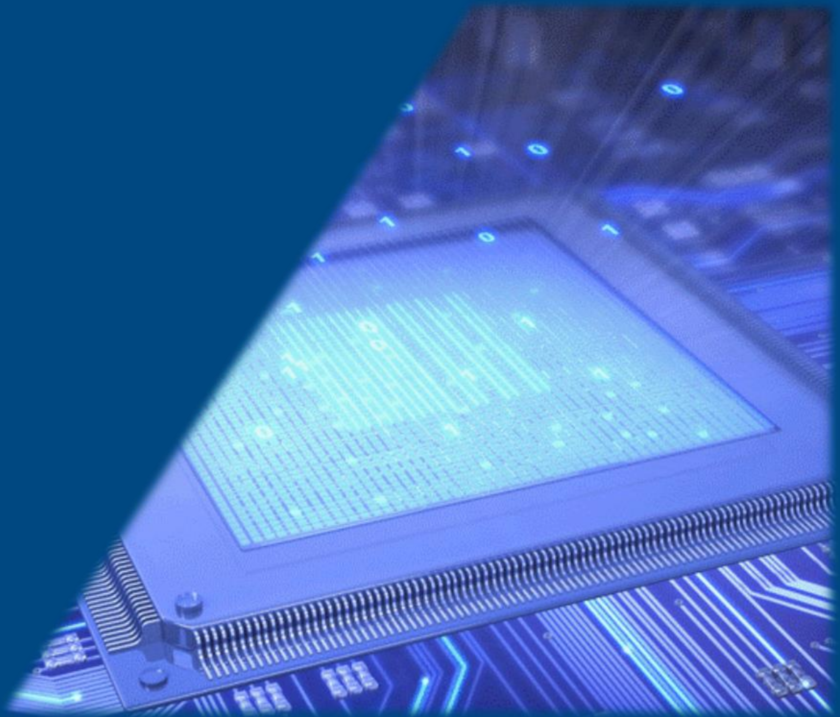
INNOVATIVE GSPS SIGNAL PROCESSING SOLUTION USING MATLAB & SIMULINK FOR FPGA/SoC

Nadereh Rooein

Principal ASIC/FPGA Application Engineer, MathWorks

nrooein@mathworks.com



Nadereh Rooein
Principal Application Engineer
nrooein@mathworks.com

Nadereh Rooein *has many years of experience in ASIC/ FPGA design and verification.*

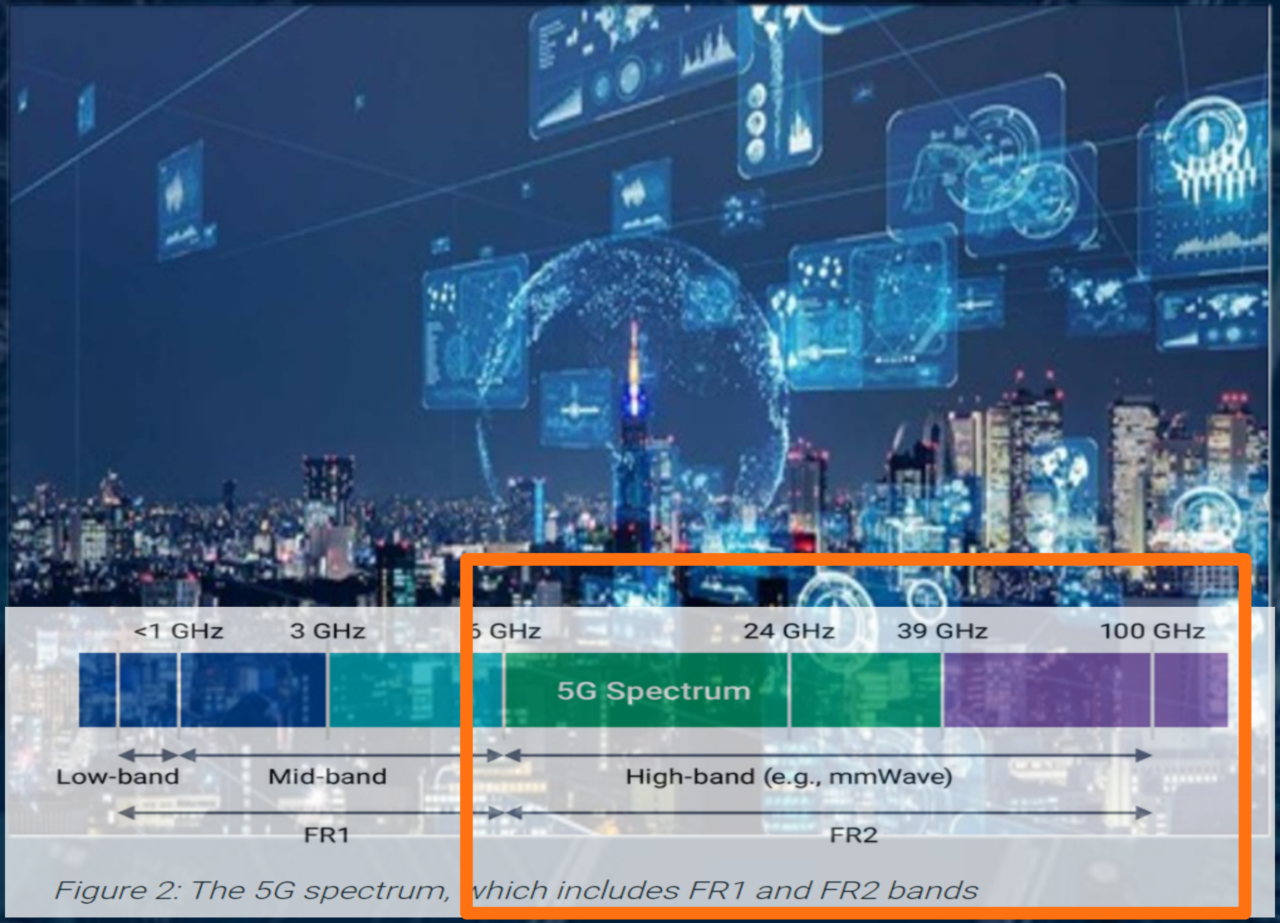
Nadereh has been a part of the Application Engineer team at MathWorks as an expert in HDL tools modeling and Verification in various application area.

This presentation, showcases MathWorks' solution that significantly eases implementing Giga Sample Per Second (GSPS) or Super-Sample-Rate (SSR) digital signal processing algorithms in MATLAB and Simulink using DSP HDL Toolbox.

Agenda

- Implementation of high throughput (GSPS) signal processing
- GSPS DDC example
- Seamless architecture selection for Optimized implementation
- Simulink simulation with Hardware Latency

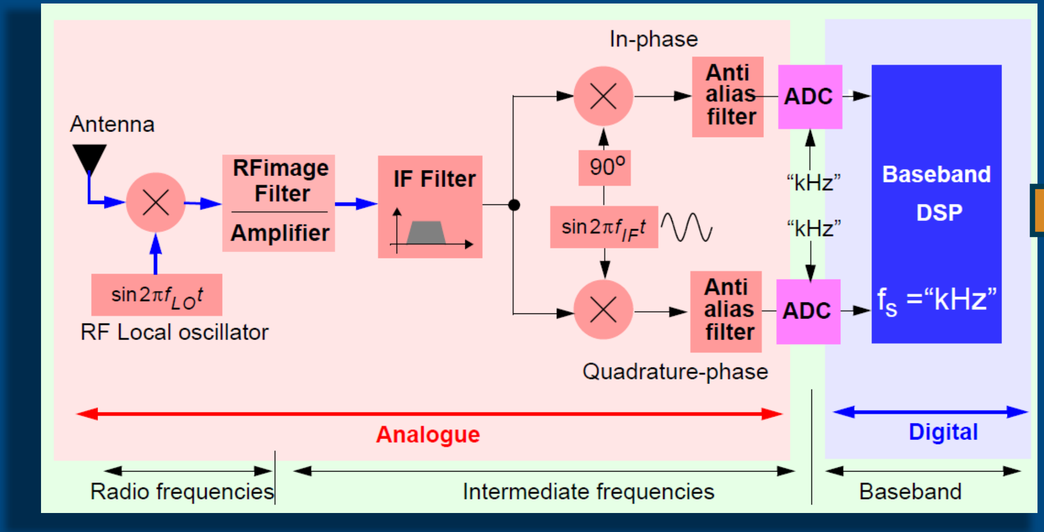
Era of High-Bandwidth GSPS Data processing

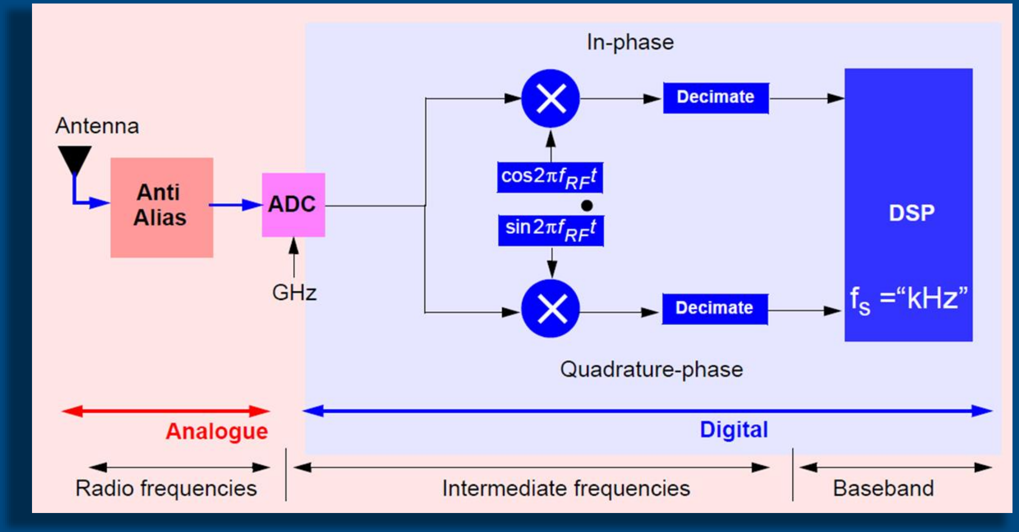


5G-FR2

Emerging Technologies Demanding Advanced DSP

In recent years, advances in analog-to-digital converters (ADC) have enabled high-bandwidth applications like RADAR, 5G (FR2) or Software-Defined Radio to deliver data from over-the-air interfaces to the DSP algorithms at “Giga Sample-Per-Second” (GSPS) rates.





GSPS Design Challenges for FPGA/ASIC

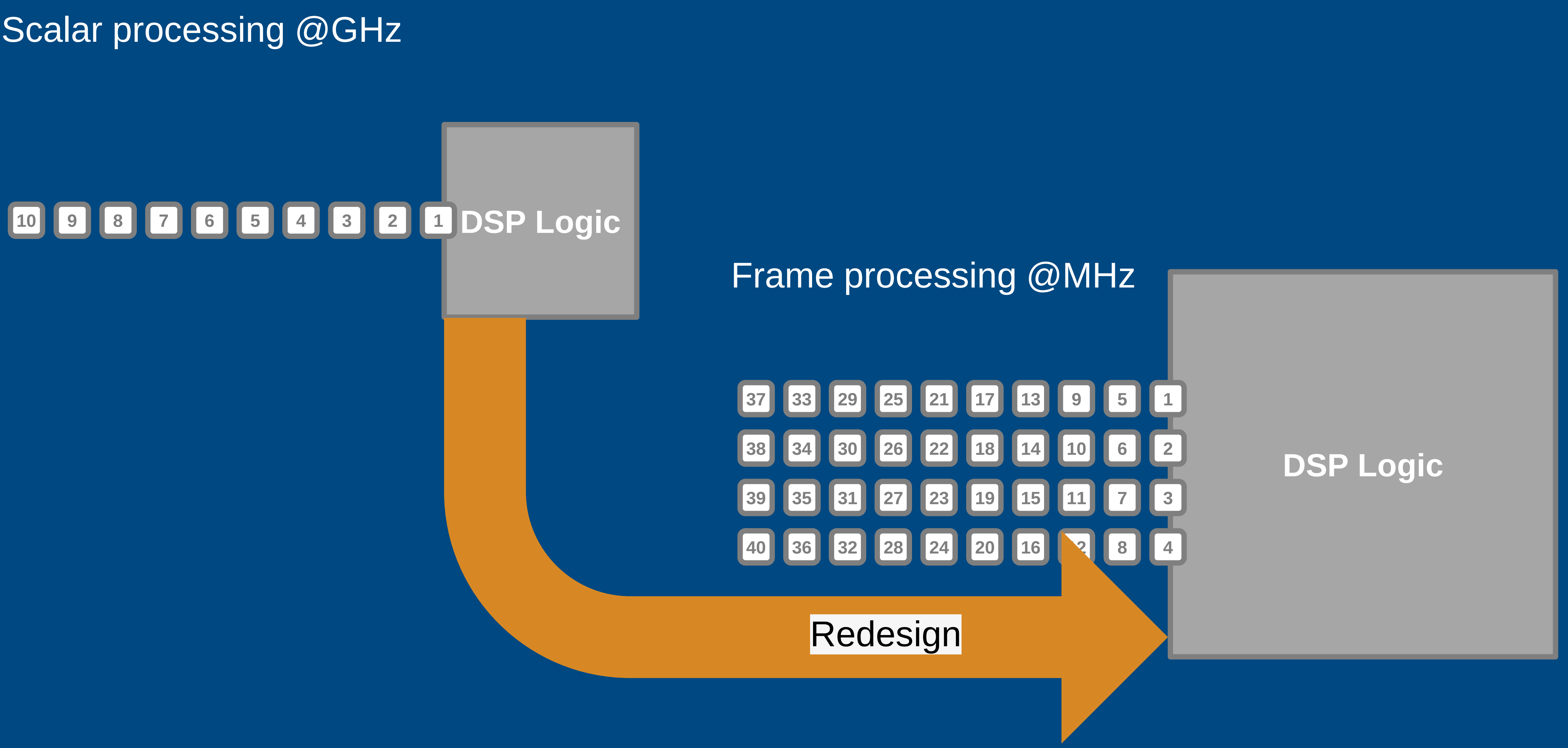
Scalar signal processing at GSPS throughput require GHz clock.

