Giaan Nguyen

Lab 3: Folded and Unfolded IIR Filter Design on FPGAs

ECEN 689-600: FPGA Information Processing Systems
Thursday, February 25, 2021

Unfolded IIR Filter Simulation

Figure 1 shows the results for simulating the unfolded IIR filter for sequence $-5, -4, \dots ,4,5$, alternating between two inputs (i.e., the first input giving $x2k = x[2k] = \{-5, -3, \dots ,3,5,5, \dots\}$ and the second giving $x2k1 = x[2k+1] = \{-4, -2, \dots ,2,4,5,5, \dots\}$. The outputs are also alternating between $y2k = y[2k]$ and $y2k1 = y[2k+1]$. When compared to the IIR filter results from Lab 1, we can see that the values are the same.

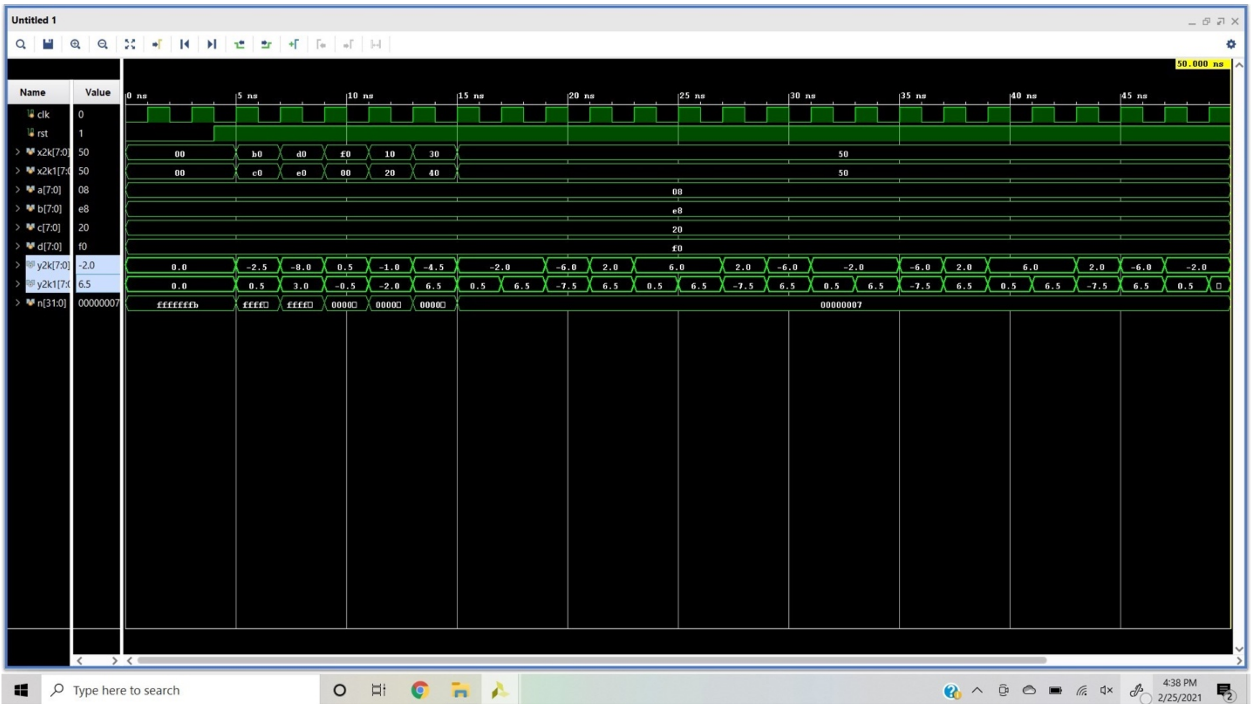
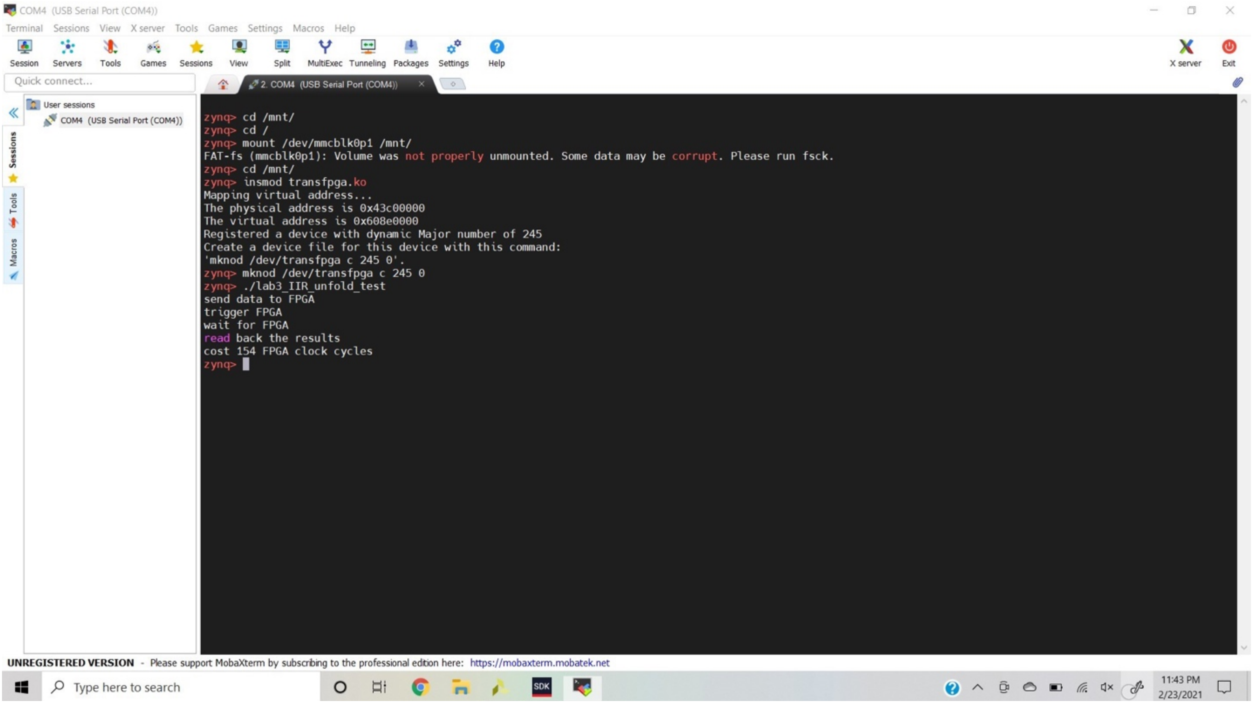


