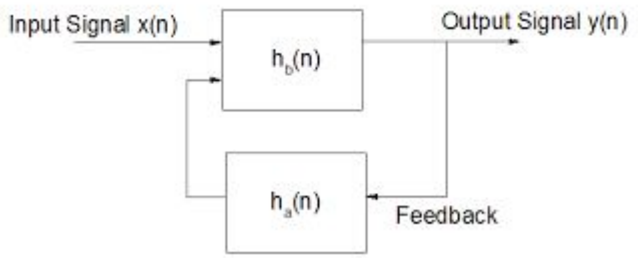
Rensselaer

why not change the world?®



IIR Filter Project Overview

Prepared By: Max Destil | Advanced VLSI Design- ECSE 6680 | 04/22/24

Project Scope and Objective

- Introduction of design of a low-pass IIR filter in MATLAB to meet specific frequency response characteristics, mirroring existing FIR filter performance
- Address the design and implementation complexities by contrasting the direct calculation approach of the IIR filter with the iterative method of the FIR filter
- Emphasize the unique challenges such as the recursive nature, stability, and quantization effects inherent to the IIR filter design process

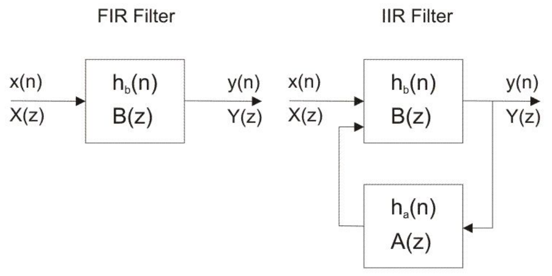


Figure 1: FIR vs. IIR

FIR vs. IIR Design Challenges

- Executed a transition from the iterative optimization process of FIR design to a calculated, formulaic approach in IIR filter order determination to match stringent frequency response criteria
- Addressed the recursive algorithmic challenges of IIR filter design, with a concentrated focus on stability and the implications of coefficient quantization to assure system reliability
- Analyzed and synthesized the theoretical and practical nuances of IIR filter complexities, applying advanced DSP principles to achieve the target specifications

MATLAB Design Process

- Utilized MATLAB's ellipord function for precision specification of the low-pass IIR filter, ensuring stringent adherence to frequency response goals
- Executed elliptic filter design using the ellip function to achieve a sharply defined rolloff characteristic with a computationally efficient filter order
- Synthesized the IIR filter algorithm to balance the minimization of passband ripple against stringent stopband attenuation requirements

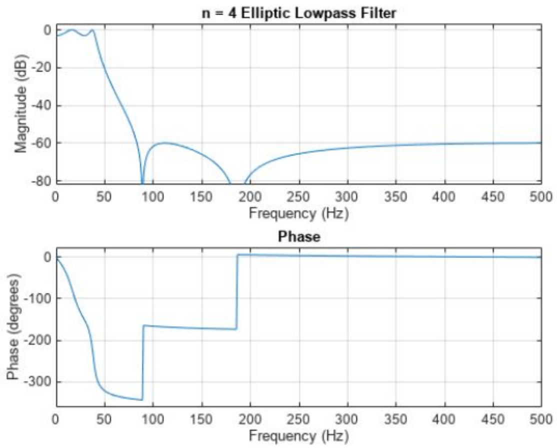


Figure 2: Example from MATLAB documentation

Transition to Hardware Design

- Verilog implementation, translating MATLAB-designed IIR filter into synthesizable RTL code, adhering to specified frequency characteristics
- Advanced pipelining to enhance data flow efficiency, reduce computational latency, and ensure high throughput in real-time processing
- Deployed parallel processing techniques, enabling concurrent computation and boosting system performance within the hardware constraints

Ensuring Filter Stability

- Developed feedback mechanism, integrating saturation arithmetic to prevent numerical overflow
- Prioritized the establishment of initial conditions and meticulously managed feedback for system stability
- Implemented robust stability controls to safeguard against potential recursive feedback instabilities
- Enforced precision regulation through COEFFICIENT_WIDTH parameters to ensure consistent performance and filter output integrity

