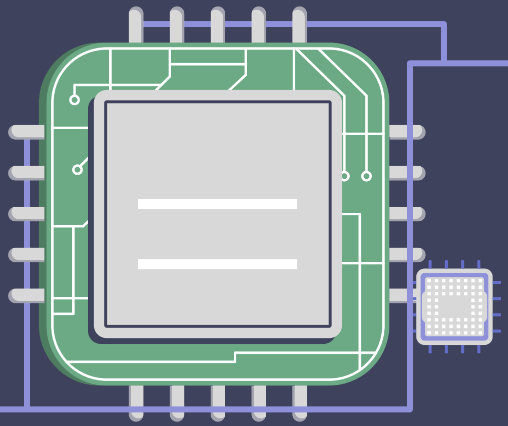
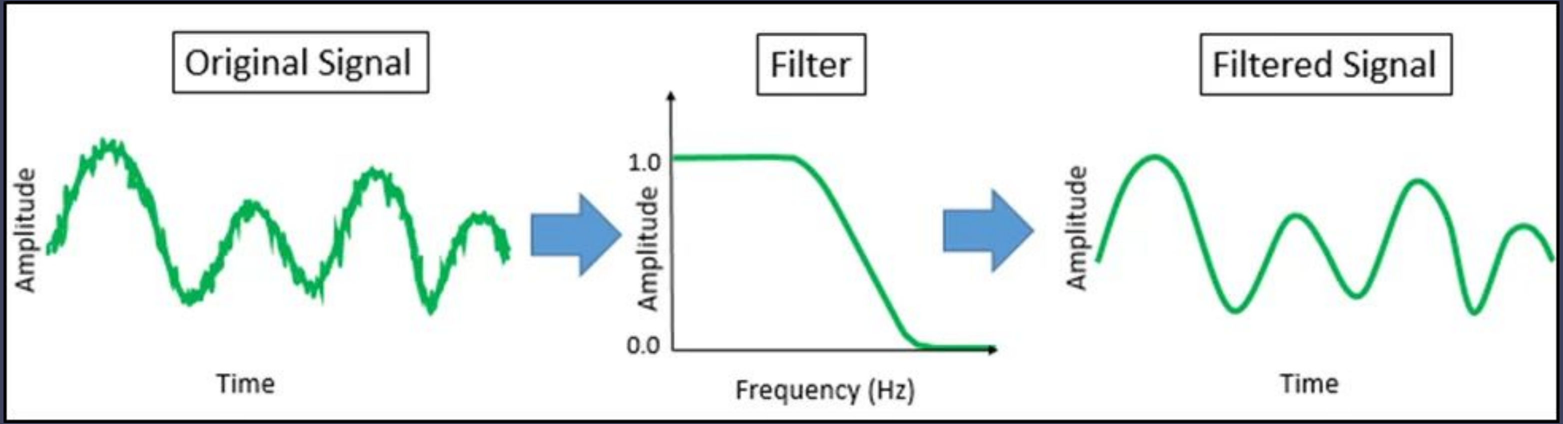
DIGITAL FILTERING USING FPGA

Group 6: Alex, Heewon, Vanshika, Yash



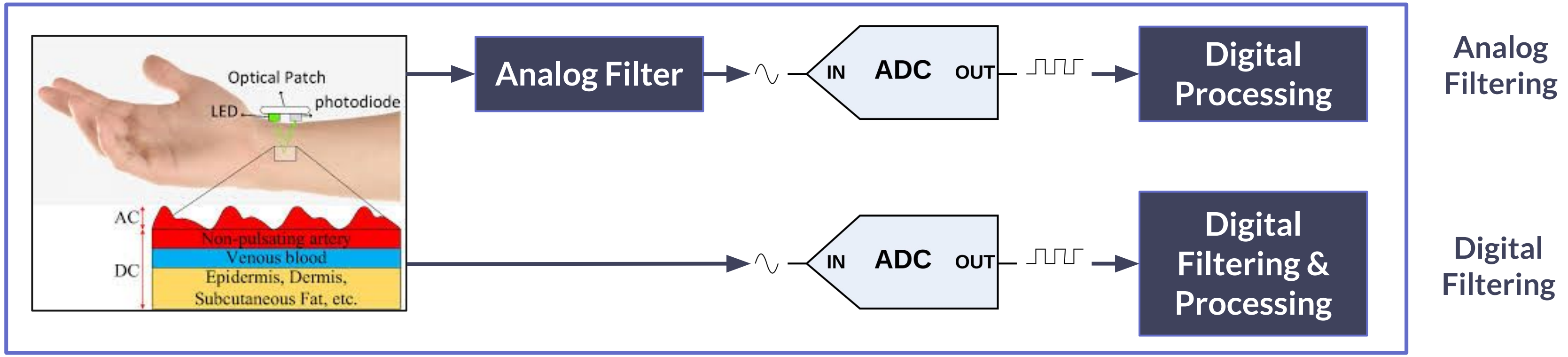
WHAT IS FILTERING



Removing unwanted frequencies (noise) from your signal by either reducing or amplifying certain frequencies.

ANALOG VS. DIGITAL

Analog filters are made of **physical components** such as resistors, capacitors, inductors, and operational amplifiers.

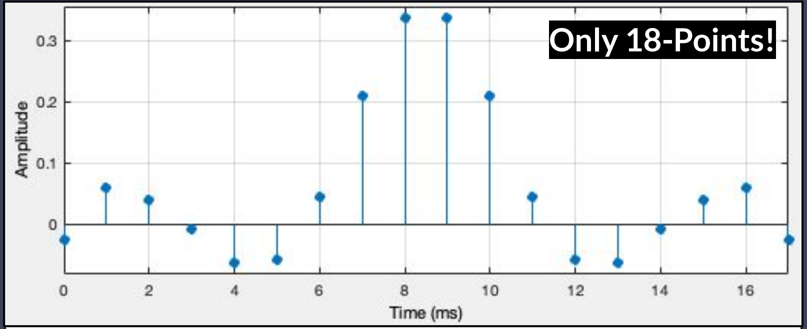


Advantages of Digital Processing

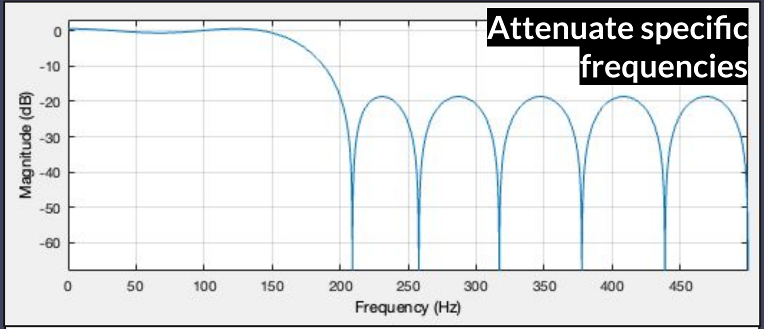
- **Flexibility:** Hardware can easily **be reprogrammed** allowing for easy testing and optimization of different filters (no modification of hardware needed)
- **Cost-Effectiveness**: The same hardware (ADC and processing unit) can be mass-produced and reused for multiple filtering applications, reducing manufacturing costs.

DIGITAL FILTERS

Digital filters use a digital processor to perform **numerical calculations** on values of the signal.



Impulse Response



Frequency Response

For the same filter specifications, much simpler filter implementation!

- Based on **convolving** your sampled input signal with the filter impulse response
- Can be implemented as a cascade of **Second order sections (SOS)**

Designed using MATLAB: [Digital] Filter Designer

APPLICATIONS

Digital filters have a variety of **biomedical applications** as they enhance the quality of physiological signals for accurate diagnosis and monitoring.

Wearable Health Devices

- Enhancing signal quality in devices like fitness trackers, glucose monitors, and heart rate monitors by reducing environmental noise.

Electrocardiogram (ECG) & HR Signal Processing

- Filtering noise from ECG signals to improve the accuracy of diagnosing heart conditions

Source: Stelo by Dexcom

Source: Apple

DESIGN SPECIFICATION AND CONSTRAINTS

Fixed Point Representation and Arithmetic for Integers

8-bit signed integer numbers

1-Bit Sign	7-Bit Exponent

Digital Filtering (specifically S.O.S.)

Module should function either as a **second order section and convolutions** to implement designed digital filter.