Figure 1. Simulation of an unfolded IIR filter.

Unfolded IIR Filter FPGA Outputs

Figure 2 shows the terminal output when implementing the unfolded IIR filter design on the FPGA, which shows that it takes 154 clock cycles to filter the input noisy sine wave. Figure 3 shows the noisy sine wave and the filtered sine wave. Interestingly, the filtered wave still lies within the envelope of the original wave and in fact still maintains some oscillatory behavior, though now more attenuated.



COM4 (USB Serial Port (COM4))
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. COM4 (USB Serial Port (COM4))
User sessions
COM4 (USB Serial Port (COM4))
Sessions Tools Macros
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
Type here to search
SDK
11:43 PM
2/23/2021

```
zynq> cd /mnt/
zynq> cd /
zynq> mount /dev/mmcblk0p1 /mnt/
FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
zynq> cd /mnt/
zynq> insmod transfpga.ko
Mapping virtual address...
The physical address is 0x43c00000
The virtual address is 0x608e0000
Registered a device with dynamic Major number of 245
Create a device file for this device with this command:
'mknod /dev/transfpga c 245 0'.
zynq> mknod /dev/transfpga c 245 0
zynq> ./lab3_IIR_unfold_test
send data to FPGA
trigger FPGA
wait for FPGA
read back the results
cost 154 FPGA clock cycles
zynq> 
```

Figure 2. Terminal output from implementing the unfolded IIR filter on an FPGA.