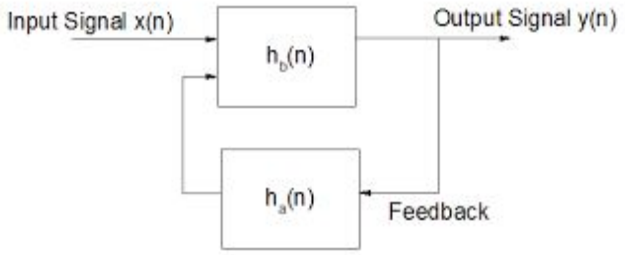


Figure 3: IIR Feedback

IIR vs. FIR Filter Performance

- Disparities in computational efficiency and resource allocation between IIR and FIR filter
- Explored the IIR filter's methodology for direct order calculation, emphasizing its contribution to optimized hardware utilization

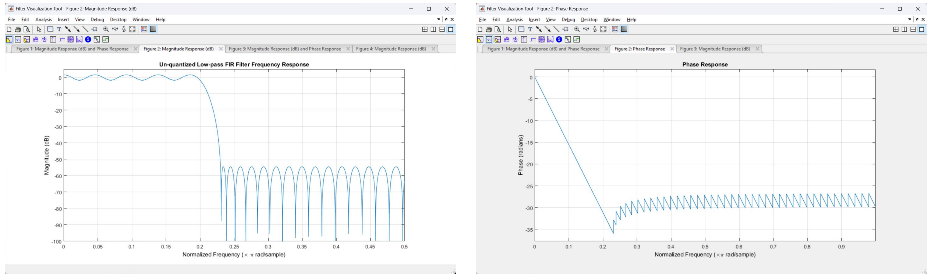


Figure 4: FIR unquantized performance

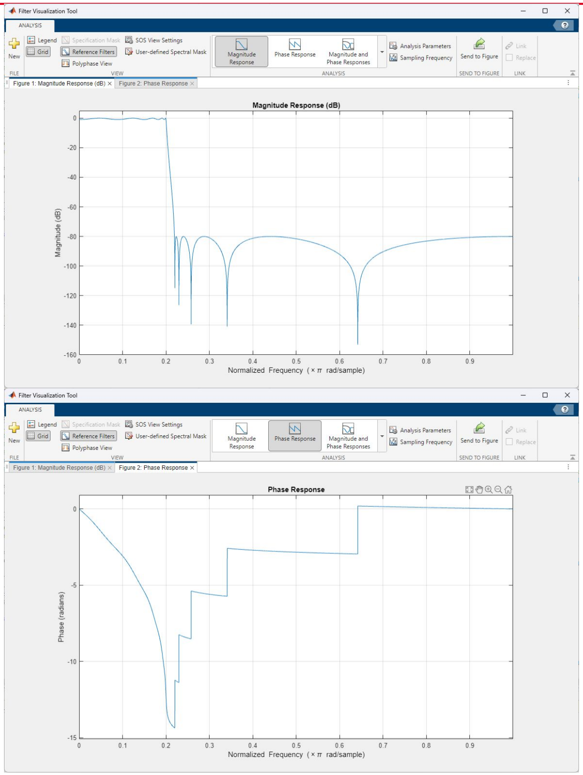
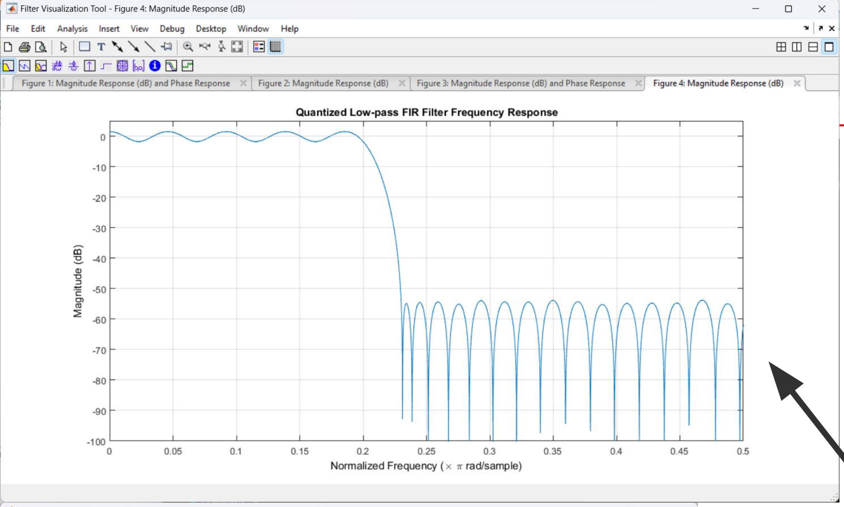
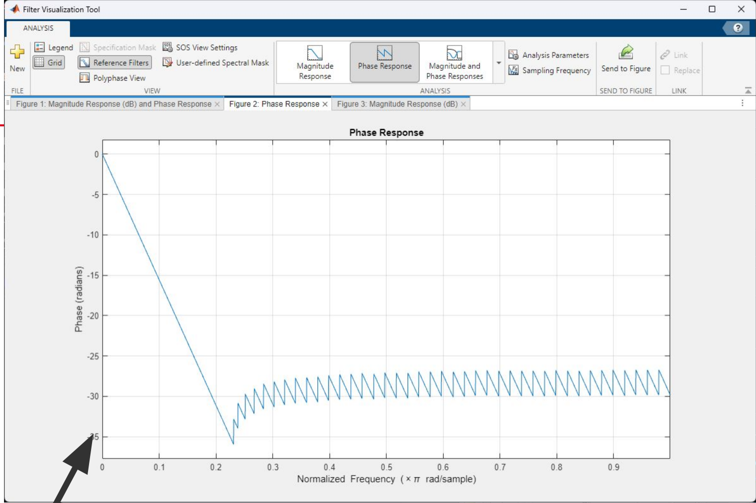
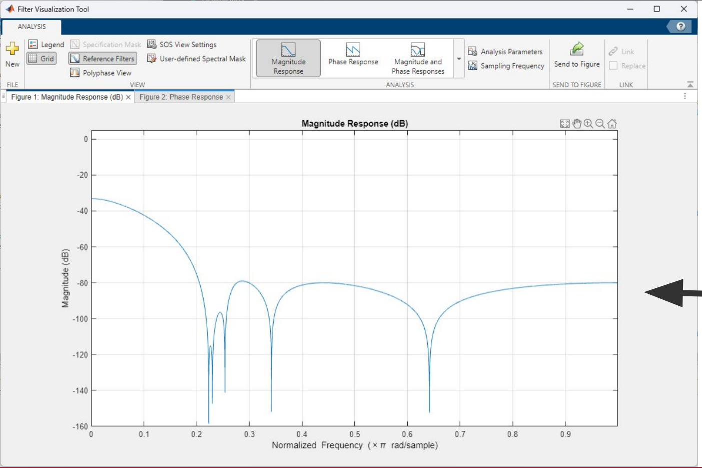