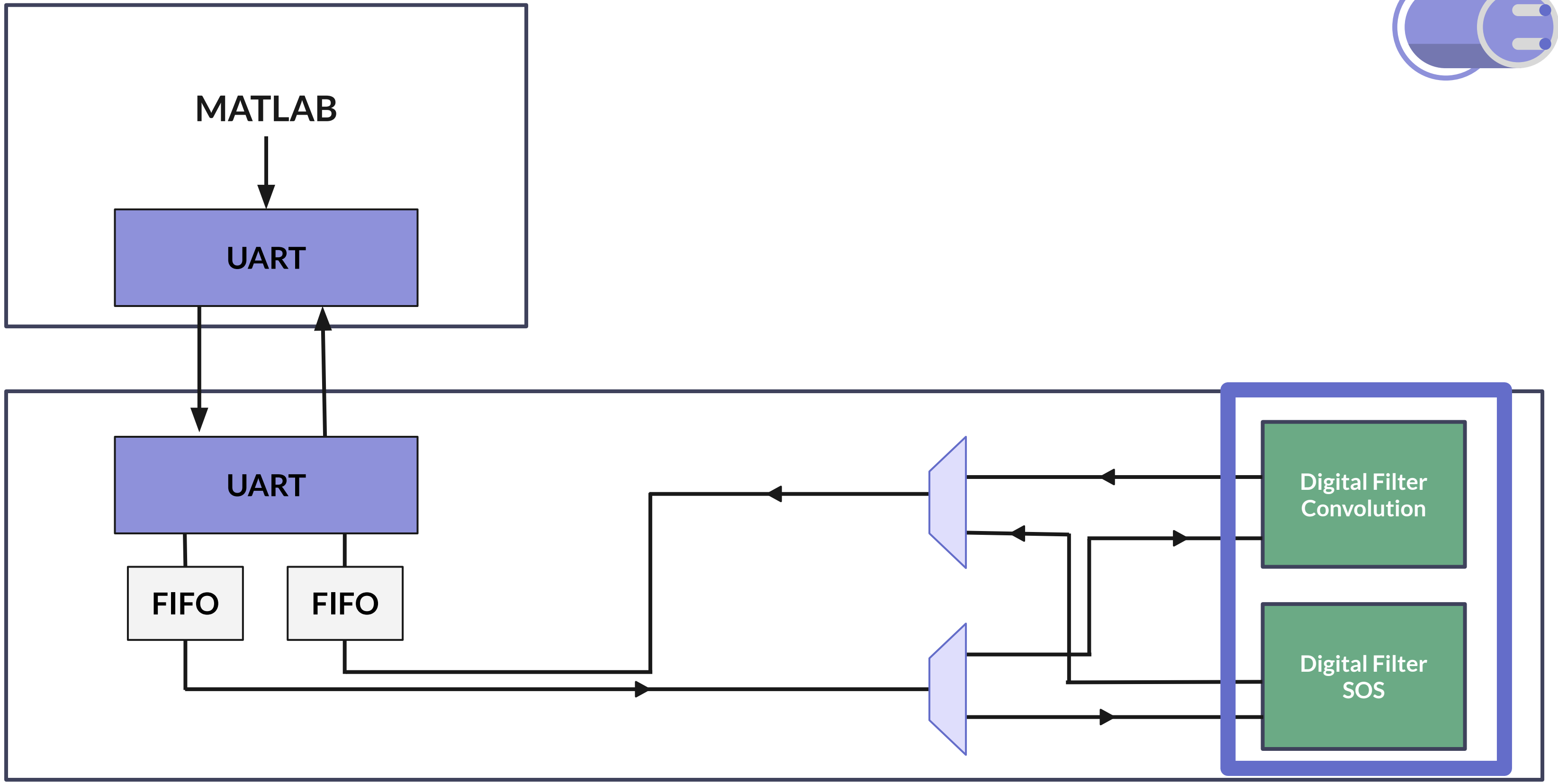
Full-Duplex UART Communication with MATLAB

Receive and transmit 16-bit signed, floating numbers (no parity)

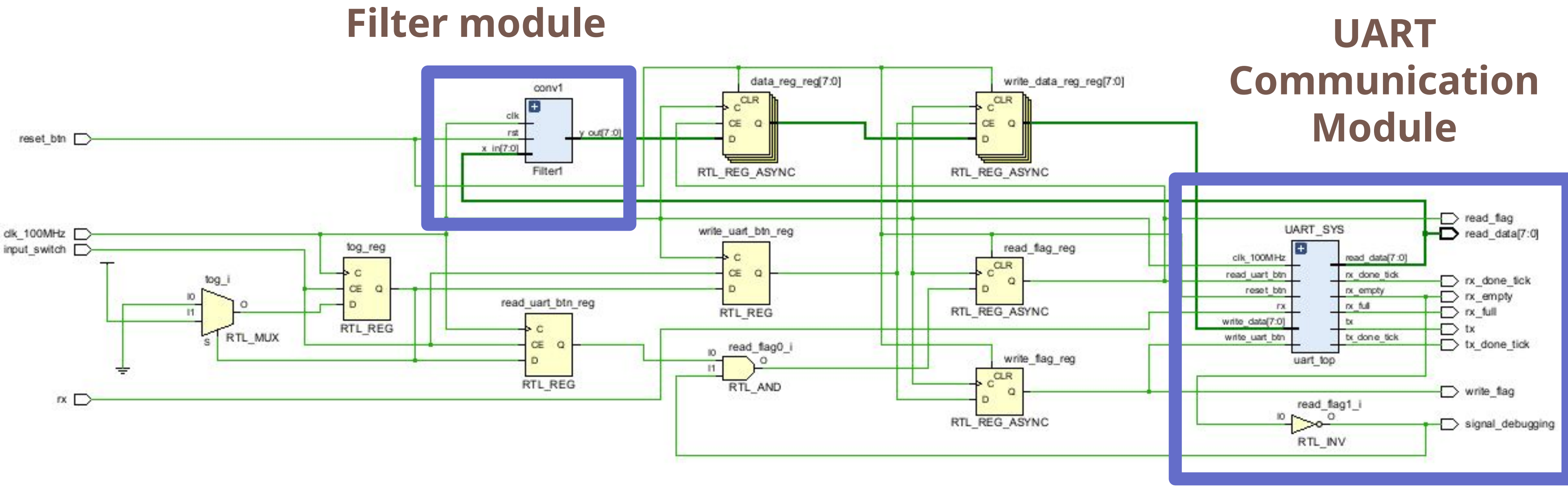
FIFO Buffers: provide temporary storage to data

Constraints

- Floating point representation — only integers implemented in filter at this moment
- FPGA board compatibility with UART