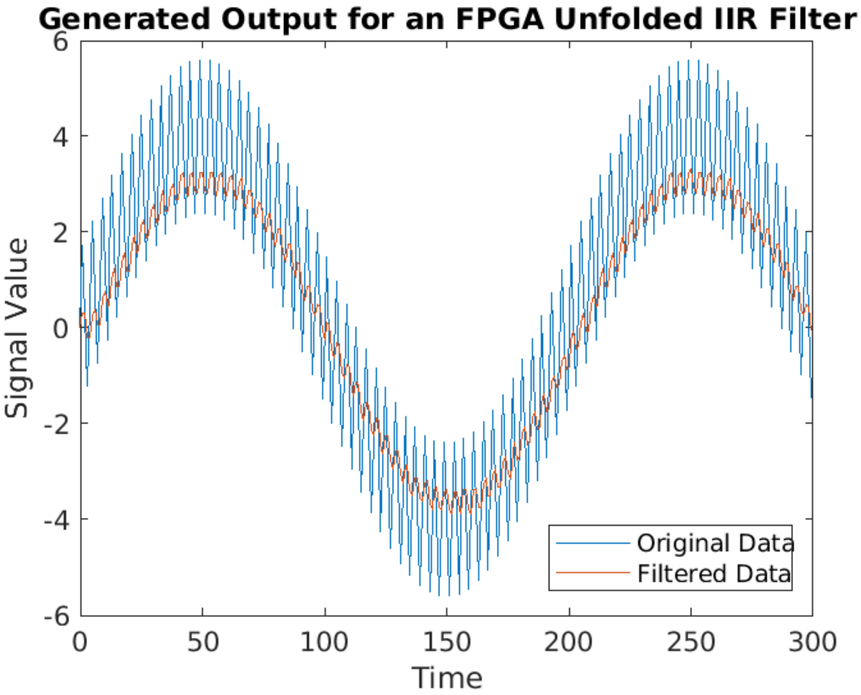
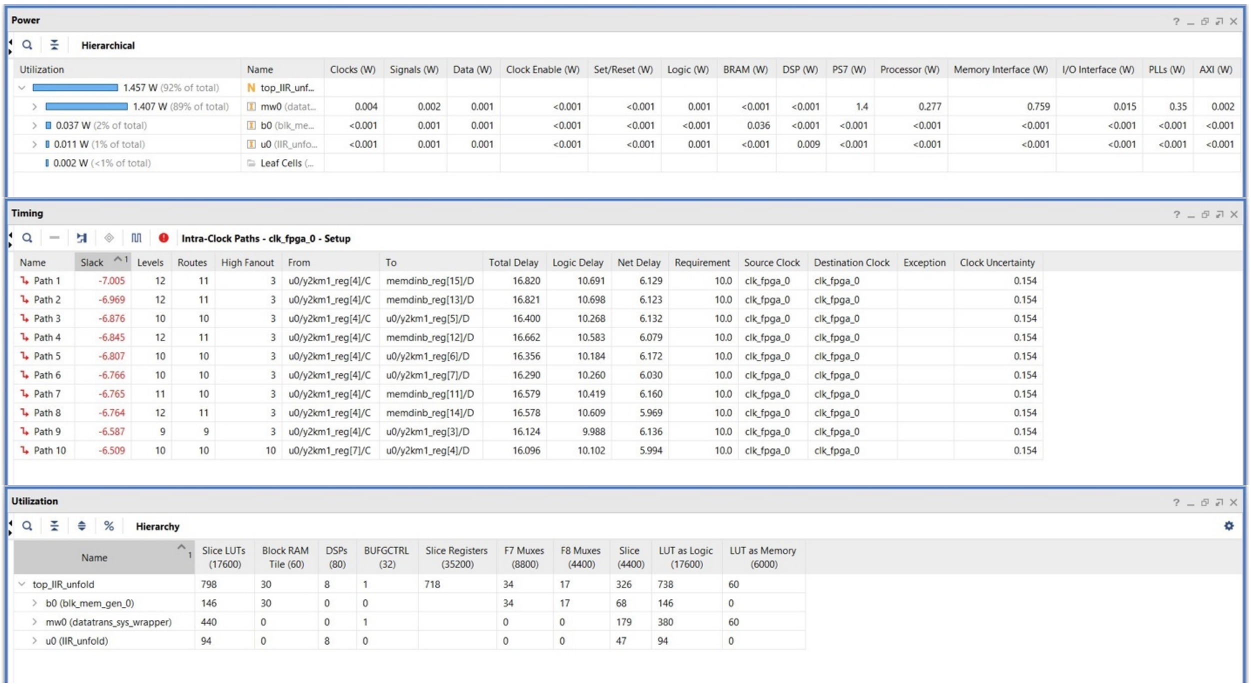


Figure 3. Original waveform and filtered waveform generated from the unfolded IIR filter implemented on the FPGA.