Figure 5: IIR unquantized performance







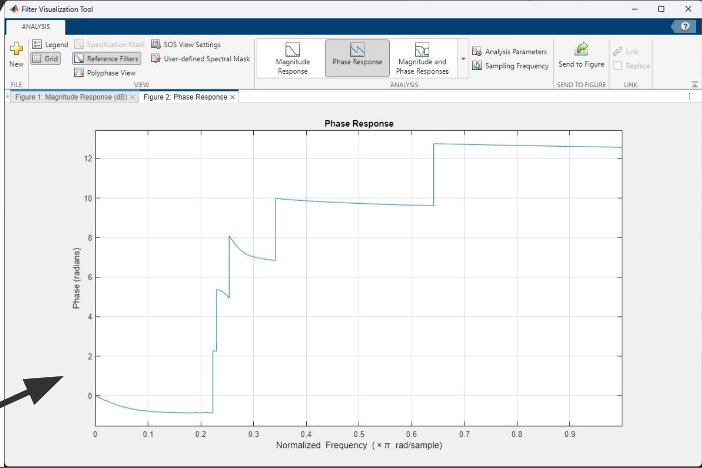


Figure 6: FIR quantized magnitude

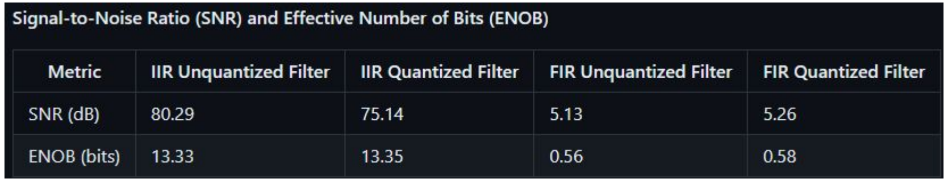
Figure 7: FIR quantized phase

Figure 8: IIR quantized magnitude

Figure 9: IIR unquantized phase

MATLAB Results and Insights

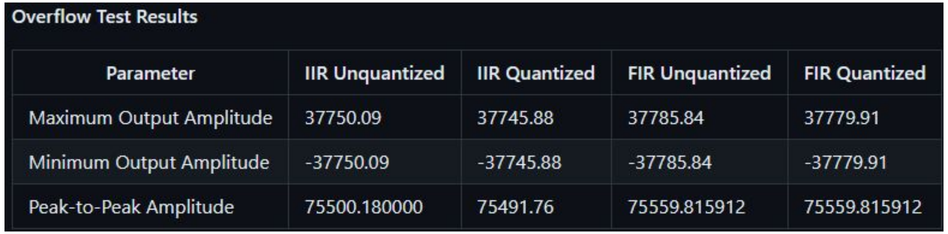
- Quantitative metrics such as SNR and ENOB as well as overflow characteristics facilitating a robust comparison of filter performances



Signal-to-Noise Ratio (SNR) and Effective Number of Bits (ENOB)

Metric	IIR Unquantized Filter	IIR Quantized Filter	FIR Unquantized Filter	FIR Quantized Filter
SNR (dB)	80.29	75.14	5.13	5.26
ENOB (bits)	13.33	13.35	0.56	0.58

Figure 10: IIR vs. FIR SNR and ENOB Comparisons



Overflow Test Results

Parameter	IIR Unquantized	IIR Quantized	FIR Unquantized	FIR Quantized
Maximum Output Amplitude	37750.09	37745.88	37785.84	37779.91
Minimum Output Amplitude	-37750.09	-37745.88	-37785.84	-37779.91
Peak-to-Peak Amplitude	75500.180000	75491.76	75559.815912	75559.815912

Figure 11: IIR vs. FIR Overflow Comparisons

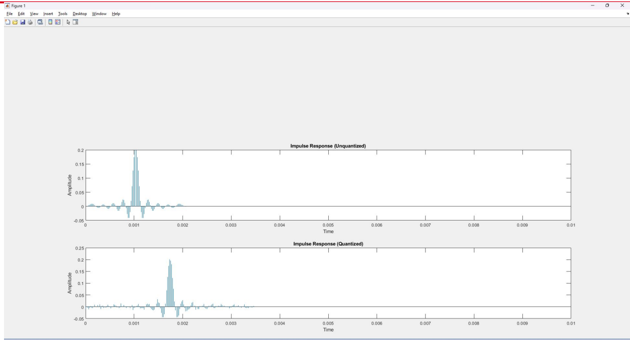


Figure 12: Impulse Response Comparison

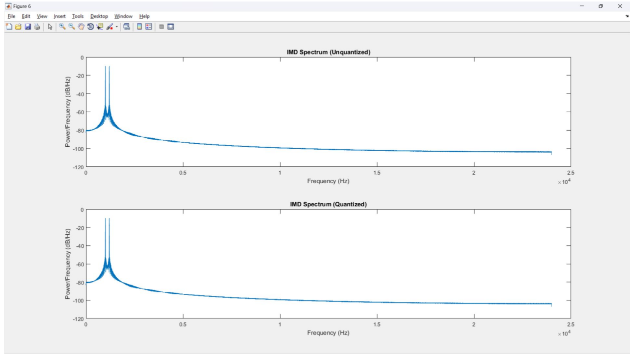
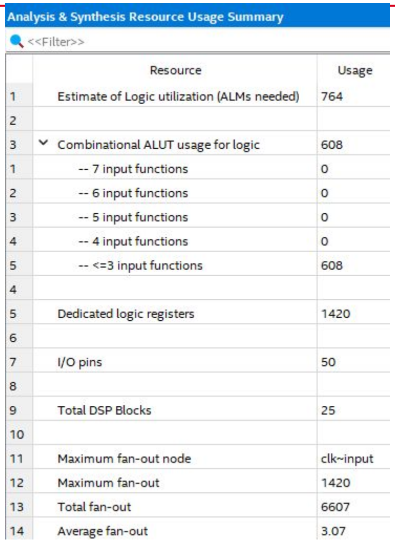


Figure 13: Intermodulation Distortion (IMD)

Hardware Performance Metrics

- High-speed clock frequency management
- FPGA resource allocation and DSP block engagement, highlighting the strategic use of Adaptive Logic Modules (ALMs)
- Granular logic utilization breakdown, showcasing the organization of combinational ALUTs and dedicated logic registers for enhanced filter operation