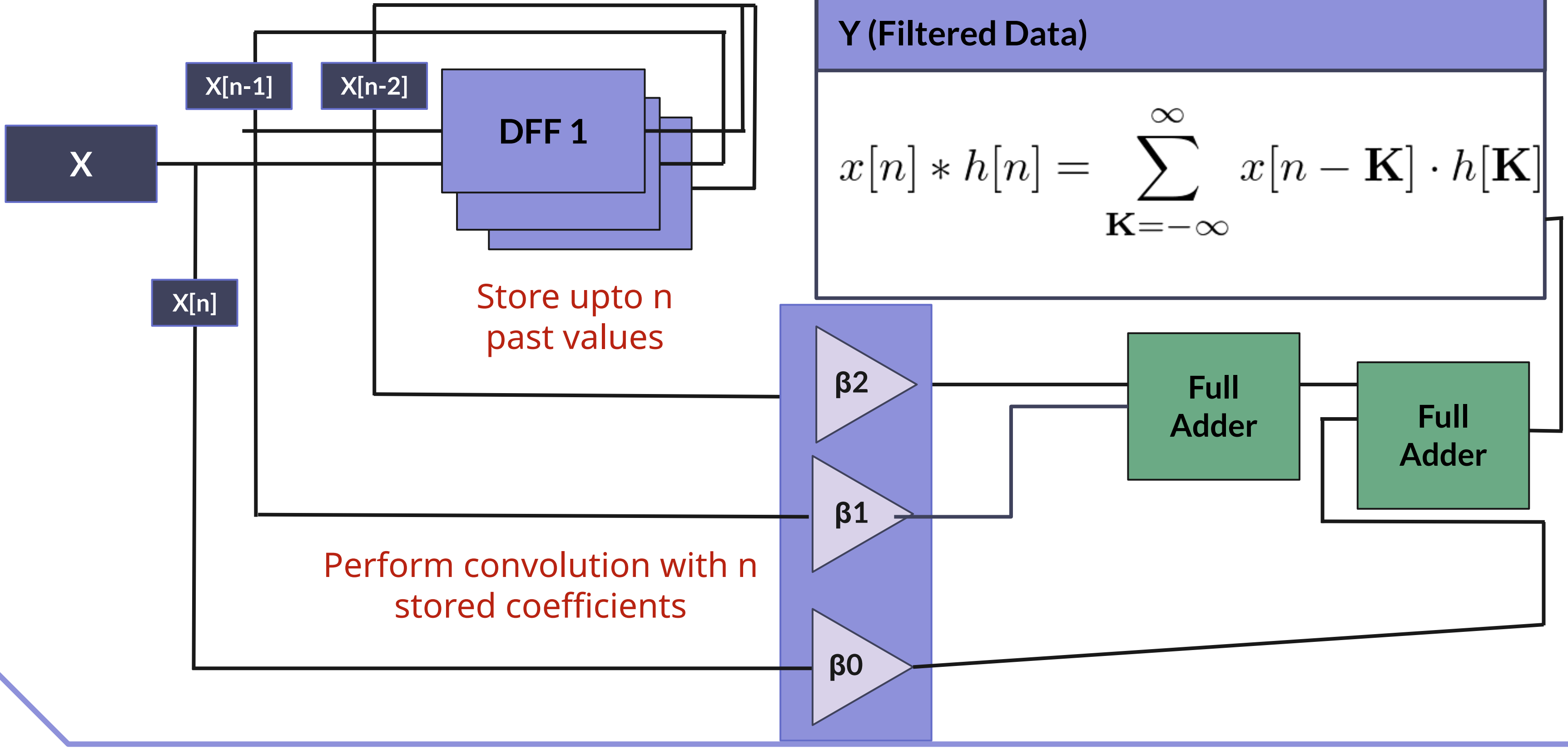


BLOCK DIAGRAM



VERILOG GENERATED BLOCK DIAGRAM

FILTERING USING CONVOLUTION



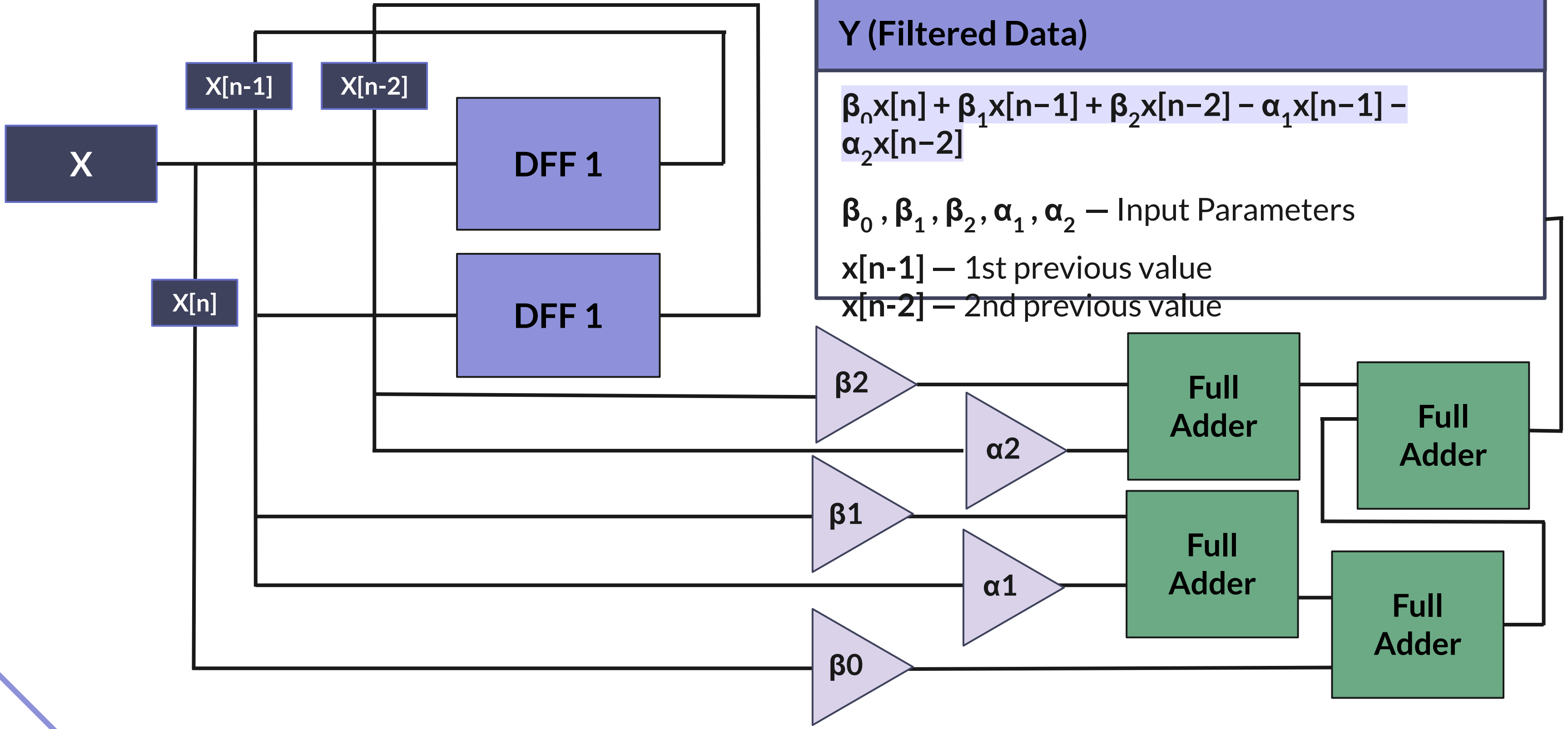
Y (Filtered Data)

$$x[n] * h[n] = \sum_{\mathbf{K}=-\infty}^{\infty} x[n-\mathbf{K}] \cdot h[\mathbf{K}]$$

Store upto n past values

Perform convolution with n stored coefficients

FIR FILTERING USING S.O.S



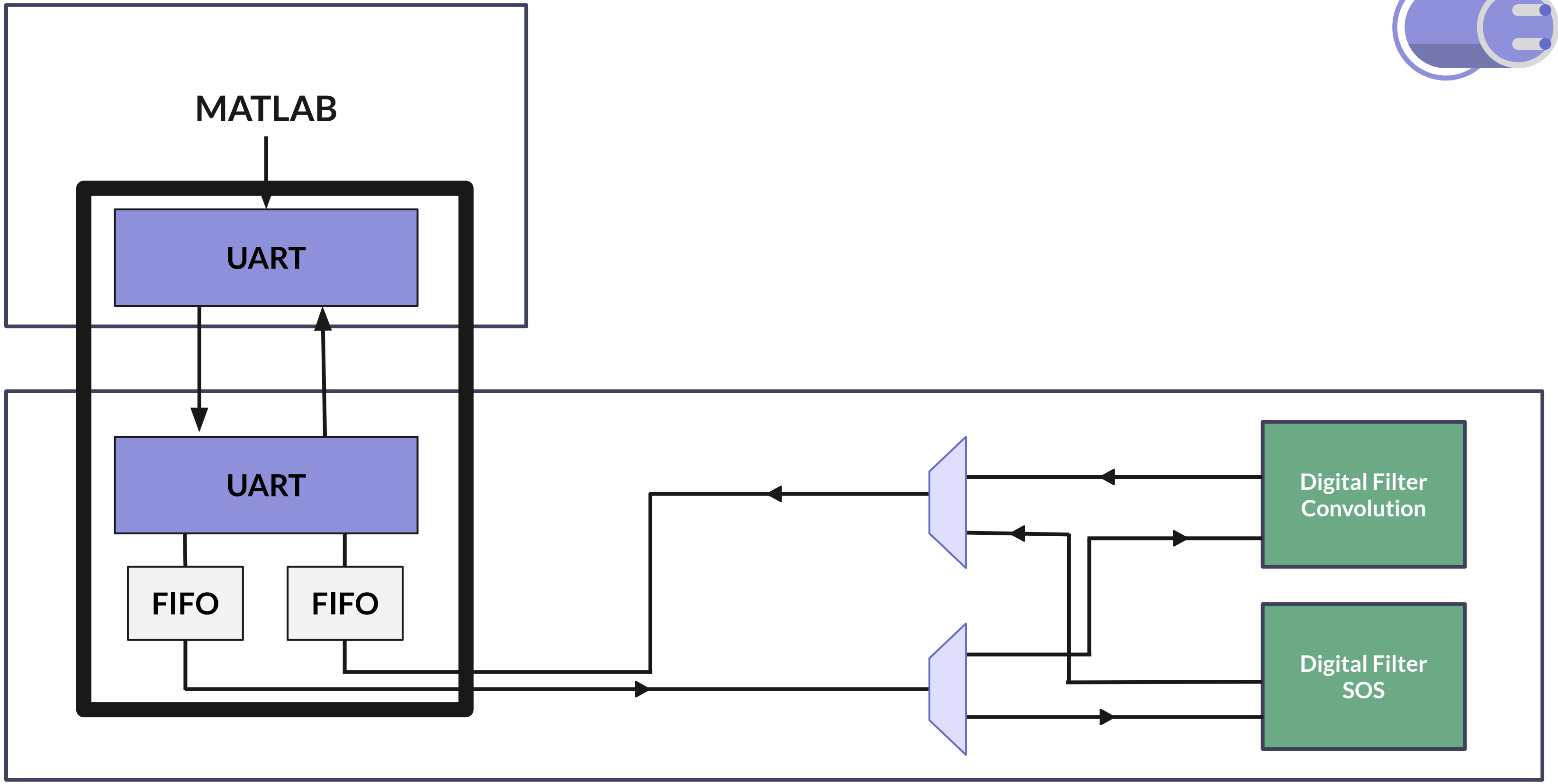
Y (Filtered Data)