Unfolded IIR Filter Implementation Summary

Figure 4 shows the power, timing, and utilization summaries for the implementation of the unfolded IIR filter.



Power

Hierarchical

Utilization	Name	Clocks (W)	Signals (W)	Data (W)	Clock Enable (W)	Set/Reset (W)	Logic (W)	BRAM (W)	DSP (W)	PS7 (W)	Processor (W)	Memory Interface (W)	I/O Interface (W)	PLLs (W)	AXI (W)
1.457 W (92% of total)	top_IIR_unf...														
1.407 W (89% of total)	mw0 (datat...	0.004	0.002	0.001	<0.001	<0.001	0.001	<0.001	<0.001	1.4	0.277	0.759	0.015	0.35	0.002
0.037 W (2% of total)	b0 (blk_me...	<0.001	0.001	0.001	<0.001	<0.001	<0.001	0.036	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.011 W (1% of total)	u0 (IIR_unfo...	<0.001	0.001	0.001	<0.001	<0.001	0.001	<0.001	0.009	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.002 W (<1% of total)	Leaf Cells (...														

Timing

Intra-Clock Paths - clk_fpga_0 - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	-7.005	12	11	3	u0/y2km1_reg[4]/C	memdinb_reg[15]/D	16.820	10.691	6.129	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 2	-6.969	12	11	3	u0/y2km1_reg[4]/C	memdinb_reg[13]/D	16.821	10.698	6.123	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 3	-6.876	10	10	3	u0/y2km1_reg[4]/C	u0/y2km1_reg[5]/D	16.400	10.268	6.132	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 4	-6.845	12	11	3	u0/y2km1_reg[4]/C	memdinb_reg[12]/D	16.662	10.583	6.079	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 5	-6.807	10	10	3	u0/y2km1_reg[4]/C	u0/y2km1_reg[6]/D	16.356	10.184	6.172	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 6	-6.766	10	10	3	u0/y2km1_reg[4]/C	u0/y2km1_reg[7]/D	16.290	10.260	6.030	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 7	-6.765	11	10	3	u0/y2km1_reg[4]/C	memdinb_reg[11]/D	16.579	10.419	6.160	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 8	-6.764	12	11	3	u0/y2km1_reg[4]/C	memdinb_reg[14]/D	16.578	10.609	5.969	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 9	-6.587	9	9	3	u0/y2km1_reg[4]/C	u0/y2km1_reg[3]/D	16.124	9.988	6.136	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 10	-6.509	10	10	10	u0/y2km1_reg[7]/C	u0/y2km1_reg[4]/D	16.096	10.102	5.994	10.0	clk_fpga_0	clk_fpga_0		0.154

Utilization

Hierarchy

Name	Slice LUTs (17600)	Block RAM Tile (60)	DSPs (80)	BUFGCTRL (32)	Slice Registers (35200)	F7 Muxes (8800)	F8 Muxes (4400)	Slice (4400)	LUT as Logic (17600)	LUT as Memory (6000)
top_IIR_unfold	798	30	8	1	718	34	17	326	738	60
b0 (blk_mem_gen_0)	146	30	0	0		34	17	68	146	0
mw0 (datatrans_sys_wrapper)	440	0	0	1		0	0	179	380	60
u0 (IIR_unfold)	94	0	8	0		0	0	47	94	0

Figure 4. Power (top), timing (middle), and utilization (bottom) summaries for the unfolded IIR filter.

The requirement column in the timing summary details the target clock period, and the slack column details the worst negative slack (WNS). The clock period can be defined as the target clock period minus the WNS. Since all entries in the requirement column are the same, we can define the maximum clock frequency and maximum throughput as:

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{(10.0 - \min\{WNS\})\ [ns]} = \frac{1}{\big(10.0 - (-7.005)\big)\ [ns]} = 0.059\ [GHz] = 59\ [MHz]$$

$$S_{max} = \frac{input\ bits \times f_{max}}{clock\ cycles} = \frac{(2 \times 8) \times 59}{154} = 6.13\ [Mbps]$$

Folded IIR Filter Simulation

Figure 5 shows the results for simulating the folded IIR filter for sequence $-5, -4, \ldots, 4, 5$ for one input, utilizing only one adder and one multiplier. As a result, we expect to see an initial delay of outputs; that is, for the first few cycles, the output is zero. From Figure 5, we see that is the case. However, I could not figure out what was wrong with my Verilog code nor debug, hence the values outputted are not the same as that in Figure 1, with the exception of the first nonzero value which seems to validate that at least $x[n]$ and $x[n-1]$ are inputted correctly, though $y[n-1]$ and $y[n-2]$ remains an issue. For the sake of time, we will continue the analysis under the assumption that it somewhat works.