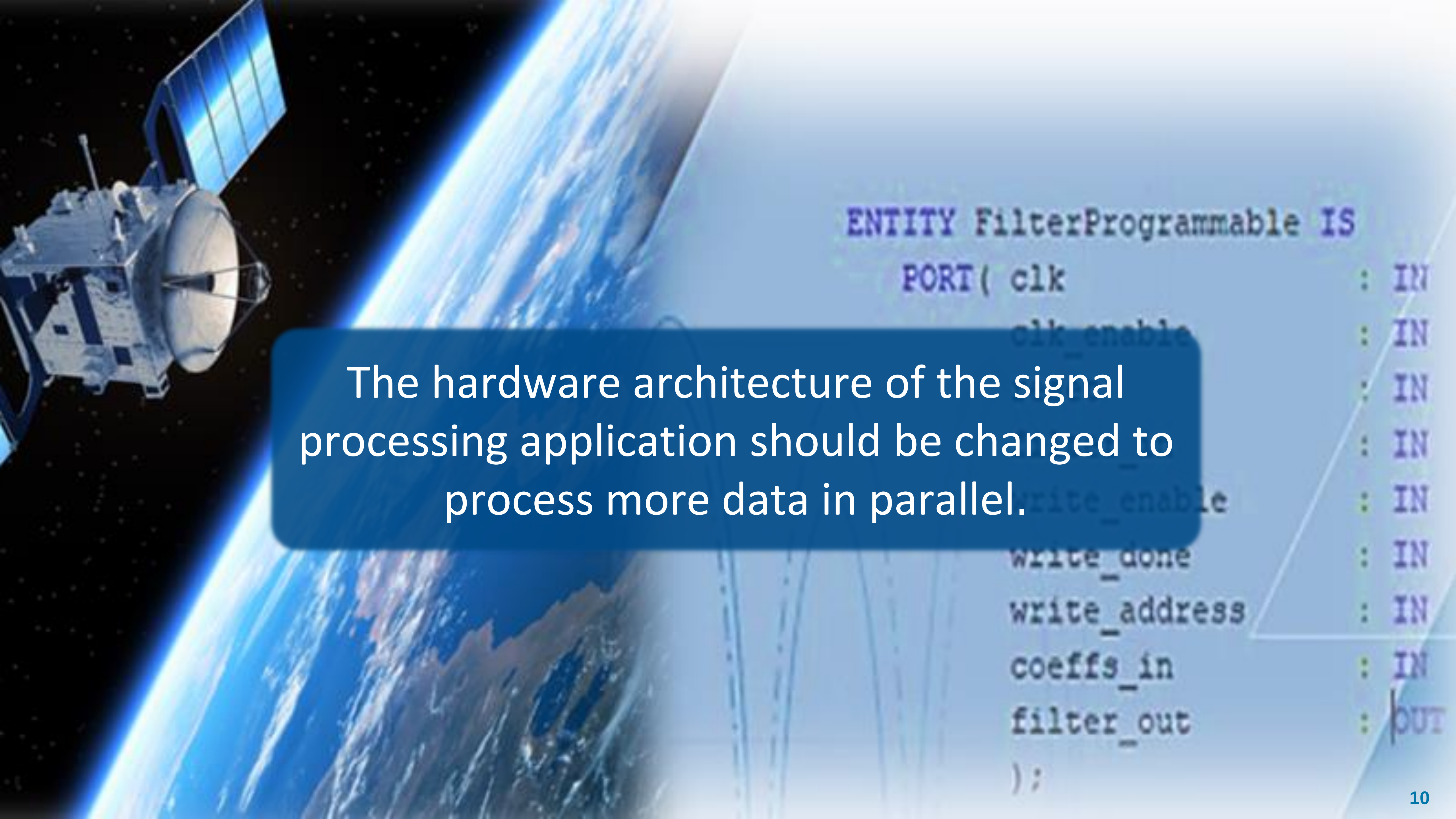
However, the max clock frequency in FPGAs and ASICs is limited and cannot catch up with the new demands.

In addition, higher clock frequency is not desirable due to the power consumption.

How to design GSPS throughput without requiring GHz clock?

Parallel (Frame) Processing





```
ENTITY FilterProgrammable IS
    PORT( clk                : IN
          clk_enable         : IN
                             : IN
                             : IN
          write_enable       : IN
          write_done         : IN
          write_address      : IN
          coeffs_in          : IN
          filter_out         : OUT
          );
```

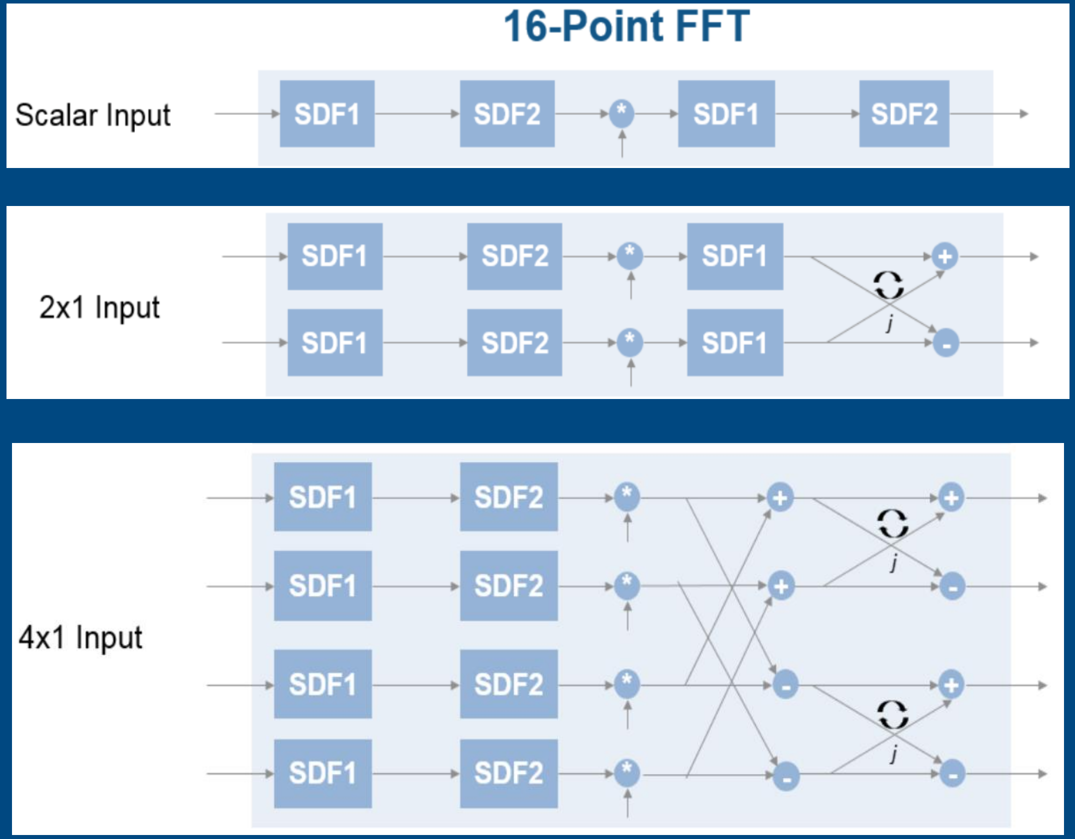
The hardware architecture of the signal processing application should be changed to process more data in parallel.

Frame-Based Design Challenges

- Frame based algorithm may not exist or well known, like any IIR filter, CIC, or Biquad.

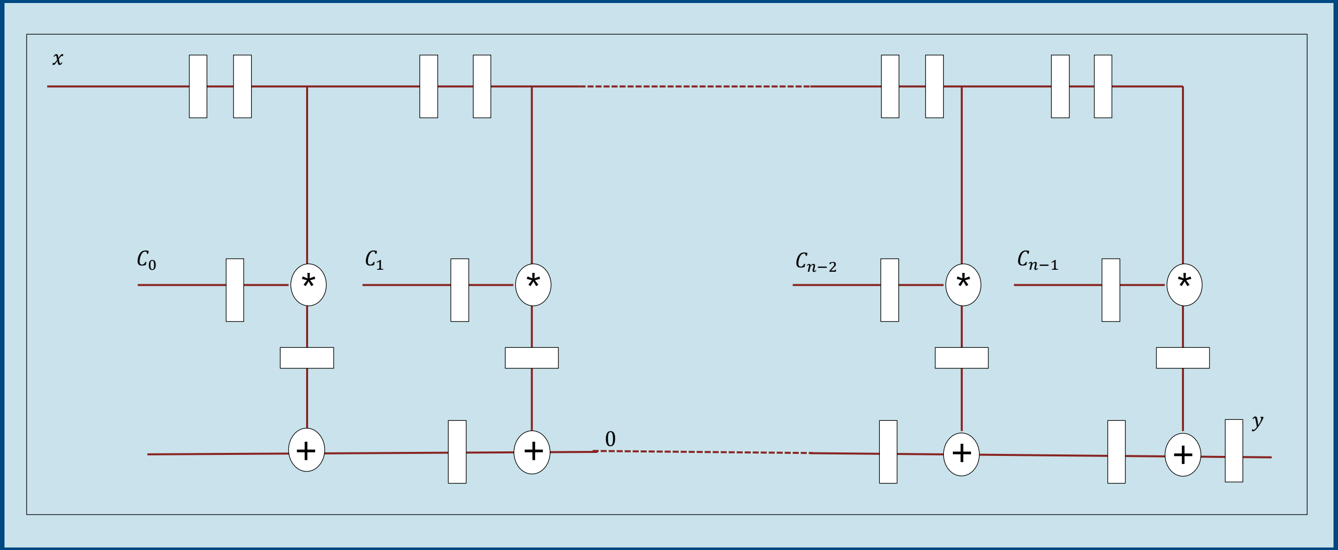
- Frame-based DSP, like any filter or FFT for different input frame size requiring extensive verification and rewriting the code and test bench.

- Design exploration for area and speed optimization is costly and the algorithm should be re-designed.