$\beta_0 x[n] + \beta_1 x[n-1] + \beta_2 x[n-2] - \alpha_1 x[n-1] - \alpha_2 x[n-2]$

$\beta_0, \beta_1, \beta_2, \alpha_1, \alpha_2$ — Input Parameters

x[n-1] — 1st previous value

x[n-2] — 2nd previous value



BLOCK DIAGRAM

UART

UART stands for **Universal Asynchronous Receiver and Transmitter** and is a serial communication interface that allows data to be transmitted and received one bit at a time over a single data line

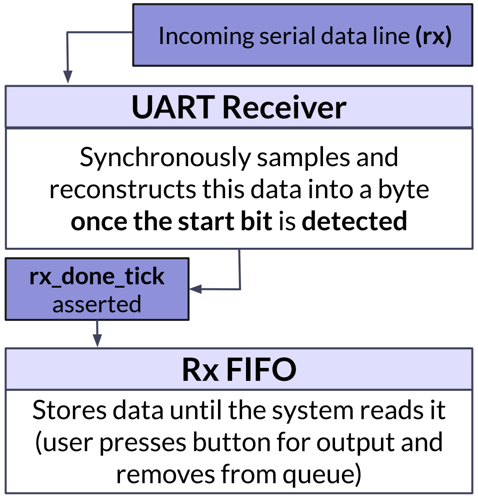
Key Components

- Baud Rate Generator
- UART Transmitter
- UART Receiver
- FIFO Buffers

FIFO

- Stores **8-bit numbers** and queue depth is **2^8**
- If the **FIFO is full**, **no new data** can be written

Data Reception



Incoming serial data line **(rx)**

UART Receiver

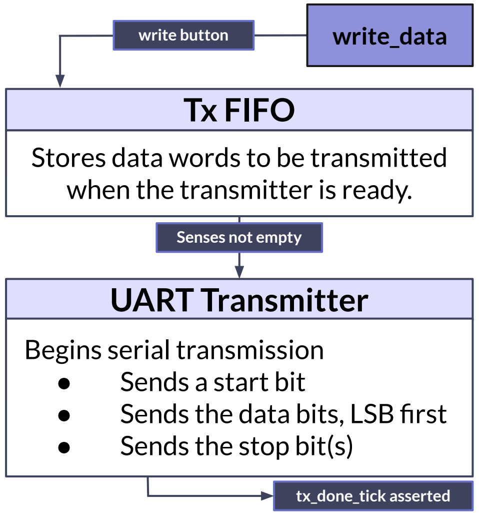
Synchronously samples and reconstructs this data into a byte **once the start bit** is **detected**

rx_done_tick asserted

Rx FIFO

Stores data until the system reads it (user presses button for output and removes from queue)

Data Transmission



write_data

write button

Tx FIFO

Stores data words to be transmitted when the transmitter is ready.

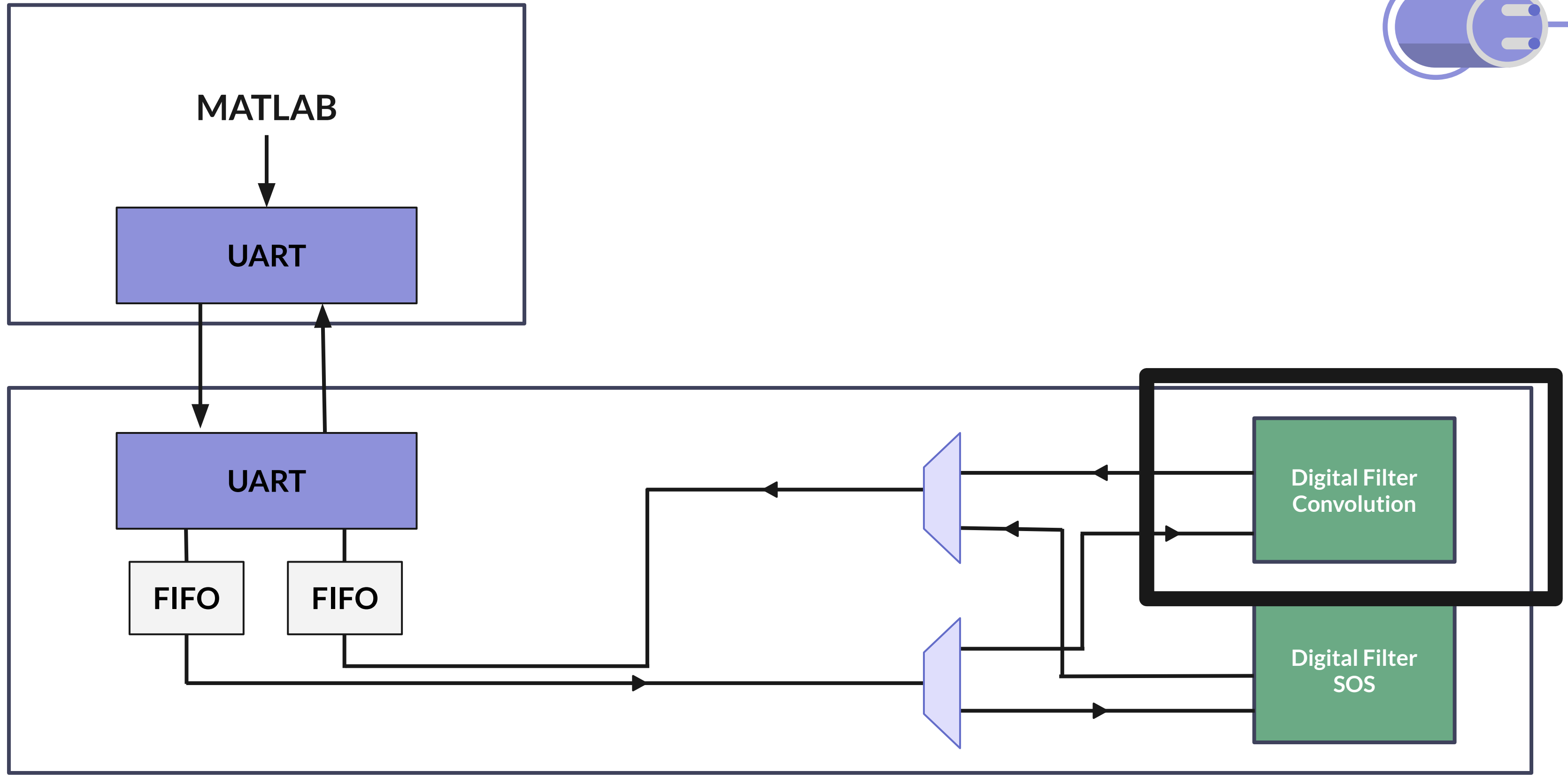
Senses not empty

UART Transmitter

Begins serial transmission

- Sends a start bit
- Sends the data bits, LSB first
- Sends the stop bit(s)