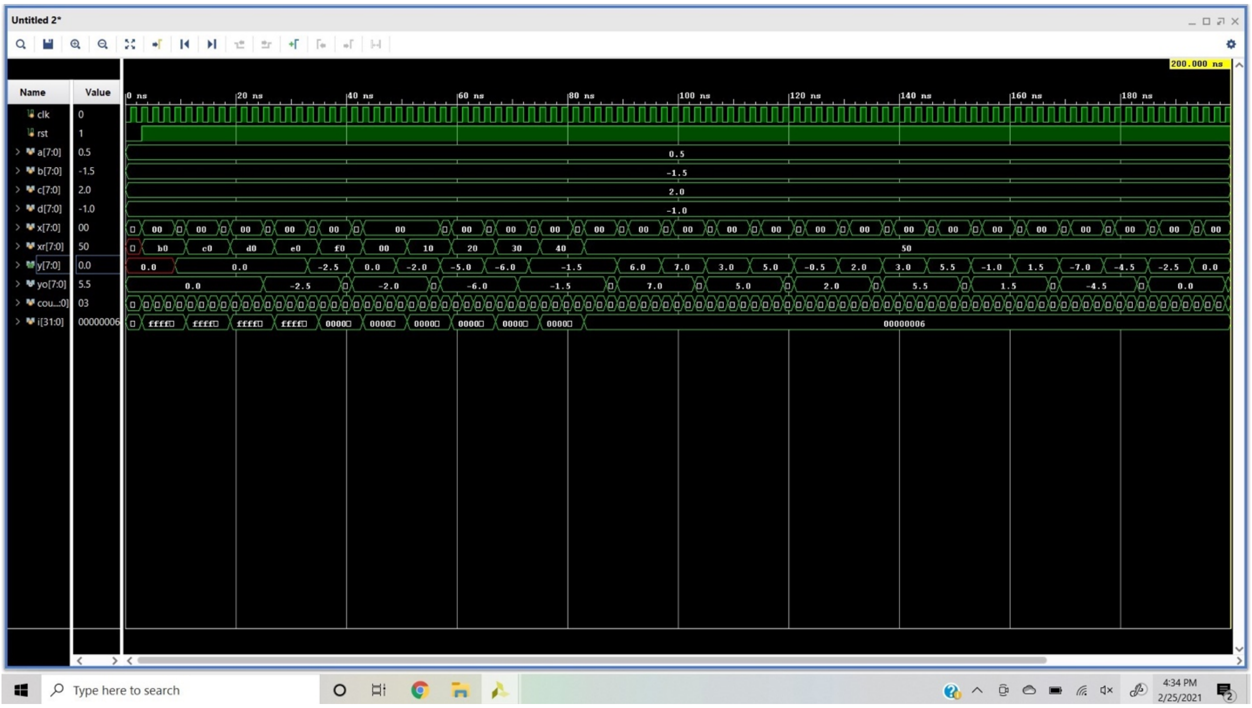
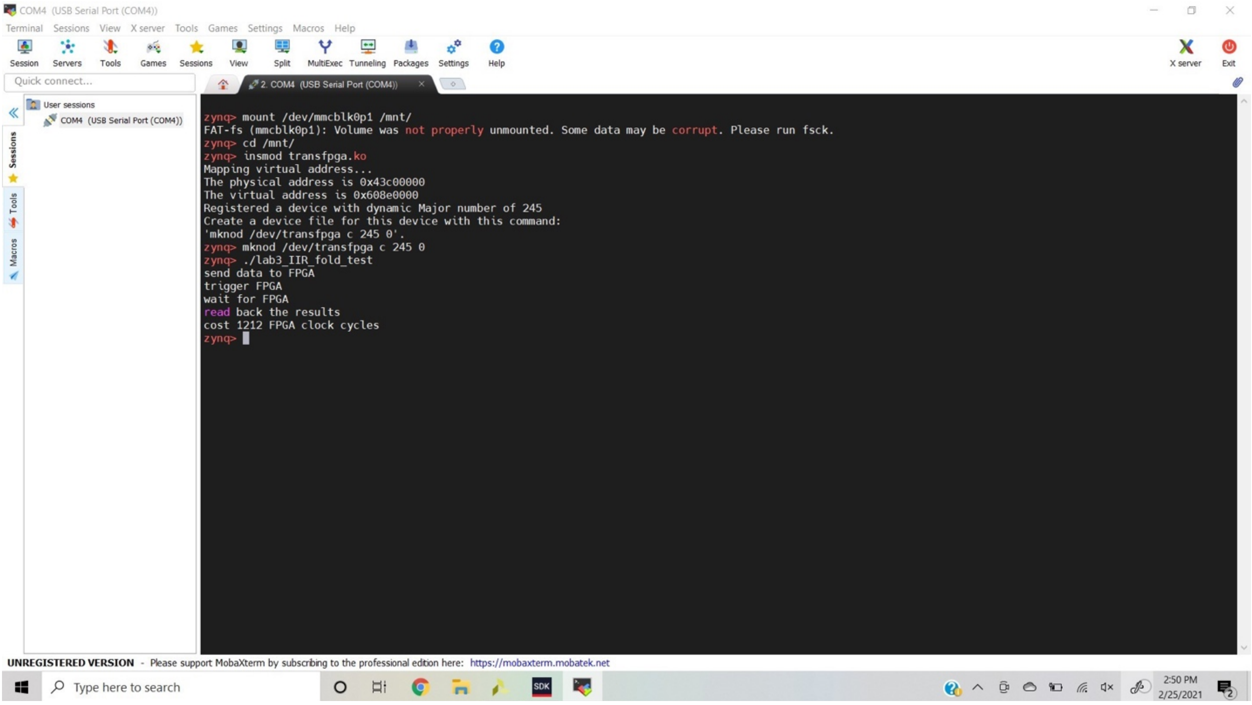