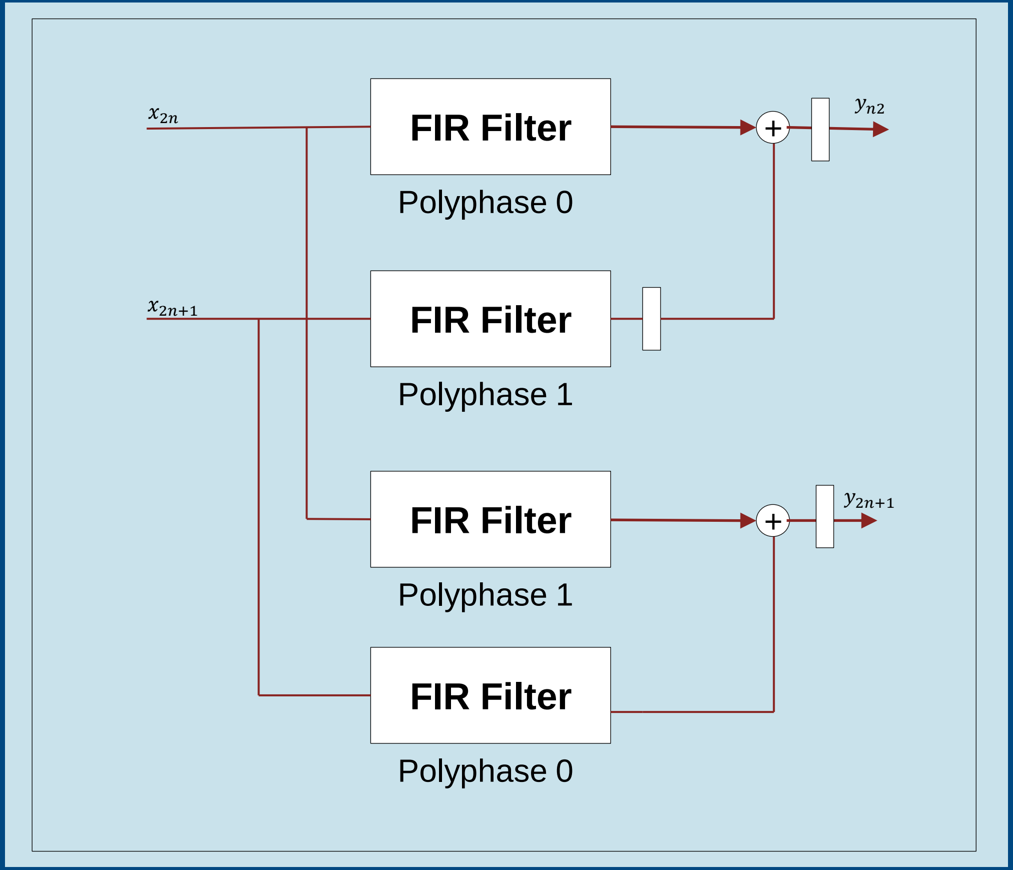


Scalar vs. Framed-Based FIR Architecture

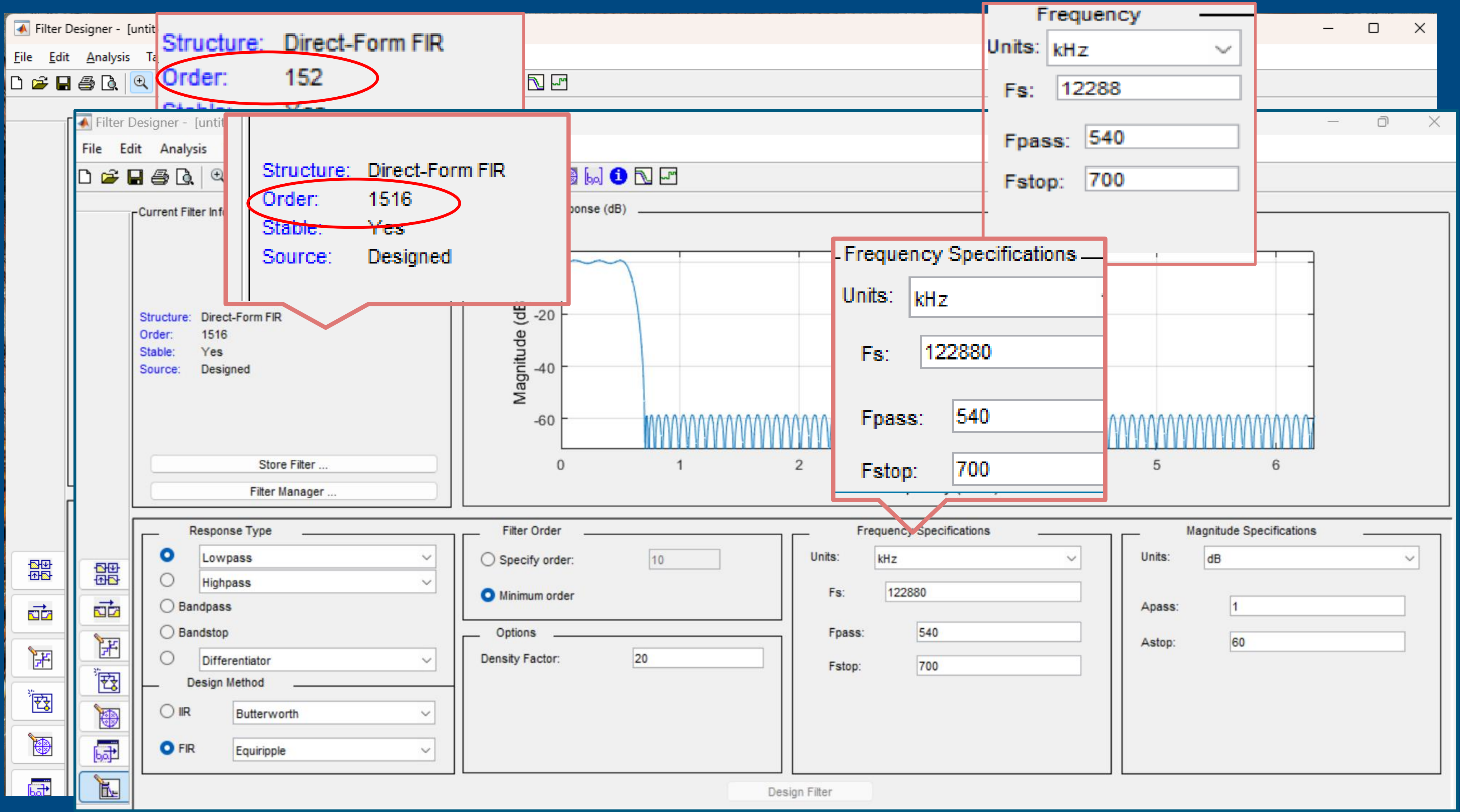
Scalar processing: Fully Parallel



Frame-Based: Polyphase



Designing High-Rate Filter Requires Significant DSP Resources



Filter Designer - [untitled]

Structure: Direct-Form FIR
Order: 152
Stable: Yes

Frequency
Units: kHz
Fs: 12288
Fpass: 540
Fstop: 700

Structure: Direct-Form FIR
Order: 1516
Stable: Yes
Source: Designed

Frequency Specifications
Units: kHz
Fs: 122880
Fpass: 540
Fstop: 700

Current Filter Information
Structure: Direct-Form FIR
Order: 1516
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: IIR Butterworth [✓ FIR Equiripple]

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

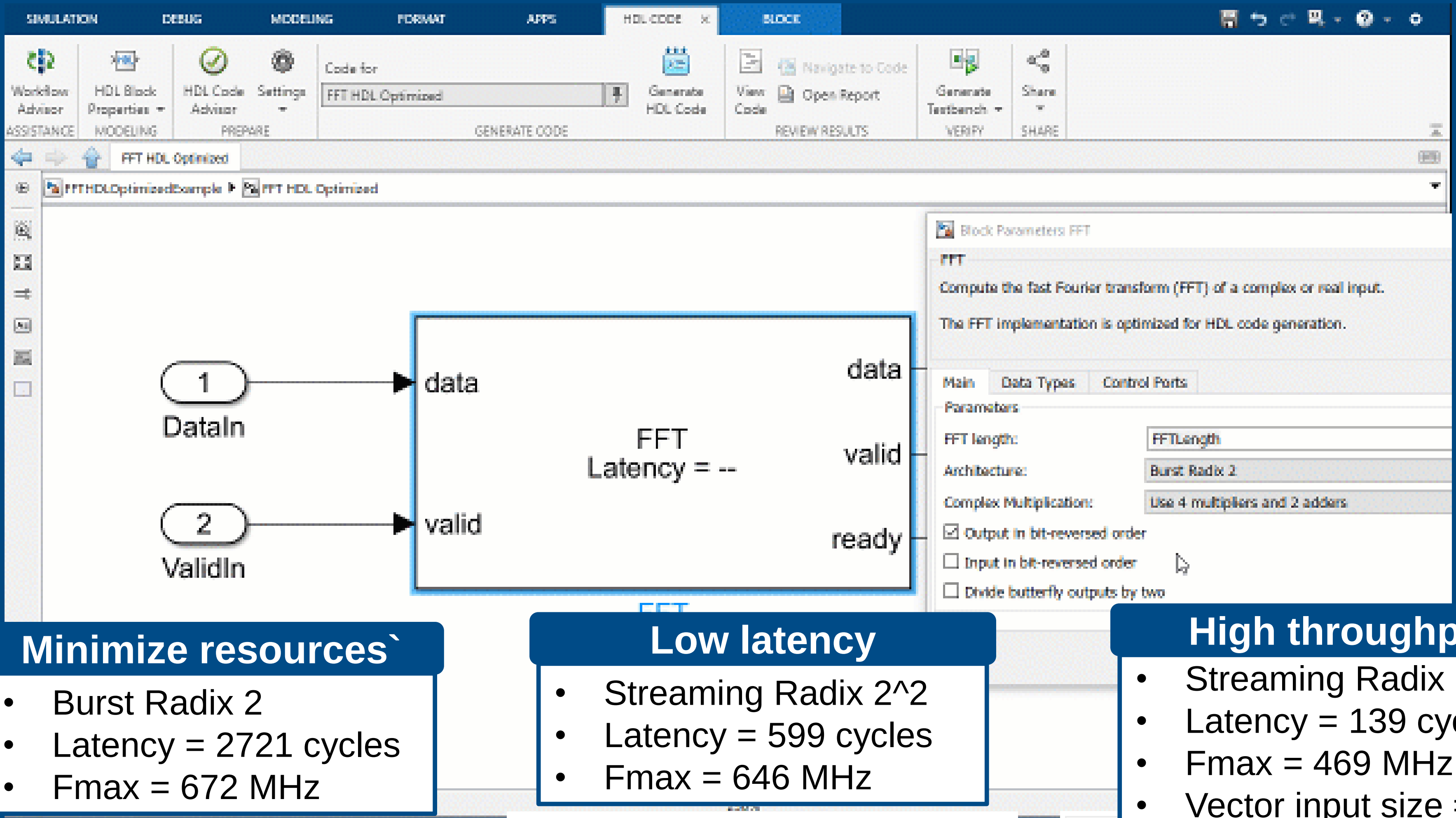
Options
Density Factor: 20

Frequency Specifications
Units: kHz
Fs: 122880
Fpass: 540
Fstop: 700

Magnitude Specifications
Units: dB
Apass: 1
Astop: 60

DSP HDL Toolbox implements DSP algorithms (FFT, IIR and FIR filters) which automatically select the proper hardware architecture based on the throughput.

FFT Implementation Exploration



Code for: FFT HDL Optimized

Block Parameters: FFT

FFT

Compute the fast Fourier transform (FFT) of a complex or real input.

The FFT implementation is optimized for HDL code generation.

Main | Data Types | Control Ports

Parameters

FFT length:	FFTLength
Architecture:	Burst Radix 2
Complex Multiplication:	Use 4 multipliers and 2 adders

☑ Output in bit-reversed order

☐ Input in bit-reversed order

☐ Divide butterfly outputs by two

Minimize resources`

- Burst Radix 2
- Latency = 2721 cycles
- Fmax = 672 MHz

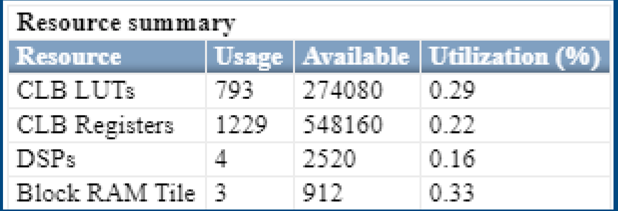
Low latency

- Streaming Radix 2^2
- Latency = 599 cycles
- Fmax = 646 MHz

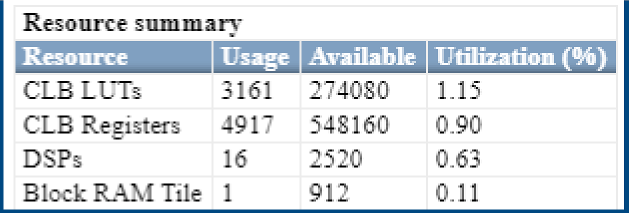
High throughput

- Streaming Radix 2^2
- Latency = 139 cycles
- Fmax = 469 MHz
- Vector input size = 8

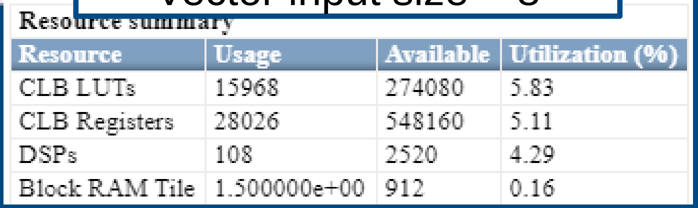
3.75 GSPS


Resource summary

Resource	Usage	Available	Utilization (%)
CLB LUTs	793	274080	0.29
CLB Registers	1229	548160	0.22
DSPs	4	2520	0.16
Block RAM Tile	3	912	0.33