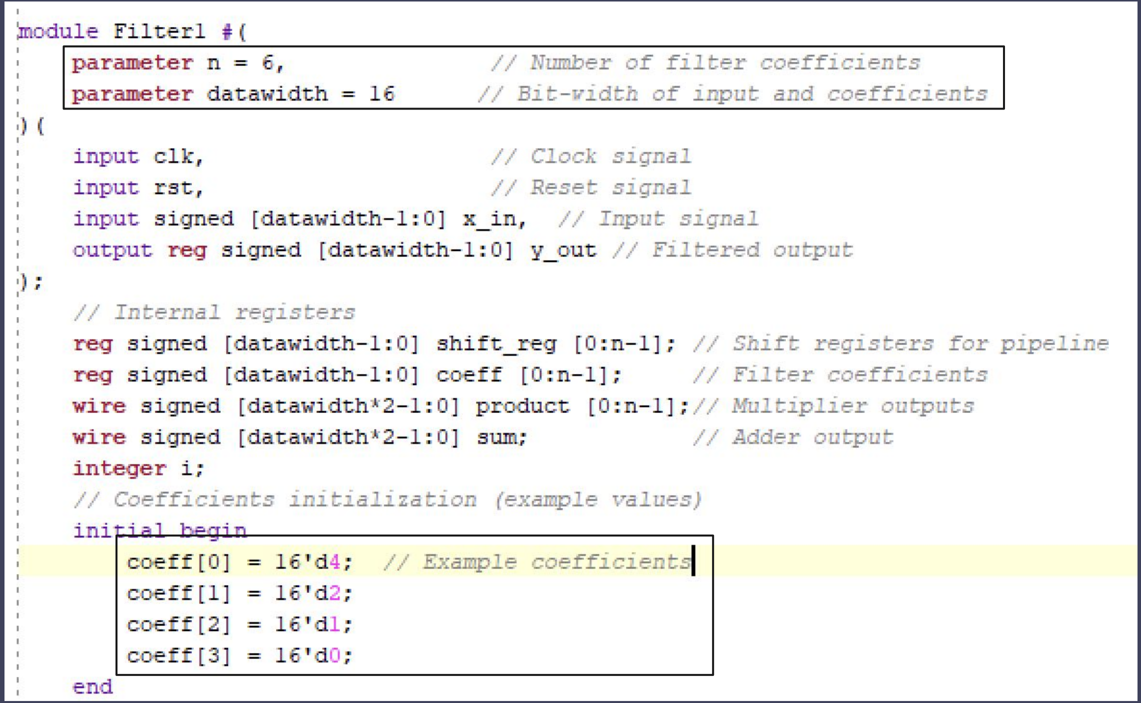
tx_done_tick asserted



BLOCK DIAGRAM

CODE SNIPPET

Convolution Filter Module



```
module Filter1 #(
    parameter n = 6,                  // Number of filter coefficients
    parameter datawidth = 16        // Bit-width of input and coefficients
)(
    input clk,                        // Clock signal
    input rst,                        // Reset signal
    input signed [datawidth-1:0] x_in,  // Input signal
    output reg signed [datawidth-1:0] y_out // Filtered output
);
    // Internal registers
    reg signed [datawidth-1:0] shift_reg [0:n-1]; // Shift registers for pipeline
    reg signed [datawidth-1:0] coeff [0:n-1];     // Filter coefficients
    wire signed [datawidth*2-1:0] product [0:n-1];// Multiplier outputs
    wire signed [datawidth*2-1:0] sum;            // Adder output
    integer i;
    // Coefficients initialization (example values)
    initial begin
        coeff[0] = 16'd4;  // Example coefficients
        coeff[1] = 16'd2;
        coeff[2] = 16'd1;
        coeff[3] = 16'd0;

    end
```

Filter Size + Precision of Arithmetic is Parameterized

RESULTS

Expected Output Length
N = (6+3) - 1 = 8

Input: 6-point signal

X[n] = [2 4 8 16 32 64]

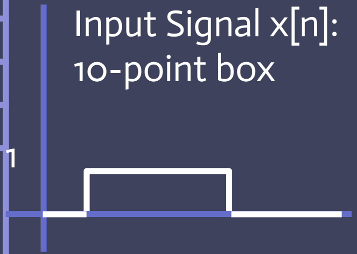
Filter: 3-point signal

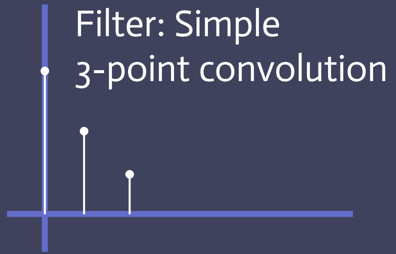
Output: 8-point signal

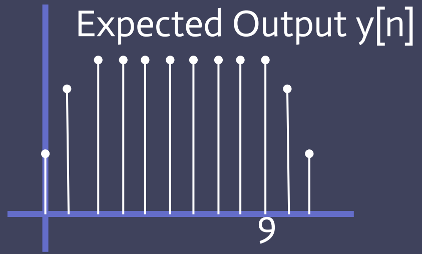
y[n] = [-8 -20 -42 -84 88 -80 96 64]

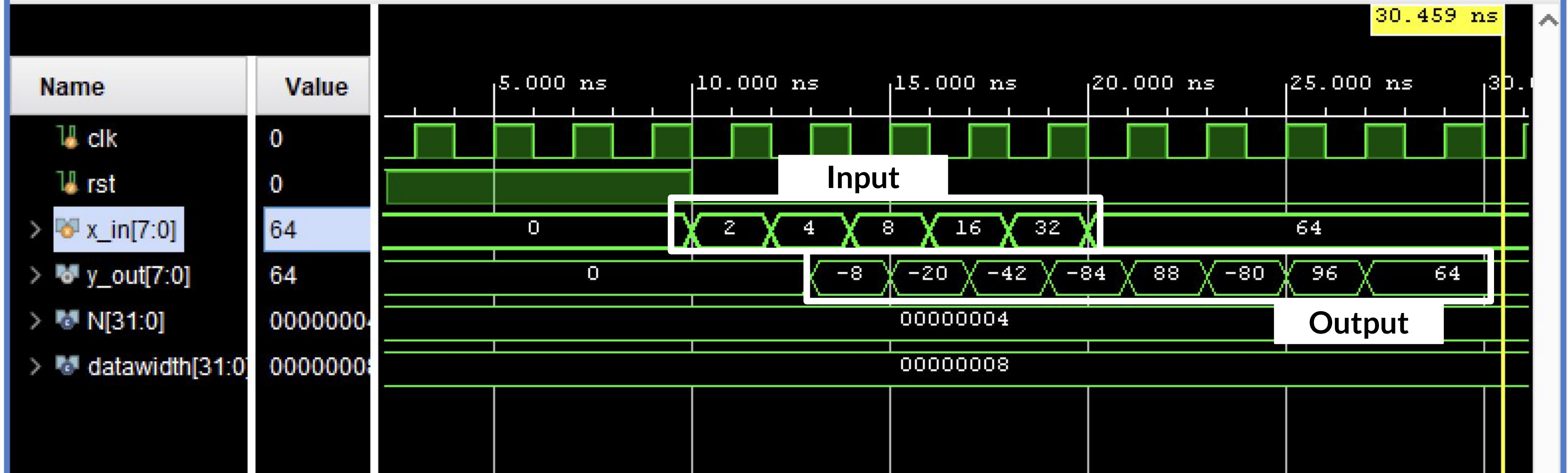
$$\begin{tikzcd}
x[n] \arrow[r] & \boxed{h[n]} \arrow[r] & y[n] \\
 & h[n] = [-4\ -2\ -1\ 0] \arrow[u] &
\end{tikzcd}$$