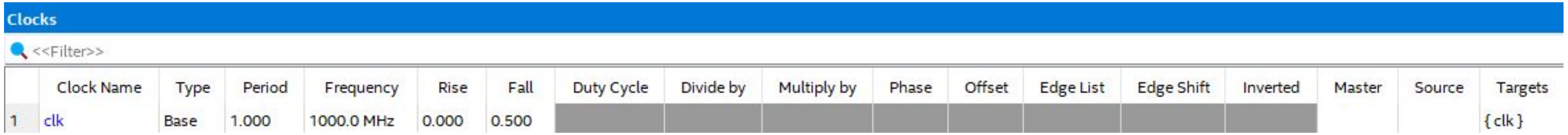


Analysis & Synthesis Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	764
2		
3	Combinational ALUT usage for logic	608
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	0
5	-- <=3 input functions	608
4		
5	Dedicated logic registers	1420
6		
7	I/O pins	50
8		
9	Total DSP Blocks	25
10		
11	Maximum fan-out node	clk~input
12	Maximum fan-out	1420
13	Total fan-out	6607
14	Average fan-out	3.07

Figure 15: Resource Usage IIR

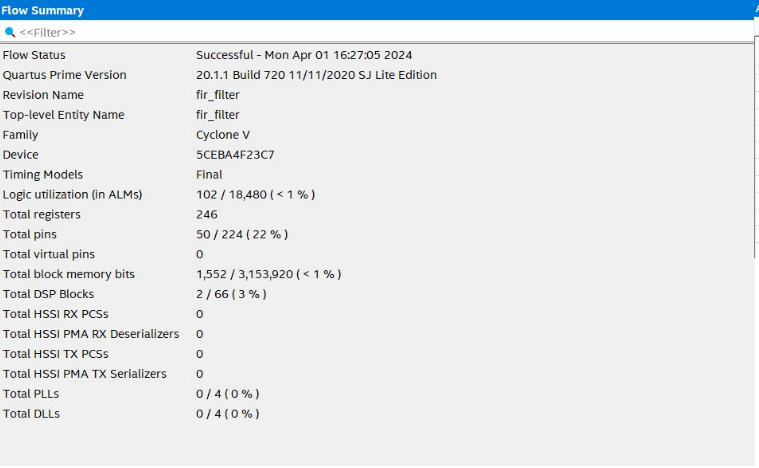


Clocks

<<Filter>>

	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle	Divide by	Multiply by	Phase	Offset	Edge List	Edge Shift	Inverted	Master	Source	Targets
1	clk	Base	1.000	1000.0 MHz	0.000	0.500											{ clk }

Figure 14: Clock Frequency Analysis Both FIR and IIR

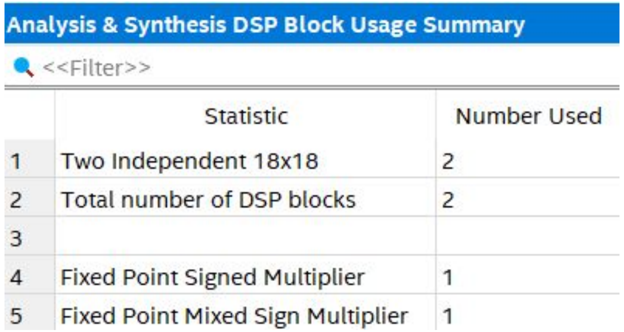


Flow Summary

<<Filter>>

Flow Status	Successful - Mon Apr 01 16:27:05 2024
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	fir_filter
Top-level Entity Name	fir_filter
Family	Cyclone V
Device	5CEBA4F23C7
Timing Models	Final
Logic utilization (in ALMs)	102 / 18,480 (< 1 %)
Total registers	246
Total pins	50 / 224 (22 %)
Total virtual pins	0
Total block memory bits	1,552 / 3,153,920 (< 1 %)
Total DSP Blocks	2 / 66 (3 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 16: Compilation Report FIR