Resource summary

Resource	Usage	Available	Utilization (%)
CLB LUTs	3161	274080	1.15
CLB Registers	4917	548160	0.90
DSPs	16	2520	0.63
Block RAM Tile	1	912	0.11


Resource summary

Resource	Usage	Available	Utilization (%)
CLB LUTs	15968	274080	5.83
CLB Registers	28026	548160	5.11
DSPs	108	2520	4.29
Block RAM Tile	1.500000e+00	912	0.16

Implementation metrics are post-synthesis, targeting Xilinx Zynq® UltraScale+ speed grade -2

Frame-Based Digital Down Converter Example

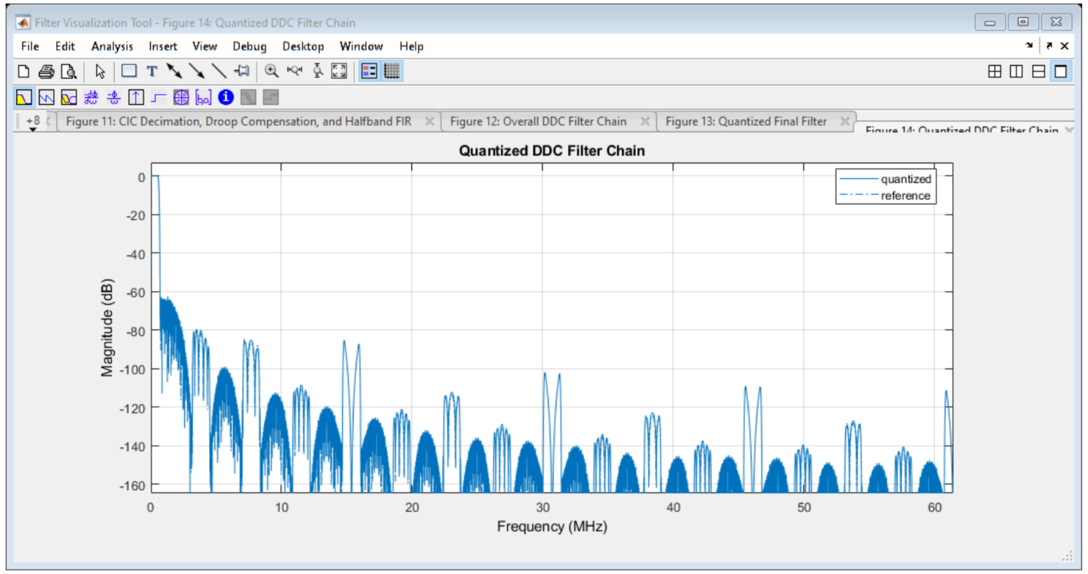
Showcasing ease of use

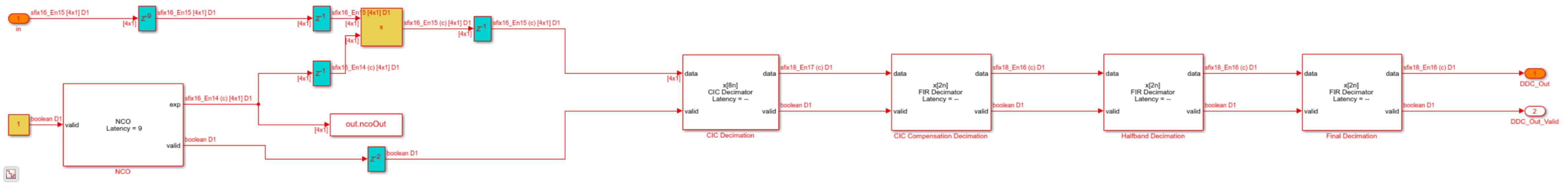
&

seamless algorithm adaption

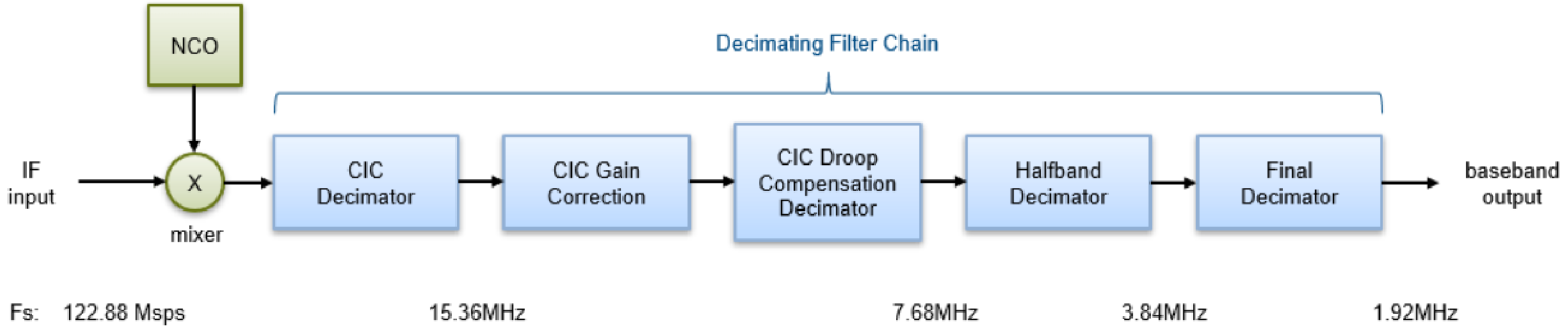
DSP HDL Toolbox – DDC Example

- This example shows how to design a digital down-converter (DDC) for radio communication applications such as LTE and generate HDL code.
- This is an example to show how it is easy to change the throughput and explore the functionality and hardware resources for a DDC filter chain.

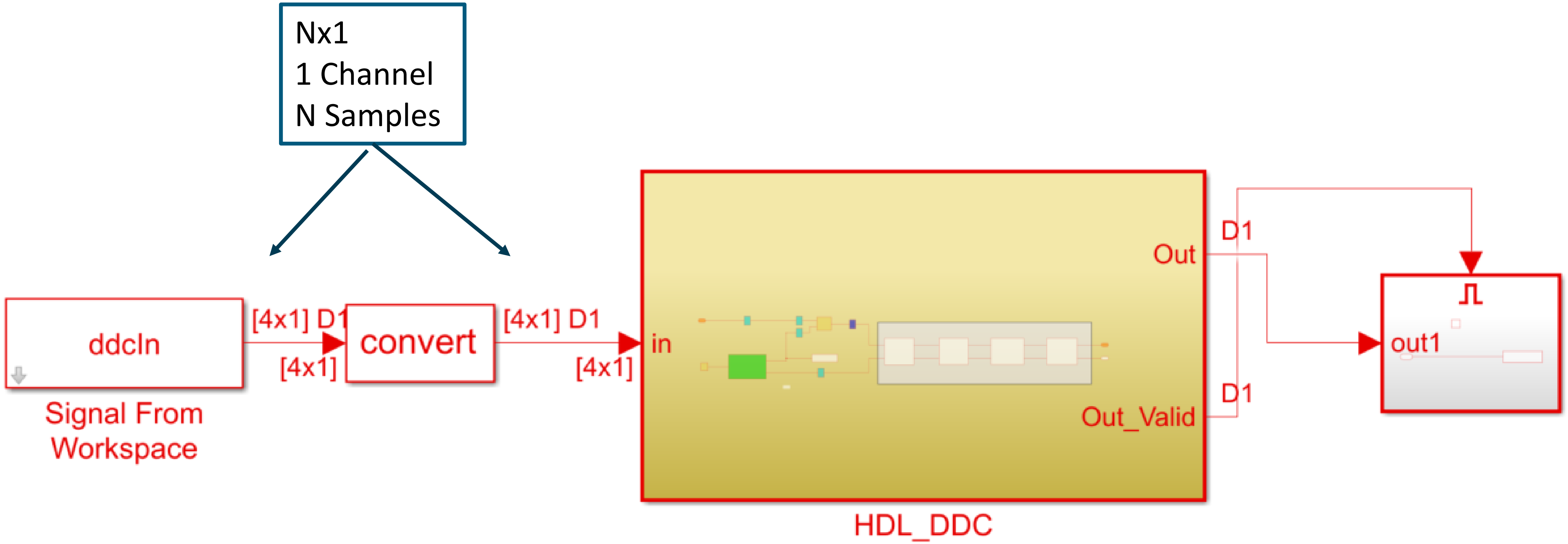


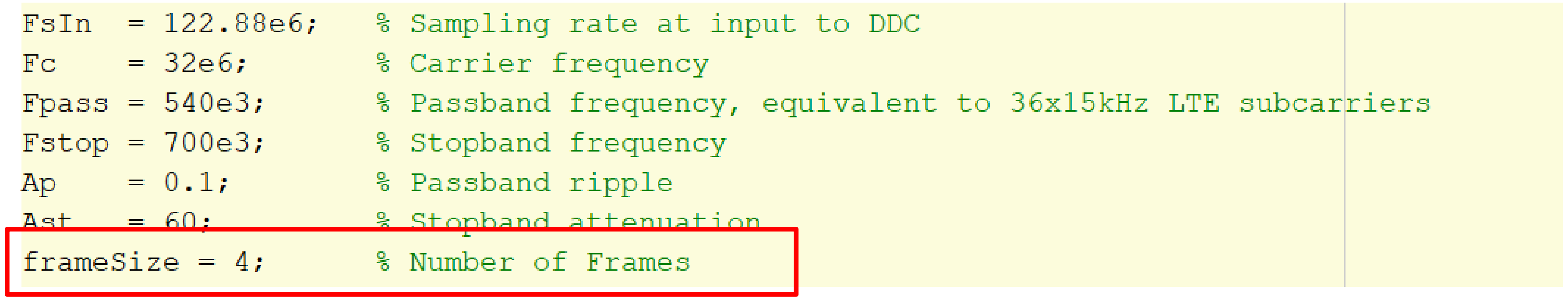


Digital Down Converter Structure



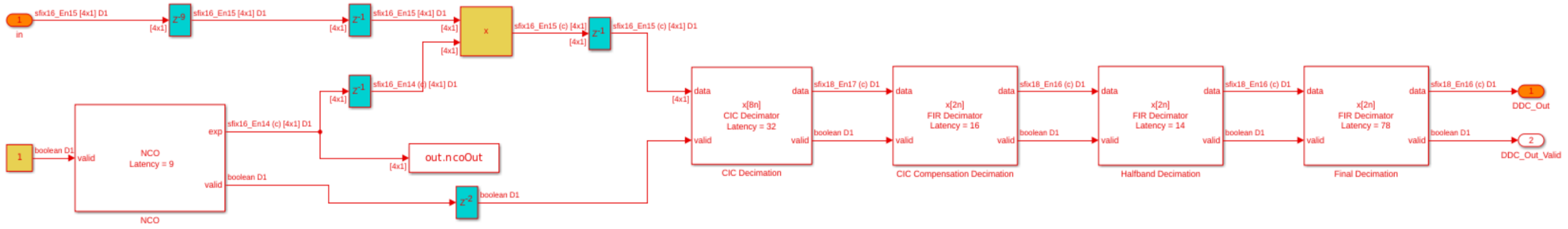
HDL implementation of DDC with Frame Processing





```
FsIn  = 122.88e6;    % Sampling rate at input to DDC
Fc    = 32e6;        % Carrier frequency
Fpass = 540e3;       % Passband frequency, equivalent to 36x15kHz LTE subcarriers
Fstop = 700e3;       % Stopband frequency
Ap    = 0.1;         % Passband ripple
Ast   = 60;          % Stopband attenuation
frameSize = 4;       % Number of Frames
```

Frame-Based DDC example

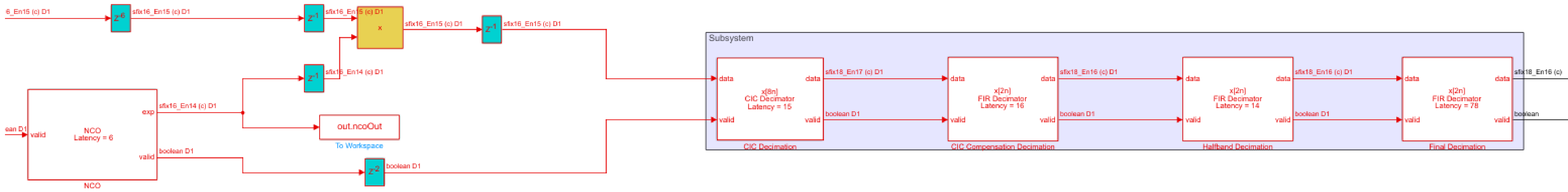


```
FsIn = 122.88e6;    % Sampling rate of DDC input
FsOut = 1.92e6;     % Sampling rate of DDC output
Fc = 32e6;          % Carrier frequency
Fpass = 540e3;      % Passband frequency, equivalent to 36x15kHz LTE subcarriers
Fstop = 700e3;      % Stopband frequency
Ap = 0.1;           % Passband ripple
Ast = 60;           % Stopband attenuation
```

```
FrameSize = 4;
```