Convolution filter Testbench

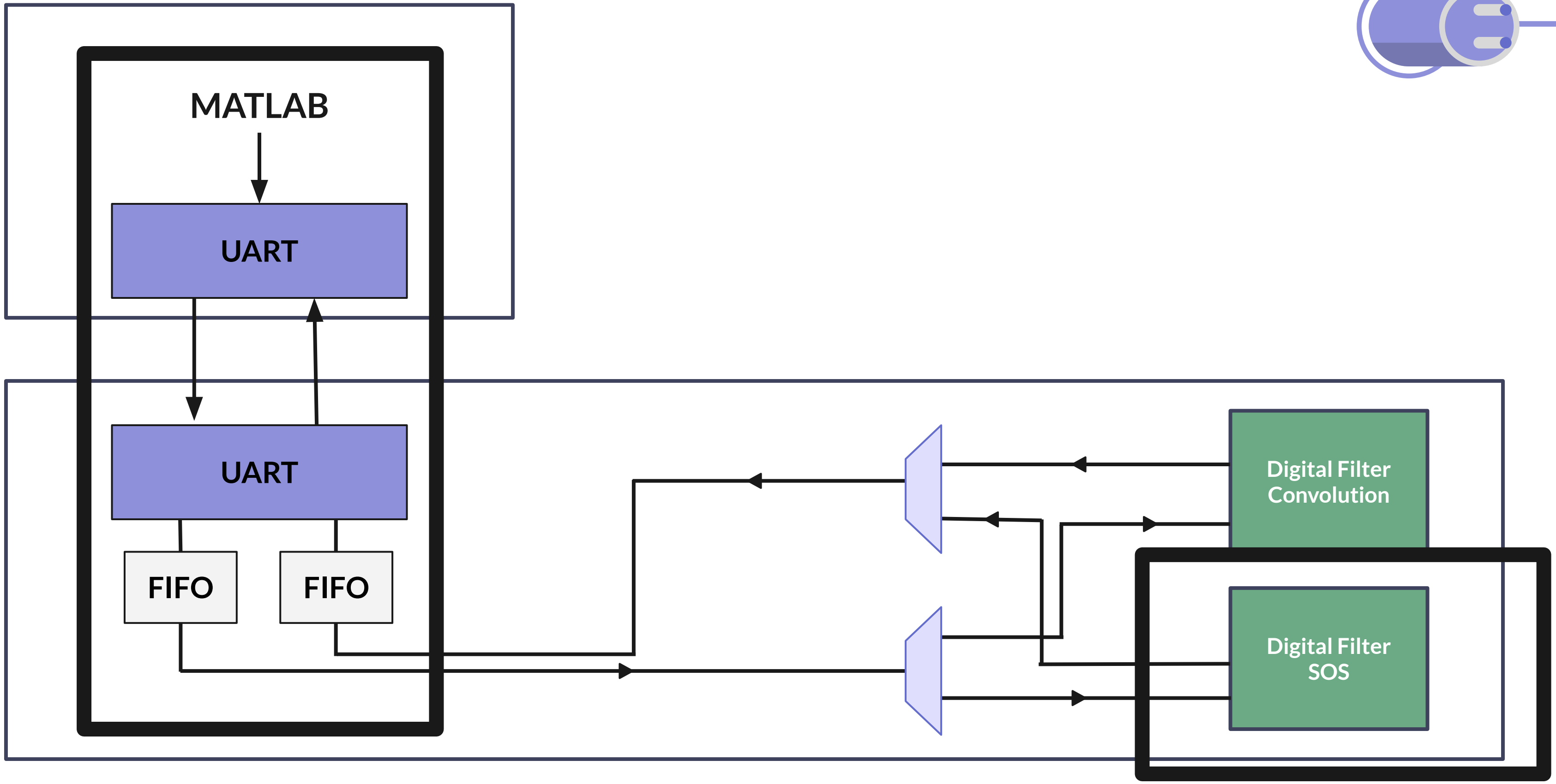






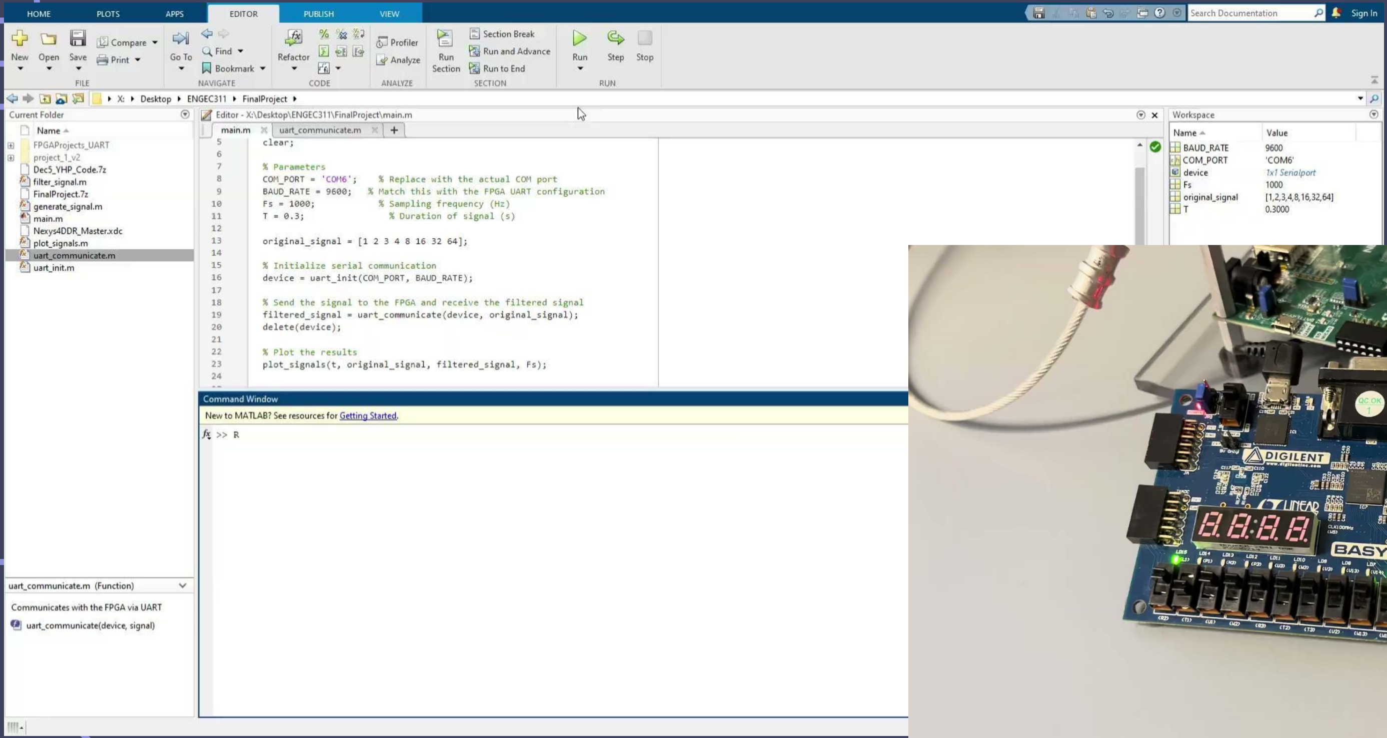


```
Name              Value
clk               0
rst               0
x_in[7:0]         64        0 | 2 | 4 | 8 | 16 | 32 | 64
y_out[7:0]        64        0 | -8 | -20 | -42 | -84 | 88 | -80 | 96 | 64
N[31:0]           0000000   00000004
datawidth[31:0]   0000000   00000008
```



BLOCK DIAGRAM

Demo of FIR Filter (SOS) Function



X: ▸ Desktop ▸ ENGEC311 ▸ FinalProject ▸

Current Folder

- FPGAProjects_UART
- project_1_v2
- Dec5_YHP_Code.7z
- filter_signal.m
- FinalProject.7z
- generate_signal.m
- main.m
- Nexys4DDR_Master.xdc
- plot_signals.m
- uart_communicate.m
- uart_init.m

uart_communicate.m (Function)

Communicates with the FPGA via UART

uart_communicate(device, signal)

Editor - X:\Desktop\ENGEC311\FinalProject\main.m

main.m | uart_communicate.m

```matlab
clear;

% Parameters
COM_PORT = 'COM6';    % Replace with the actual COM port
BAUD_RATE = 9600;   % Match this with the FPGA UART configuration
Fs = 1000;            % Sampling frequency (Hz)
T = 0.3;                % Duration of signal (s)

original_signal = [1 2 3 4 8 16 32 64];

% Initialize serial communication
device = uart_init(COM_PORT, BAUD_RATE);

% Send the signal to the FPGA and receive the filtered signal
filtered_signal = uart_communicate(device, original_signal);
delete(device);

% Plot the results
plot_signals(t, original_signal, filtered_signal, Fs);
```