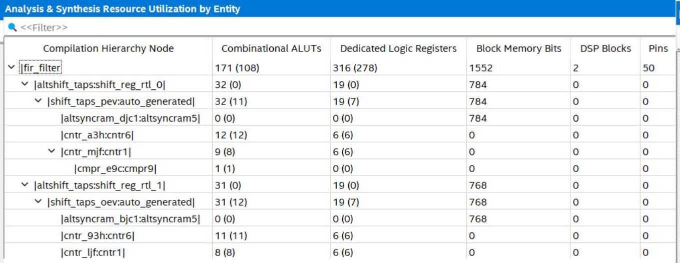


Analysis & Synthesis DSP Block Usage Summary

<<Filter>>

	Statistic	Number Used
1	Two Independent 18x18	2
2	Total number of DSP blocks	2
3		
4	Fixed Point Signed Multiplier	1
5	Fixed Point Mixed Sign Multiplier	1

Figure 17: DSP Usage FIR



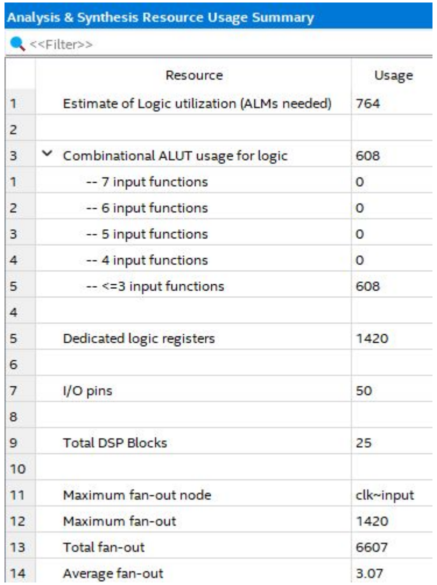
Analysis & Synthesis Resource Utilization by Entity

<<Filter>>

Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	DSP Blocks	Pins
|fir_filter	171 (108)	316 (278)	1552	2	50
|altshift_taps:shift_reg_rtl_0|	32 (0)	19 (0)	784	0	0
|shift_taps_pev:auto_generated|	32 (11)	19 (7)	784	0	0
|altsyncram_djc1:altsyncram5|	0 (0)	0 (0)	784	0	0
|cntr_a3h:cntr6|	12 (12)	6 (6)	0	0	0
|cntr_mjf:cntr1|	9 (8)	6 (6)	0	0	0
|cmpr_e9c:cmpr9|	1 (1)	0 (0)	0	0	0
|altshift_taps:shift_reg_rtl_1|	31 (0)	19 (0)	768	0	0
|shift_taps_oev:auto_generated|	31 (12)	19 (7)	768	0	0
|altsyncram_bjc1:altsyncram5|	0 (0)	0 (0)	768	0	0
|cntr_93h:cntr6|	11 (11)	6 (6)	0	0	0
|cntr_ljf:cntr1|	8 (8)	6 (6)	0	0	0

Figure 18: Synthesis Resource Utilization

**FIR

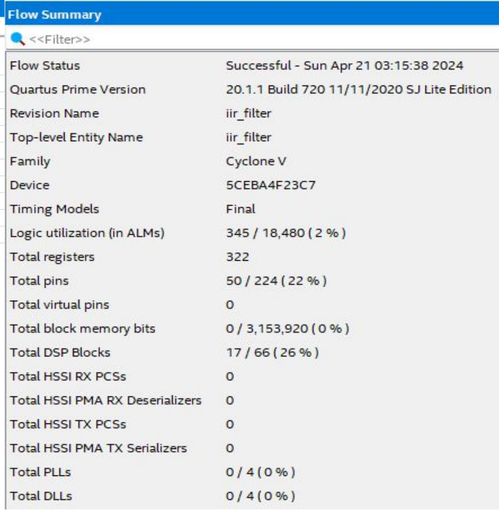


Analysis & Synthesis Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	764
2		
3	Combinational ALUT usage for logic	608
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	0
5	-- <=3 input functions	608
4		
5	Dedicated logic registers	1420
6		
7	I/O pins	50
8		
9	Total DSP Blocks	25
10		
11	Maximum fan-out node	clk~input
12	Maximum fan-out	1420
13	Total fan-out	6607
14	Average fan-out	3.07