Digital-Down-Converter-for-LTE

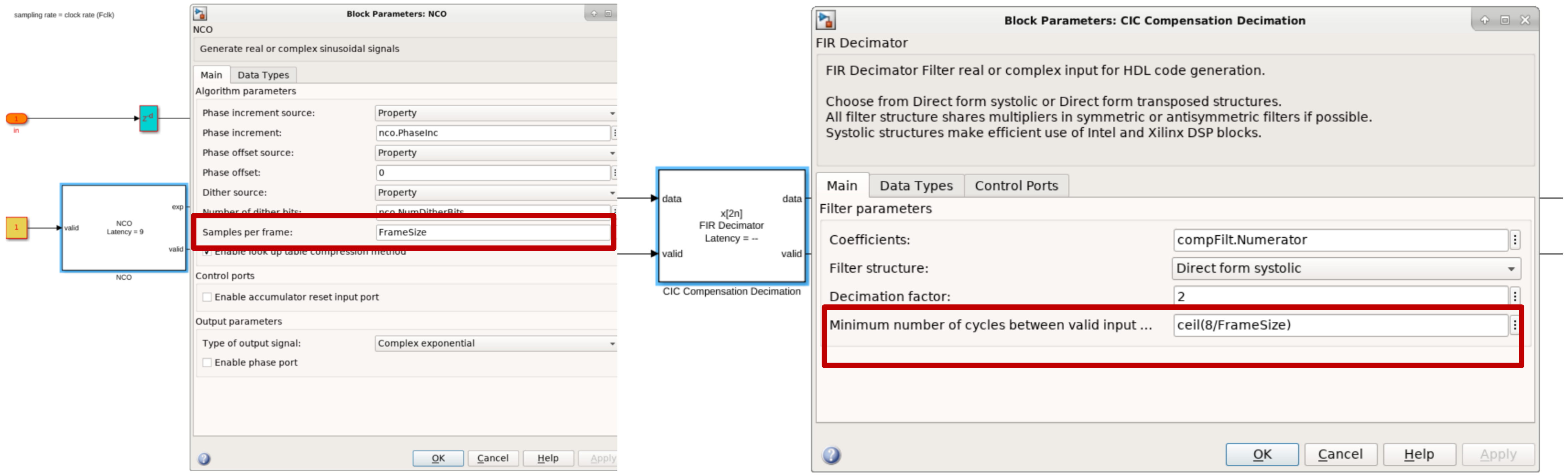
Scalar DDC



```
FsIn = 122.88e6;     % Sampling rate of DDC input
FsOut = 1.92e6;      % Sampling rate of DDC output
Fc = 32e6;           % Carrier frequency
Fpass = 540e3;       % Passband frequency, equivalent to 36x15kHz LTE subcarriers
Fstop = 700e3;       % Stopband frequency
Ap = 0.1;            % Passband ripple
Ast = 60;            % Stopband attenuation
```

```
FrameSize = 1;
```

Use Frame-size to Parameterize the Model



Block Parameters: NCO

NCO

Generate real or complex sinusoidal signals

Main | Data Types

Algorithm parameters

Phase increment source:	Property
Phase increment:	nco.PhaseInc
Phase offset source:	Property
Phase offset:	0
Dither source:	Property
Number of dither bits:	nco.NumDitherBits
Samples per frame:	FrameSize

[✓] Enable look up table compression method

Control ports

[ ] Enable accumulator reset input port

Output parameters

Type of output signal:	Complex exponential

[ ] Enable phase port

OK | Cancel | Help | Apply

Block Parameters: CIC Compensation Decimation

FIR Decimator

FIR Decimator Filter real or complex input for HDL code generation.

Choose from Direct form systolic or Direct form transposed structures.
All filter structure shares multipliers in symmetric or antisymmetric filters if possible.
Systolic structures make efficient use of Intel and Xilinx DSP blocks.

Main | Data Types | Control Ports

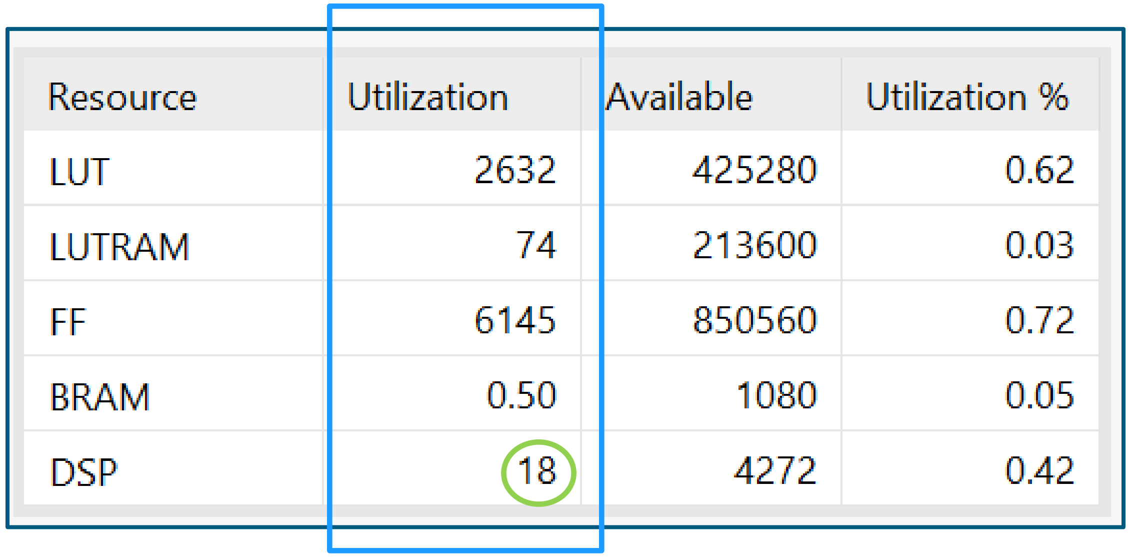
Filter parameters

Coefficients:	compFilt.Numerator
Filter structure:	Direct form systolic
Decimation factor:	2
Minimum number of cycles between valid input ...	ceil(8/FrameSize)

OK | Cancel | Help | Apply

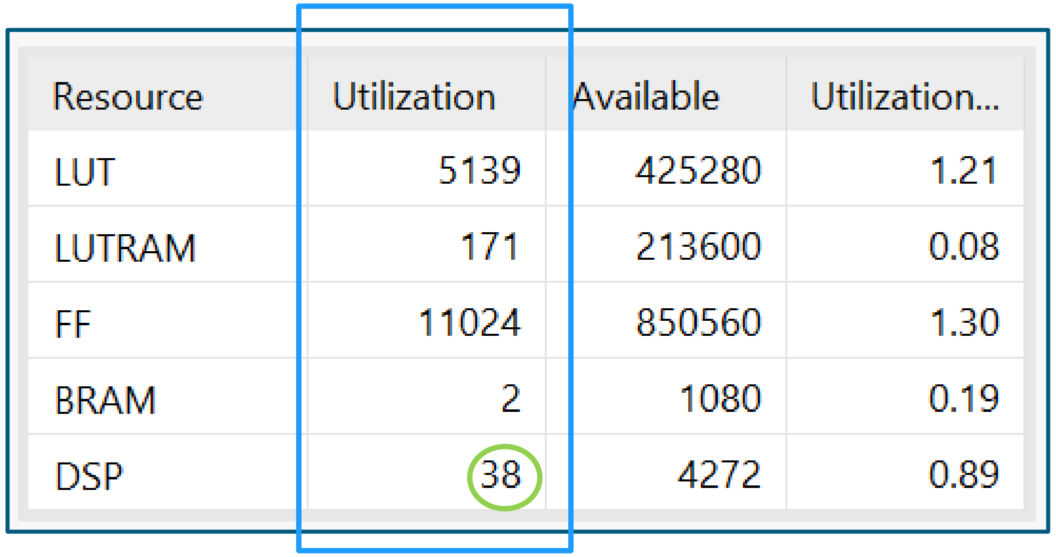
Resource utilization

Scalar processing



Resource	Utilization	Available	Utilization %
LUT	2632	425280	0.62
LUTRAM	74	213600	0.03
FF	6145	850560	0.72
BRAM	0.50	1080	0.05
DSP	18	4272	0.42

Frame processing



Resource	Utilization	Available	Utilization...
LUT	5139	425280	1.21
LUTRAM	171	213600	0.08
FF	11024	850560	1.30
BRAM	2	1080	0.19
DSP	38	4272	0.89

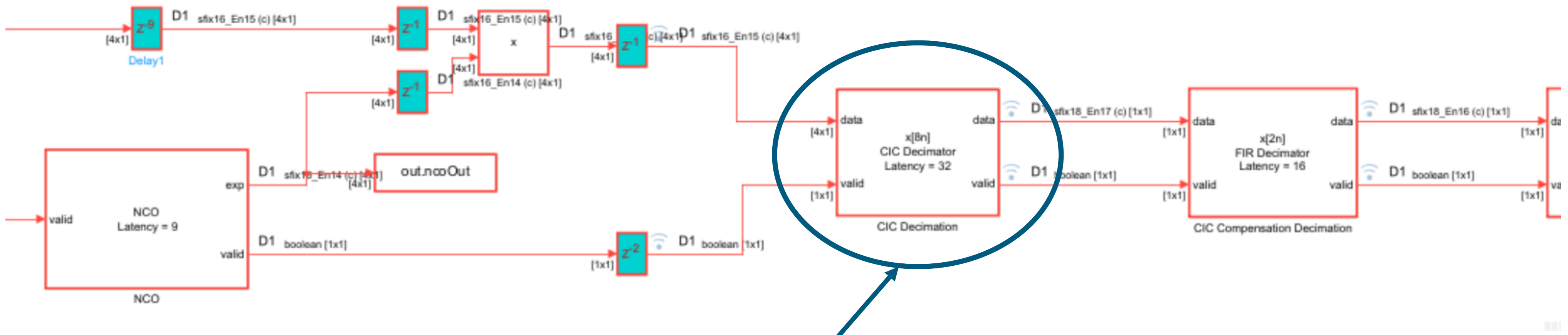
The resources are increased by factor of almost 2
While the input frame size increased by 4

Performance of the Frame Processing DDC
Zynq UltraScale+ RFSoC

Processing	Max. Clock Frequency MHz	Throughput Sample/Second Msps
Scalar	412	412
Frame	408	408X4 = 1632

Achieved 1.6 Giga Sample Per Second

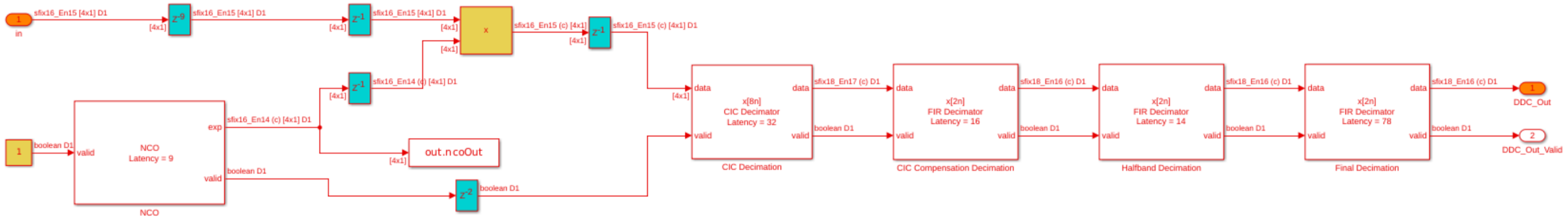
Reduce the Excessive Number of Multipliers



In a DDC with frame input, optimizing the first down-sample filter is essential to minimize multiplier usage.
Frame based CIC Decimator provides a simple solution to bring down the sampling rate significantly without using a lot of resource or any multiplier.