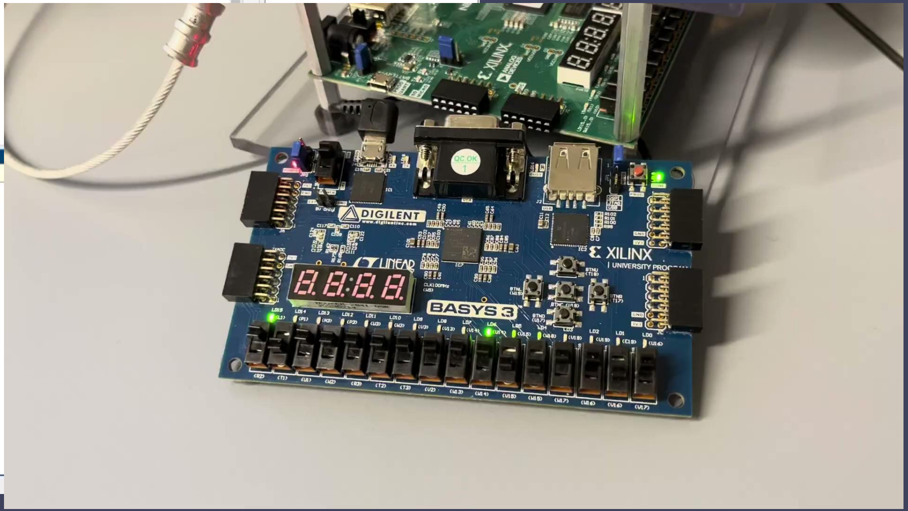
Workspace

Name	Value
BAUD_RATE	9600
COM_PORT	'COM6'
device	1x1 Serialport
Fs	1000
original_signal	[1,2,3,4,8,16,32,64]
T	0.3000

Command Window

New to MATLAB? See resources for Getting Started.

```
>> R
```



SUCCESSES

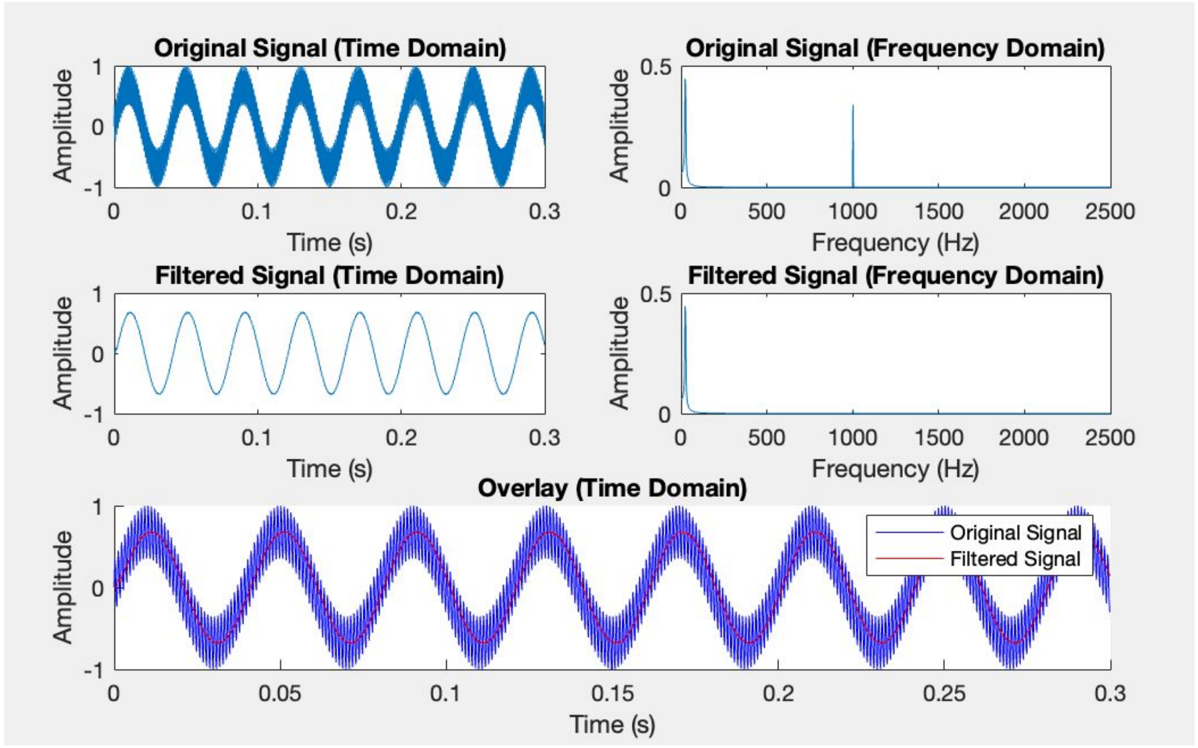
- Currently is able to **receive and transmit data**
- The SOS filter and convolution filter are able to **adjust to different signals**, able to vary with the number of coefficients and number of bits that represent the coefficients numbers
- Convolution Module Verified in Testbenches

FAILURES

- Floating point numbers not fully verified and implemented
- Optimization for Speed (delay in code in form of debouncers)
- Full Duplex-UART (using a buffer for now)
- SOS Filter (truncation and overflow is causing the error)
- Not storing the filtered data correctly on the FPGA, resulting in incorrect data being transmitted.

FUTURE DIRECTIONS SLIDE

- Mux to control between convolution and SOS – right now we need to program in separately
- Being able dynamically program filter coefficients.



Questions?

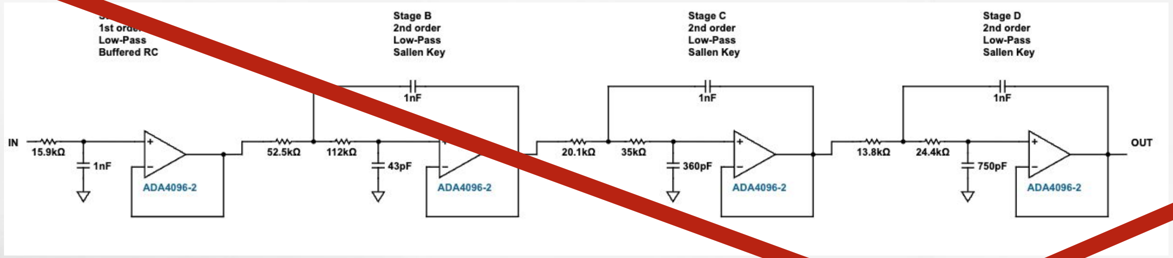
Digital Filtering using FPGA

ENGEC311: Final Project

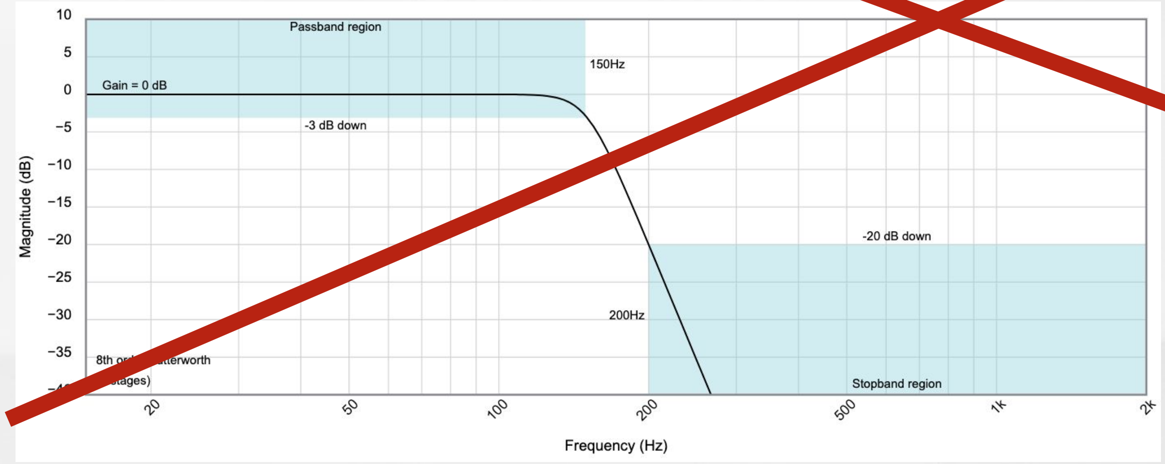
Group Members: Alex, Heewon, Vanshika, Yash

Supplemental Information

Analog Filter



1st order Low-Pass Buffered RC
Stage B 2nd order Low-Pass Sallen Key
Stage C 2nd order Low-Pass Sallen Key
Stage D 2nd order Low-Pass Sallen Key
IN
15.9kΩ
1nF
ADA4096-2
52.5kΩ
112kΩ
43pF
1nF
ADA4096-2
20.1kΩ
35kΩ
360pF
1nF
ADA4096-2
13.8kΩ
24.4kΩ
750pF
1nF
ADA4096-2
OUT