Figure 5. Simulation of a folded IIR filter.

Folded IIR Filter FPGA Outputs

Figure 6 shows the terminal output when implementing the folded IIR filter design on the FPGA, which shows that it takes 1212 clock cycles to filter the input noisy sine wave. Figure 7 shows the noisy sine wave and the filtered sine wave. Interestingly, the filtered wave somewhat still lies within the envelope of the original wave and in fact still maintains some oscillatory behavior, though now more attenuated.


COM4 (USB Serial Port (COM4))
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. COM4 (USB Serial Port (COM4))
User sessions
COM4 (USB Serial Port (COM4))
Sessions Tools Macros
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
Type here to search
2:50 PM
2/25/2021

```
zynq> mount /dev/mmcblk0p1 /mnt/
FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
zynq> cd /mnt/
zynq> insmod transfpga.ko
Mapping virtual address...
The physical address is 0x43c00000
The virtual address is 0x608e0000
Registered a device with dynamic Major number of 245
Create a device file for this device with this command:
'mknod /dev/transfpga c 245 0'.
zynq> mknod /dev/transfpga c 245 0
zynq> ./lab3_IIR_fold_test
send data to FPGA
trigger FPGA
wait for FPGA
read back the results
cost 1212 FPGA clock cycles
zynq>
```

Figure 6. Terminal output from implementing the folded IIR filter on an FPGA.