Figure 19: Resource Usage FIR

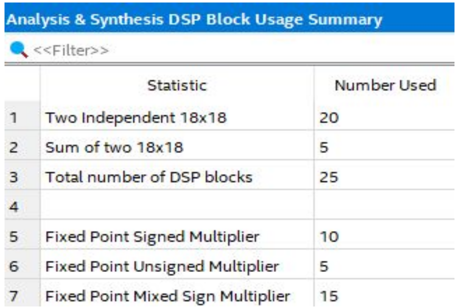


Flow Summary

<<Filter>>

Flow Status	Successful - Sun Apr 21 03:15:38 2024
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	iir_filter
Top-level Entity Name	iir_filter
Family	Cyclone V
Device	5CEBA4F23C7
Timing Models	Final
Logic utilization (in ALMs)	345 / 18,480 (2 %)
Total registers	322
Total pins	50 / 224 (22 %)
Total virtual pins	0
Total block memory bits	0 / 3,153,920 (0 %)
Total DSP Blocks	17 / 66 (26 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 20: Compilation Report IIR

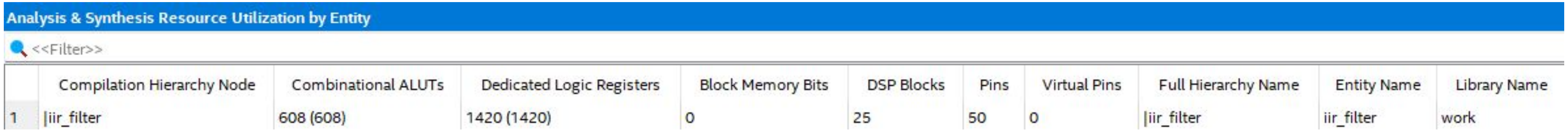


Analysis & Synthesis DSP Block Usage Summary

<<Filter>>

	Statistic	Number Used
1	Two Independent 18x18	20
2	Sum of two 18x18	5
3	Total number of DSP blocks	25
4		
5	Fixed Point Signed Multiplier	10
6	Fixed Point Unsigned Multiplier	5
7	Fixed Point Mixed Sign Multiplier	15

Figure 21: DSP Usage IIR



Analysis & Synthesis Resource Utilization by Entity

<<Filter>>

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	DSP Blocks	Pins	Virtual Pins	Full Hierarchy Name	Entity Name	Library Name
1	|iir_filter	608 (608)	1420 (1420)	0	25	50	0	|iir_filter	iir_filter	work

Figure 22: Synthesis Resource Utilization IIR

Summary

Project Synthesis: Culminated the design and implementation of an IIR filter by integrating MATLAB's analytical prowess with Verilog's robust synthesis capabilities, achieving the delicate balance between computational efficiency and system stability

Hardware Realization: Translated DSP algorithms into RTL code, successfully demonstrating the project's efficacy in a real-world VLSI environment, while ensuring that system performance was not compromised

Looking Forward: This venture into IIR filter design paves the way for my future projects in DSP, AI and VLSI design, with the aim to continually refine my approaches to complex system implementations

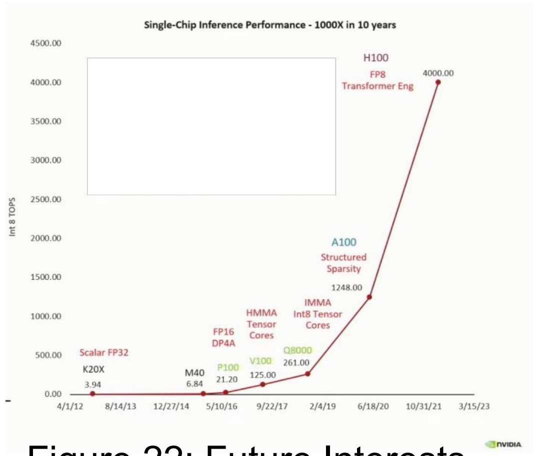


Figure 22: Future Interests

Q&A

Any questions?

RENSSELAER POLYTECHNIC INSTITUTE
KNOWLEDGE AND THOROUGHNESS
1824

Rensselaer

why not change the world?®

Resources

Title Image:

Introduction to IIR Filters - Circuit Cellar

Slide 2:

introduction-iir-filter - MIKROE

Slide 4:

Minimum order for elliptic filters - MATLAB ellipord (mathworks.com)

Slide 6:

Introduction to IIR Filters - Circuit Cellar

Slide 13:

Video Shows How Engineers Fuel Huang's Law | NVIDIA Blogs

GitHub Page:

ADVANCED-VLSI-DESIGN---ECSE-6680/Projects at main · maxdoublee/ADVANCED-VLSI-DESIGN---ECSE-6680 (github.com)