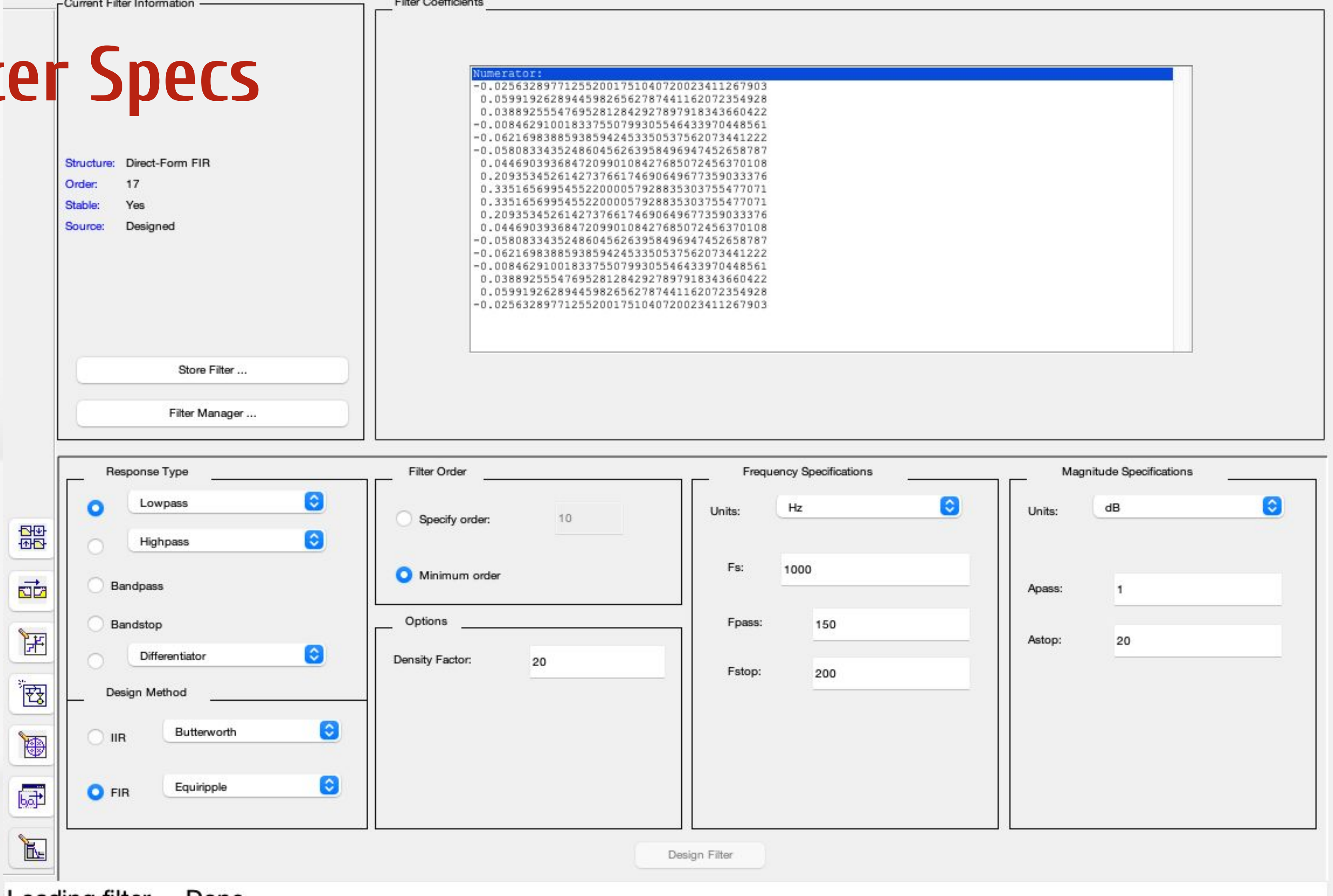


Very complex and hard to build

Tolerances of Resistors and Capacitors used need to be very low increasing cost and development time.

Designed using
Analog Devices: Analog Filter Wizard

Filter Specs



Current Filter Information

Structure: Direct-Form FIR
Order: 17
Stable: Yes
Source: Designed

Filter Coefficients

```
Numerator:
-0.025632897712552001751040720023411267903
 0.059919262894459826562787441162072354928
 0.038892555476952812842927897918343660422
-0.008462910018337550799305546433970448561
-0.062169838859385942453350537562073441222
-0.058083343524860456263958496947452658787
 0.044690393684720990108427685072456370108
 0.209353452614273766174690649677359033376
 0.335165699545522000057928835303755477071
 0.335165699545522000057928835303755477071
 0.209353452614273766174690649677359033376
 0.044690393684720990108427685072456370108
-0.058083343524860456263958496947452658787
-0.062169838859385942453350537562073441222
-0.008462910018337550799305546433970448561
 0.038892555476952812842927897918343660422
 0.059919262894459826562787441162072354928
-0.025632897712552001751040720023411267903
```

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: IIR Butterworth [✓ FIR Equiripple]

Filter Order: Specify order: 10 [✓ Minimum order]

Options: Density Factor: 20

Frequency Specifications
Units: Hz
Fs: 1000
Fpass: 150
Fstop: 200

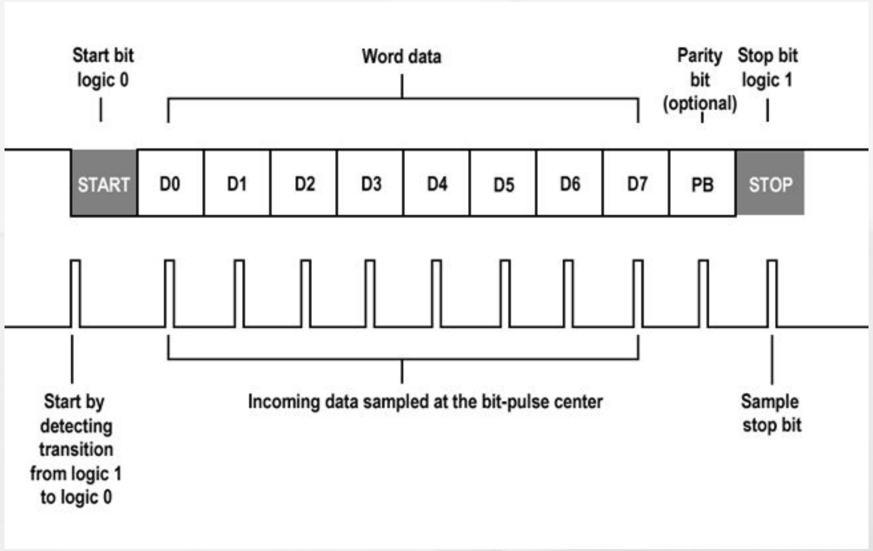
Magnitude Specifications
Units: dB
Apass: 1
Astop: 20

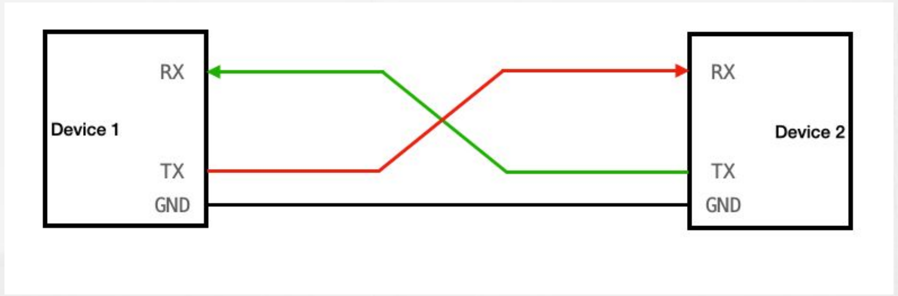
Loading filter ... Done

UART: Universal Asynchronous Receiver/Transmitter

UART will take transmit data 8 bits at a time

This is not a clocked communication protocol, meaning both devices must ‘talk’ as the same speed **(Baud Rate, bits per second)**


Start bit
logic 0
Word data
Parity bit (optional)
Stop bit logic 1
START
D0
D1
D2
D3
D4
D5
D6
D7
PB
STOP
Start by detecting transition from logic 1 to logic 0
Incoming data sampled at the bit-pulse center
Sample stop bit


Device 1
RX
TX
GND
Device 2
RX
TX
GND