- **DSP HDL offers an innovative approach for designing frame-based CIC or IIR filter chains, optimized for GSPS sampling frequencies.**

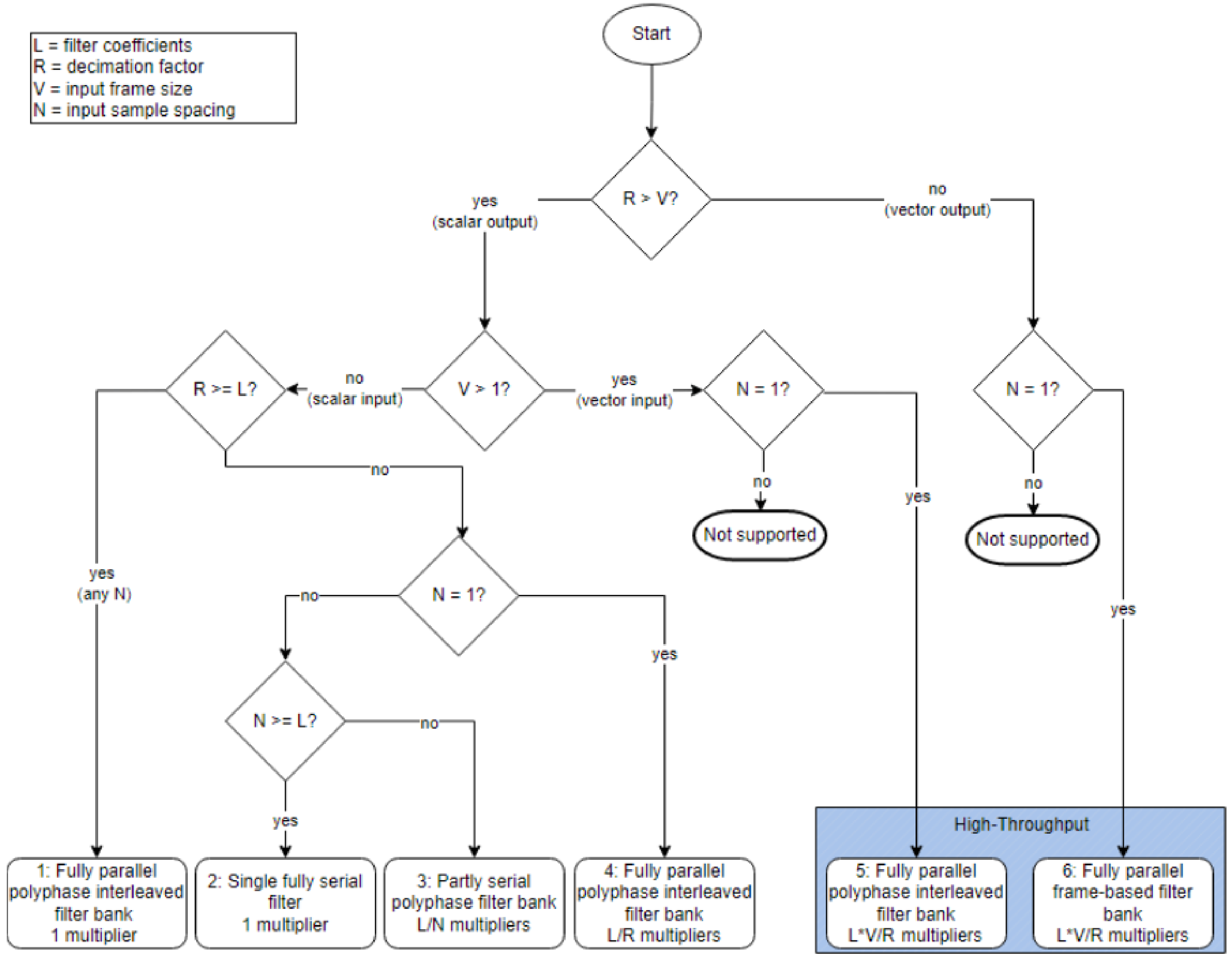
Algorithm Adaptation based on Frame Size



Frame size	CIC input		CIC Compensation input		Half band input		Final filter input	
	Processing	Sharing	Processing	Sharing	Processing	Sharing	Processing	Sharing
1	Scalar in-Scalar out	1	Scalar in-Scalar out	8	Scalar in-Scalar out	16	Scalar in-Scalar out	32
4	Frame in-Scalar out	1	Scalar in-Scalar out	2	Scalar in-Scalar out	4	Scalar in-Scalar out	8
16	Frame in- Frame out	1	Frame in-Scalar out	1	Scalar in-Scalar out	1	Scalar in-Scalar out	2
32	Frame in-Frame out	1	Frame in-Frame out	1	Frame in-Scalar out	1	Scalar in-Scalar out	1

FIR Decimator Optimization Technique

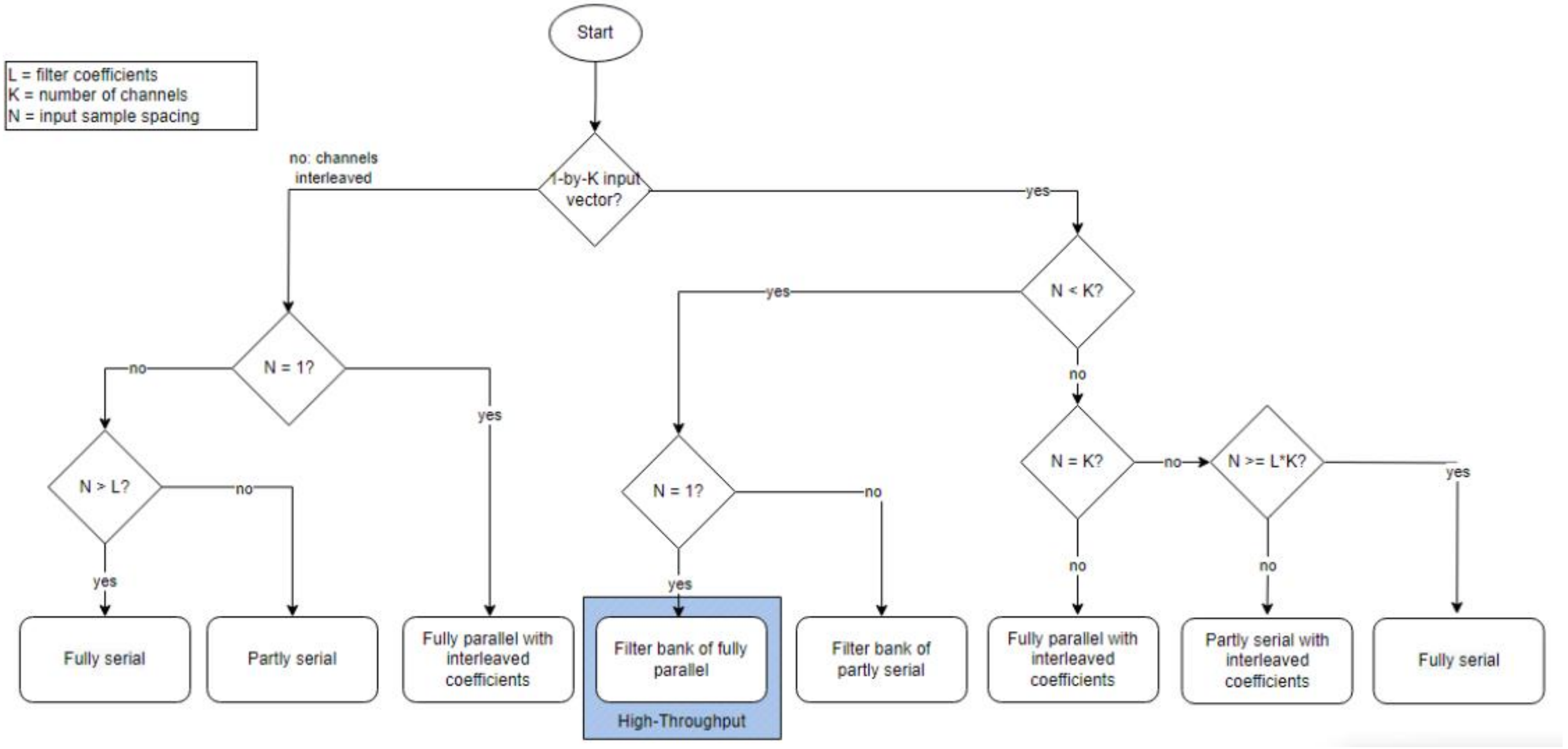
For Automatic architecture selection



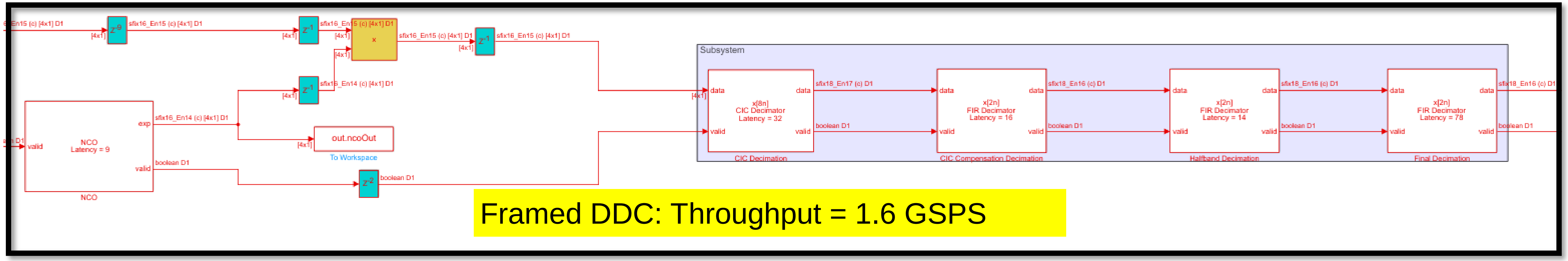
L = filter coefficients
R = decimation factor
V = input frame size
N = input sample spacing
Start
R > V?
yes (scalar output)
no (vector output)
V > 1?
no (scalar input)
yes (vector input)
R >= L?
N = 1?
N = 1?
no
Not supported
yes
N = 1?
no
Not supported
yes
yes (any N)
no
N = 1?
no
yes
N >= L?
no
yes
1: Fully parallel polyphase interleaved filter bank 1 multiplier
2: Single fully serial filter 1 multiplier
3: Partly serial polyphase filter bank L/N multipliers
4: Fully parallel polyphase interleaved filter bank L/R multipliers
High-Throughput
5: Fully parallel polyphase interleaved filter bank L*V/R multipliers
6: Fully parallel frame-based filter bank L*V/R multipliers

Link to the documentation

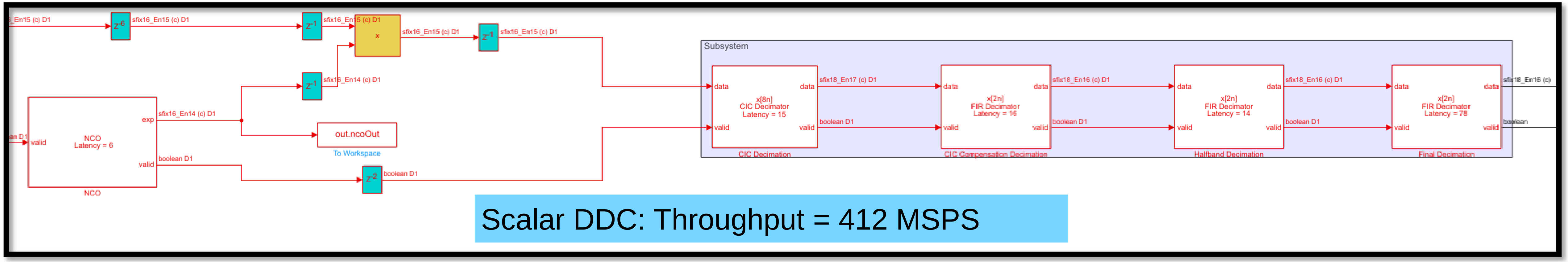
Discrete FIR Filter – Automatic Architecture Selection



DSP Engineers can use the **same algorithm for both scalar and frame processing** and easily explore area, speed and throughput trade-offs targeting FPGAs and SoCs.



