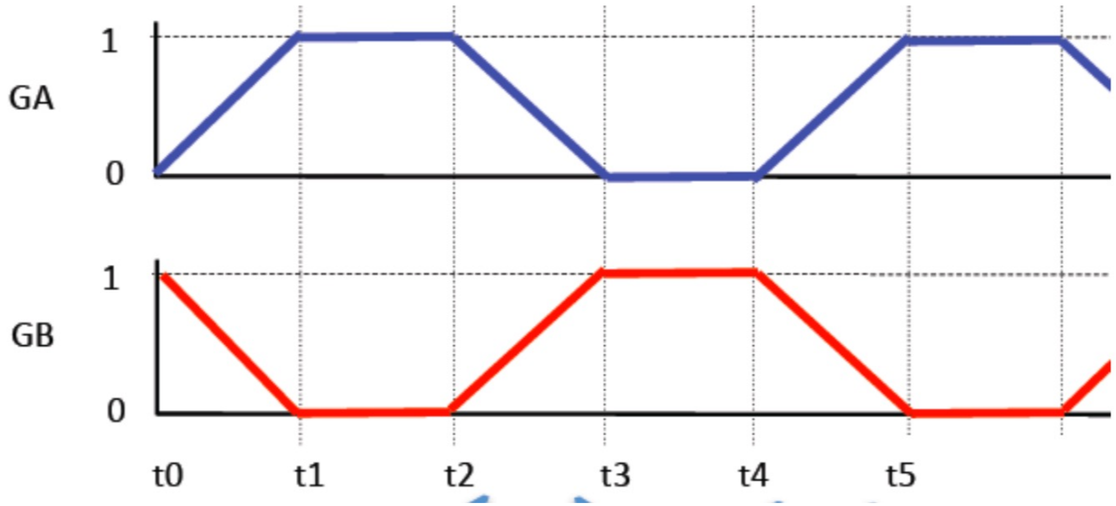


- This challenge is based on an extension to Lab 6. Instead of FIFO, you use a RAM to implement a variable delay block. Note also that you need to compute the delay in ms and display this!

Beyond the challenges – A Voice Changer


Sound source x(t)
Delay by KA samples
Z^{-KA}
Delay by KB samples
Z^{-KB}
G_A
G_B
+
Output with pitch changed y(t)


Delay (ms)
KA
Don't care
Delay (ms)
KB
Don't care
Don't care


GA
GB
t0
t1
t2
t3
t4
t5