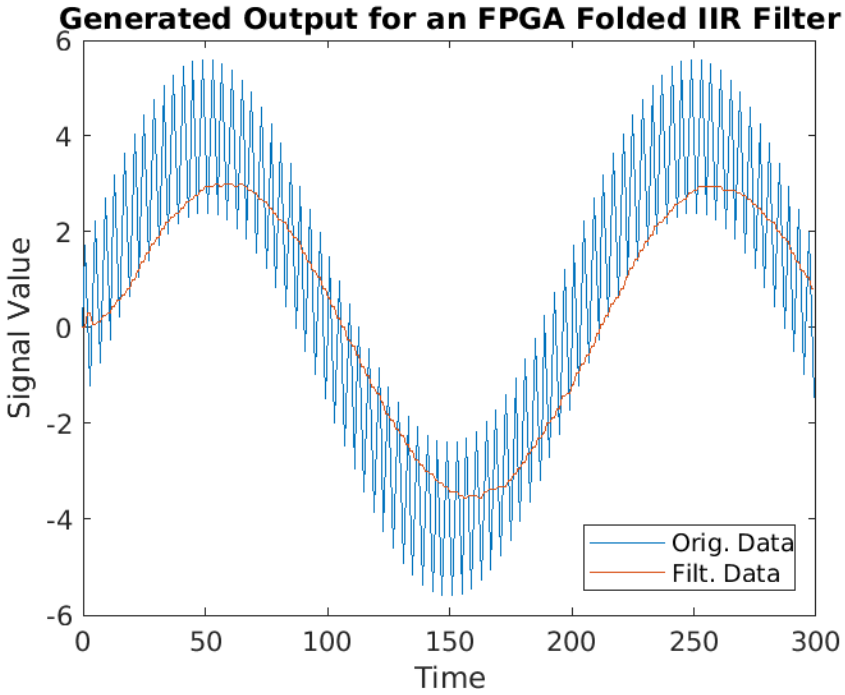
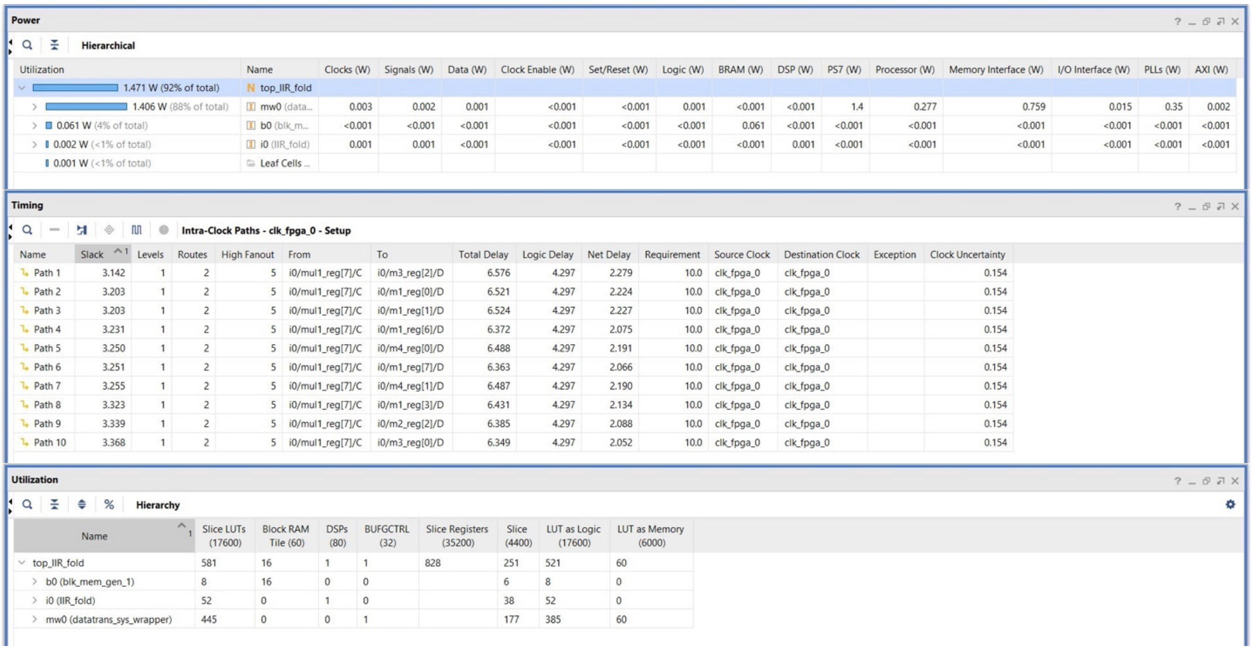


Figure 7. Original waveform and filtered waveform generated from the folded IIR filter implemented on the FPGA.

Folded IIR Filter Implementation Summary

Figure 8 shows the power, timing, and utilization summaries for the implementation of the folded IIR filter.