Framed DDC: Throughput = 1.6 GSPS



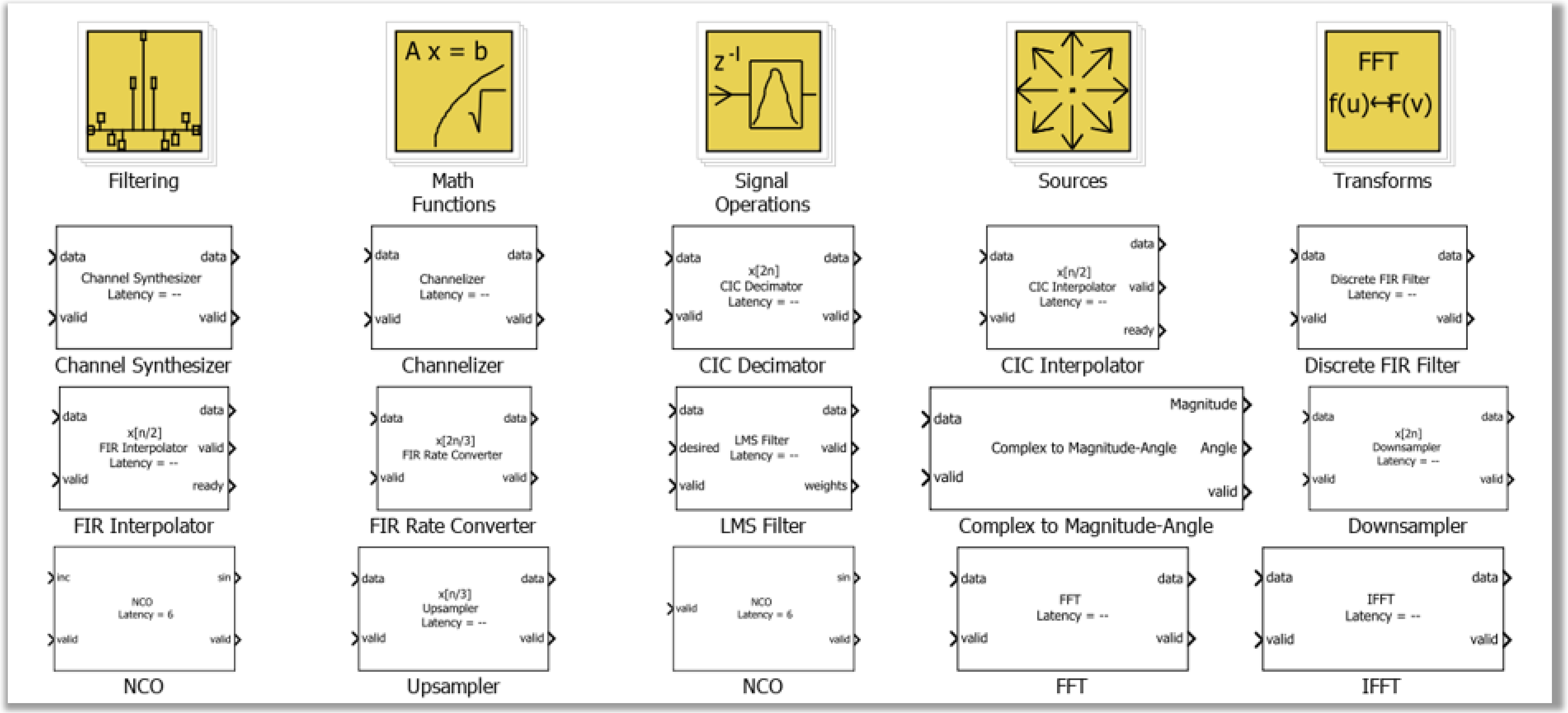
Scalar DDC: Throughput = 412 MSPS

DSP HDL IPs

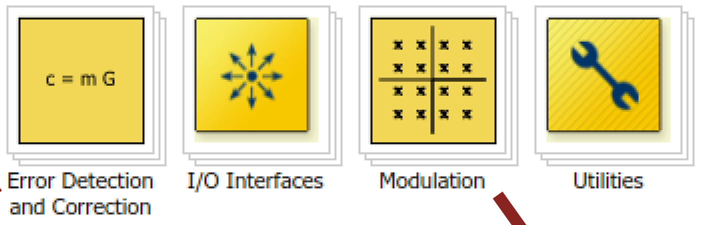
- ✓ Provide hardware-optimized algorithms that model streaming data interfaces, hardware latency, and control signals in MATLAB and Simulink®.
- ✓ Can process a number of samples in parallel to achieve high throughput such as gigasample-per-second (GSPS) rates.
- ✓ You can change the block parameters to explore different hardware implementations.
- ✓ These blocks support HDL code generation and deployment to FPGAs with HDL Coder™

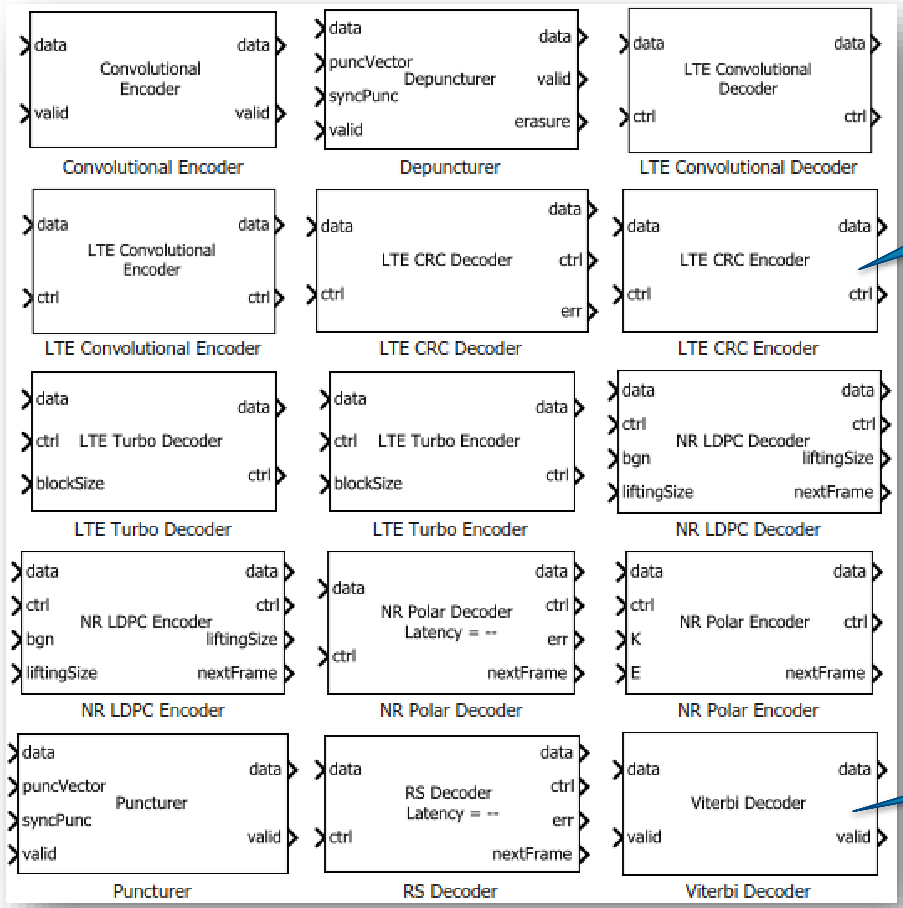
Simulink Simulation with Hardware Latency

DSP HDL Toolbox
IP Blocks



Wireless HDL Toolbox

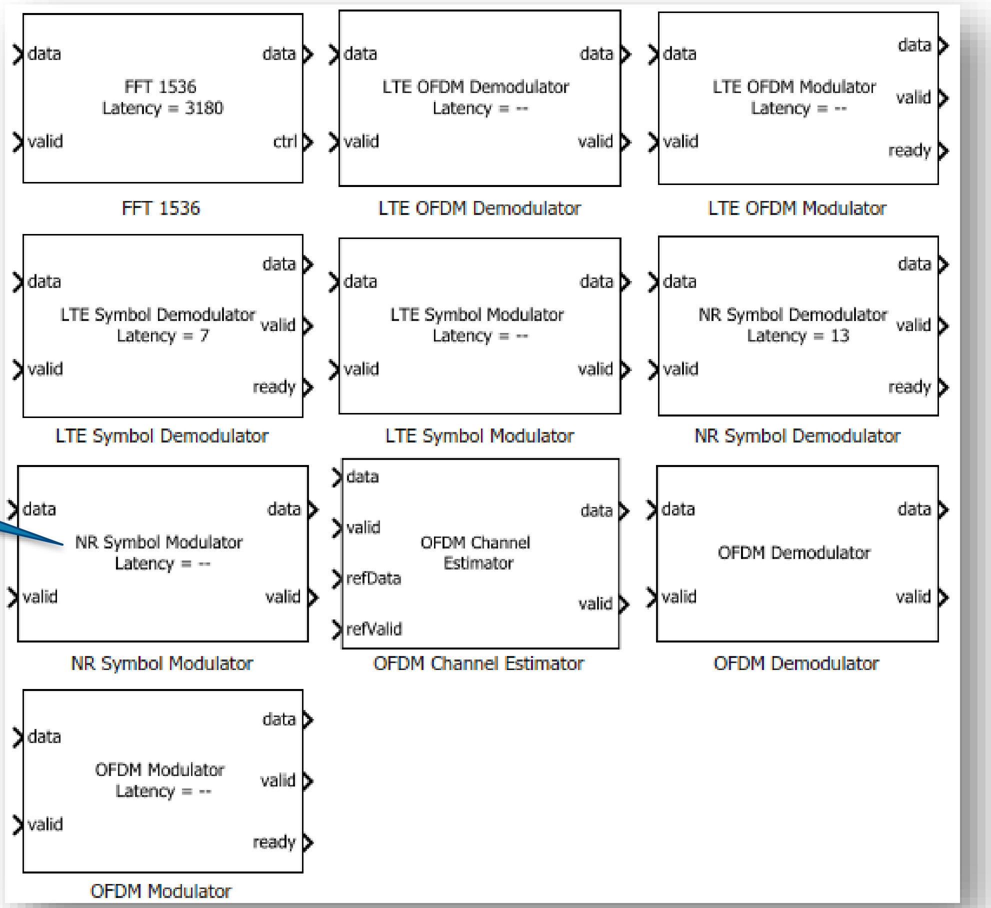




LTE

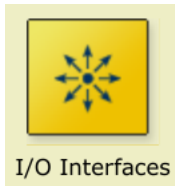
5G

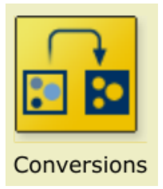
Generalized

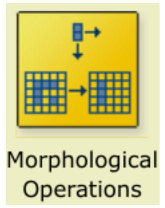


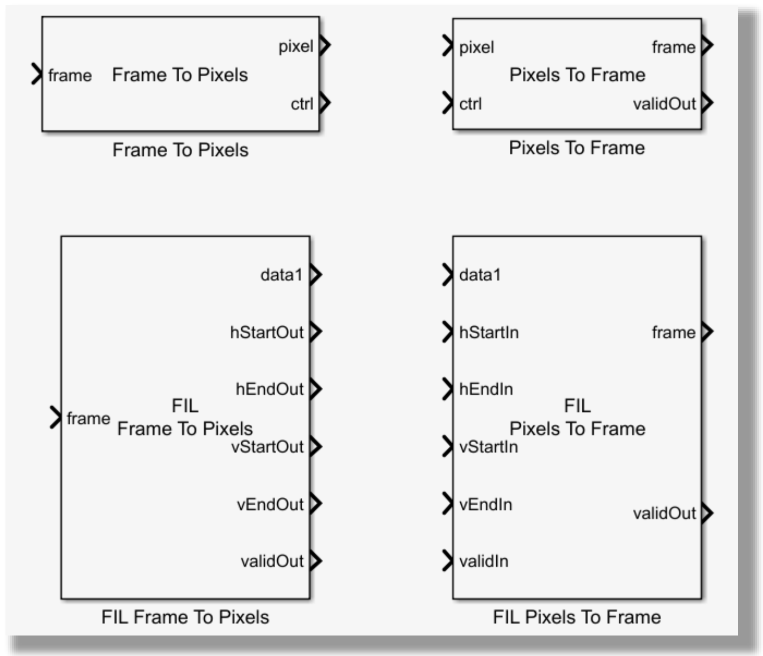
Vision HDL Toolbox

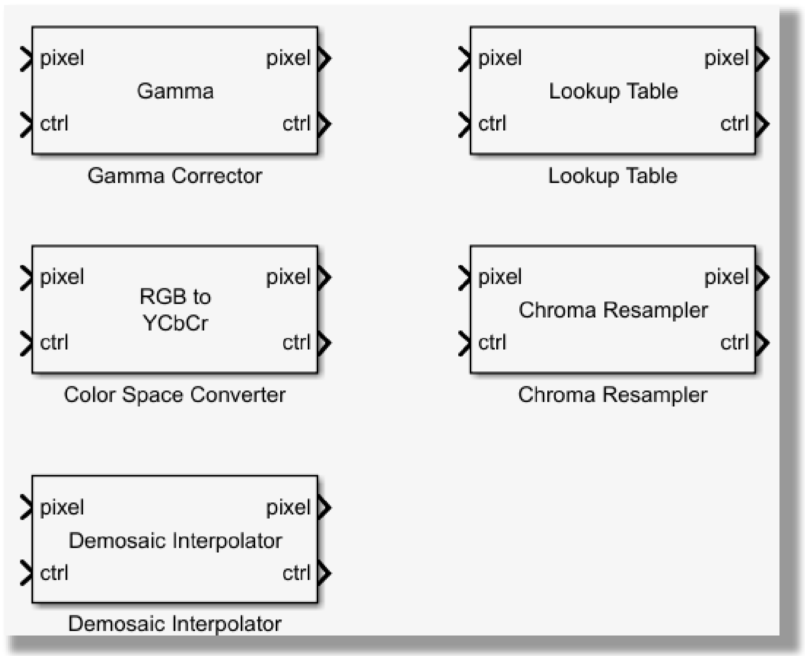
- Simulate with latency but it does not show it on the mask.
- The vision blocks use a control bus signal, which includes Valid in and Valid out.