Power

Hierarchical

Utilization	Name	Clocks (W)	Signals (W)	Data (W)	Clock Enable (W)	Set/Reset (W)	Logic (W)	BRAM (W)	DSP (W)	PS7 (W)	Processor (W)	Memory Interface (W)	I/O Interface (W)	PLLs (W)	AXI (W)
1.471 W (92% of total)	top_IIR_fold														
1.406 W (88% of total)	mw0 (data...	0.003	0.002	0.001	<0.001	<0.001	0.001	<0.001	<0.001	1.4	0.277	0.759	0.015	0.35	0.002
0.061 W (4% of total)	b0 (blk_m...	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	0.061	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.002 W (<1% of total)	i0 (IIR_fold)	0.001	0.001	<0.001	<0.001	<0.001	<0.001	<0.001	0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.001 W (<1% of total)	Leaf Cells ...														

Timing

Intra-Clock Paths - clk_fpga_0 - Setup

Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	3.142	1	2	5	i0/mul1_reg[7]/C	i0/m3_reg[2]/D	6.576	4.297	2.279	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 2	3.203	1	2	5	i0/mul1_reg[7]/C	i0/m1_reg[0]/D	6.521	4.297	2.224	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 3	3.203	1	2	5	i0/mul1_reg[7]/C	i0/m1_reg[1]/D	6.524	4.297	2.227	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 4	3.231	1	2	5	i0/mul1_reg[7]/C	i0/m1_reg[6]/D	6.372	4.297	2.075	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 5	3.250	1	2	5	i0/mul1_reg[7]/C	i0/m4_reg[0]/D	6.488	4.297	2.191	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 6	3.251	1	2	5	i0/mul1_reg[7]/C	i0/m1_reg[7]/D	6.363	4.297	2.066	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 7	3.255	1	2	5	i0/mul1_reg[7]/C	i0/m4_reg[1]/D	6.487	4.297	2.190	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 8	3.323	1	2	5	i0/mul1_reg[7]/C	i0/m1_reg[3]/D	6.431	4.297	2.134	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 9	3.339	1	2	5	i0/mul1_reg[7]/C	i0/m2_reg[2]/D	6.385	4.297	2.088	10.0	clk_fpga_0	clk_fpga_0		0.154
Path 10	3.368	1	2	5	i0/mul1_reg[7]/C	i0/m3_reg[0]/D	6.349	4.297	2.052	10.0	clk_fpga_0	clk_fpga_0		0.154

Utilization

Hierarchy

Name	Slice LUTs (17600)	Block RAM Tile (60)	DSPs (80)	BUFGCTRL (32)	Slice Registers (35200)	Slice (4400)	LUT as Logic (17600)	LUT as Memory (6000)
top_IIR_fold	581	16	1	1	828	251	521	60
b0 (blk_mem_gen_1)	8	16	0	0		6	8	0
i0 (IIR_fold)	52	0	1	0		38	52	0
mw0 (datatrans_sys_wrapper)	445	0	0	1		177	385	60

Figure 8. Power (top), timing (middle), and utilization (bottom) summaries for the folded IIR filter.

As with the unfolded case, we can define the maximum clock frequency and maximum throughput as such:

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{(10.0 - \min\{WNS\})\ [ns]} = \frac{1}{(10.0 - 3.142)\ [ns]} = 146\ [MHz]$$

$$S_{max} = \frac{input\ bits \times f_{max}}{clock\ cycles} = \frac{8 \times 146}{1212} = 0.964\ [Mbps] = 964\ [Kbps]$$

Questions

1. **According to your report, what are the pros and cons for unfolding and folding?**

Even with the situation in the folded IIR filter, we can still see the advantages and disadvantages in both unfolded and folded IIR filters.

For the unfolded filter, we see that the clock cycles underwent are way less than that for the folded filter (154 clock cycles < 1212 clock cycles); we can attribute this to the unfolded filter being able to output two values per cycle versus the usual one, as well as the folded filter requiring four cycles per output. The throughput is also greater for the unfolded filter, which again makes sense due as we are getting two inputs and two outputs per cycle. However, its performance in power and utilization is not as good as its folded counterpart.

The folded filter consumes lower power than unfolded filter (0.002 W < 0.011 W) and uses less slice LUTs (52 < 94 for respective modules, 8 < 146 for respective block memories). This indicates that the folded filter utilizes less resources (i.e., less functional blocks) within the FPGA in comparison to the unfolded version, which should make sense as we are using one adder and one multiplier, versus the unfolded filter which requires more operations.

Basically, unfolding is ideal for fast computation, whereas folding is ideal for efficient computation (with respect to resources).