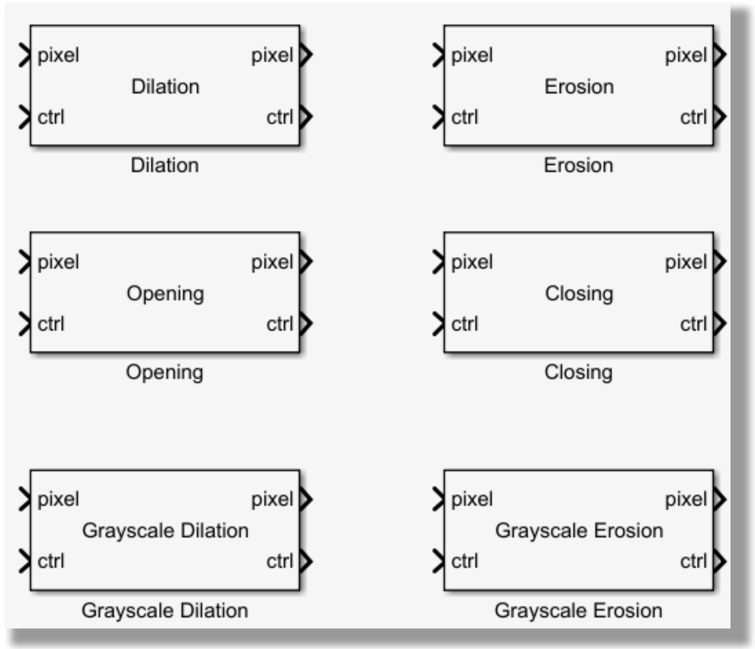




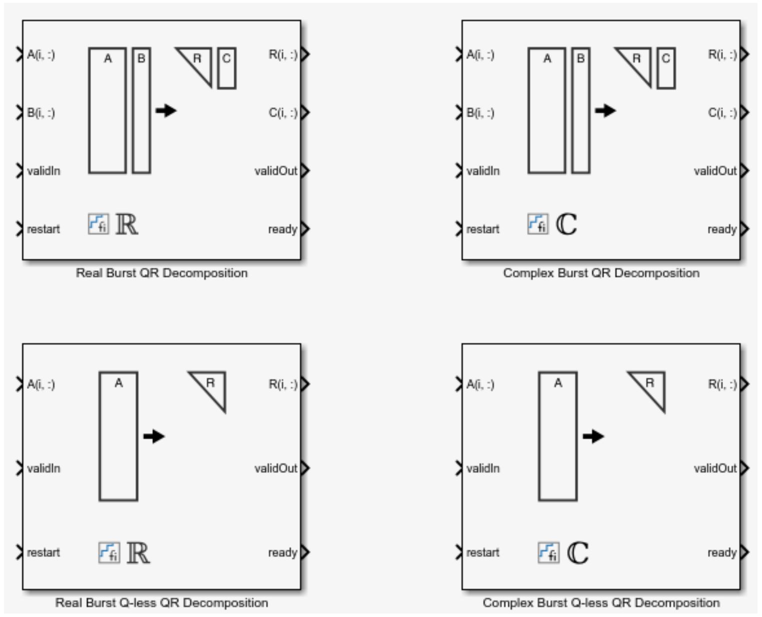




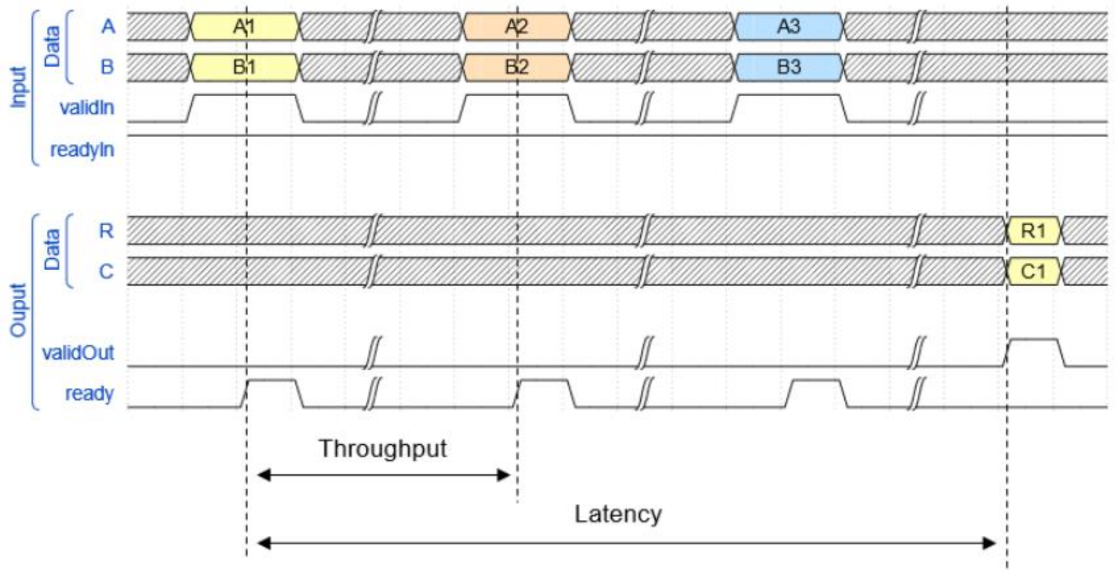




Fixed-Point Designer HDL Support Library



A(i, :)
B(i, :)
validIn
restart
R(i, :)
C(i, :)
validOut
ready
Real Burst QR Decomposition
Complex Burst QR Decomposition
Real Burst Q-less QR Decomposition
Complex Burst Q-less QR Decomposition



Implementation	Throughput	Latency	Area
Systolic	C	$O(n)$	$O(mn^2)$
Partial-Systolic	C	$O(m)$	$O(n^2)$
Partial-Systolic with Forgetting Factor	C	$O(n)$	$O(n^2)$
Burst	$O(n)$	$O(mn)$	$O(n)$

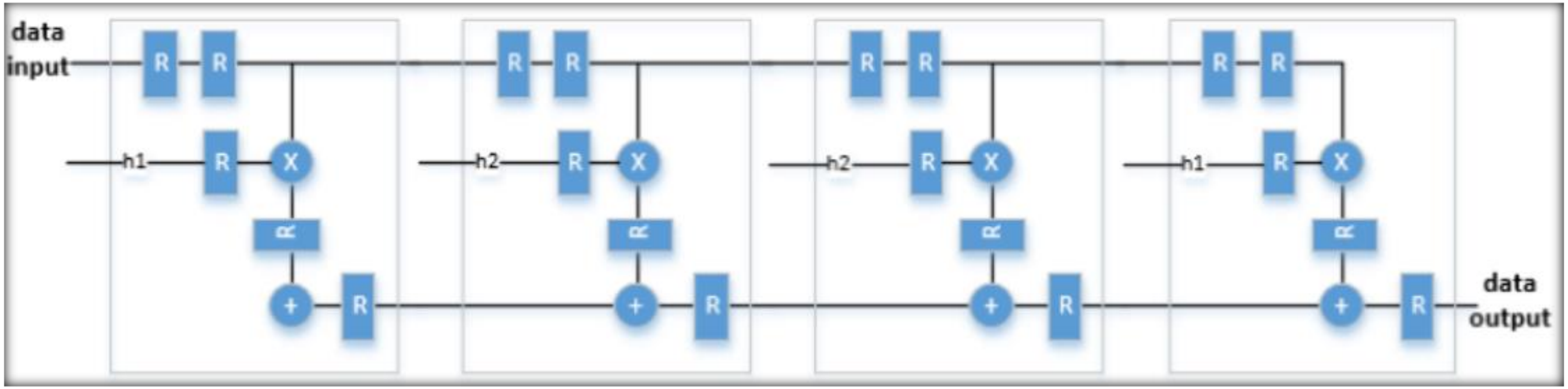
Choose a Block for HDL-Optimized Fixed-Point Matrix Operation

Cycle-Accurate Hardware Latency Simulation

The blocks model architectural latency including Pipeline registers and resource sharing.

[FIR Filter Architectures for FPGAs and ASICs]()

Fully Parallel Systolic Architecture

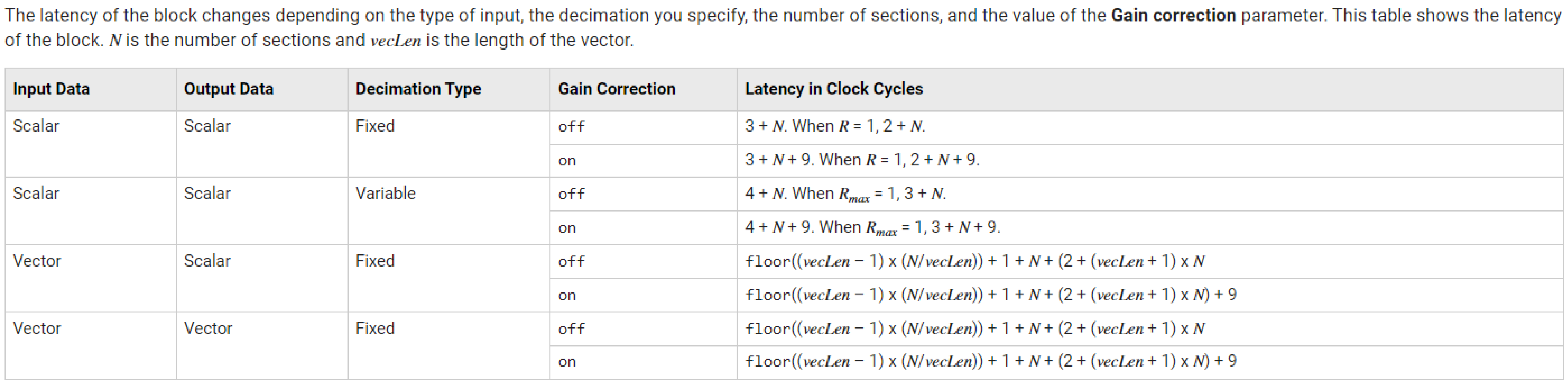


Cycle-Accurate Hardware Latency Simulation

The latency between valid input data and the corresponding valid output data depends on block parameters

- Block architecture,
- Input frame-size
- Spacing between validIn samples
- Filter/FFT length

CIC Decimator Latency



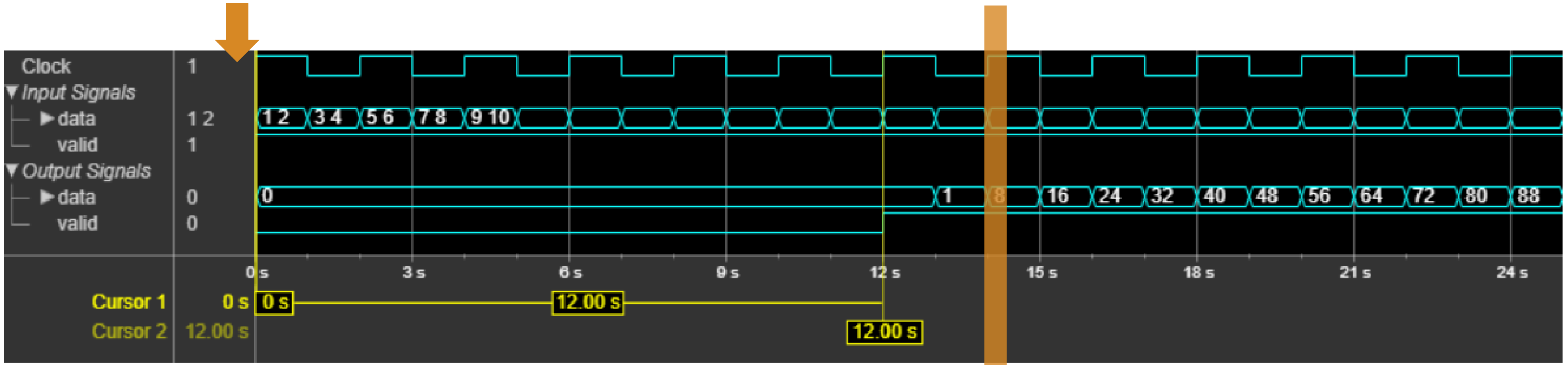
The latency of the block changes depending on the type of input, the decimation you specify, the number of sections, and the value of the **Gain correction** parameter. This table shows the latency of the block. N is the number of sections and $vecLen$ is the length of the vector.

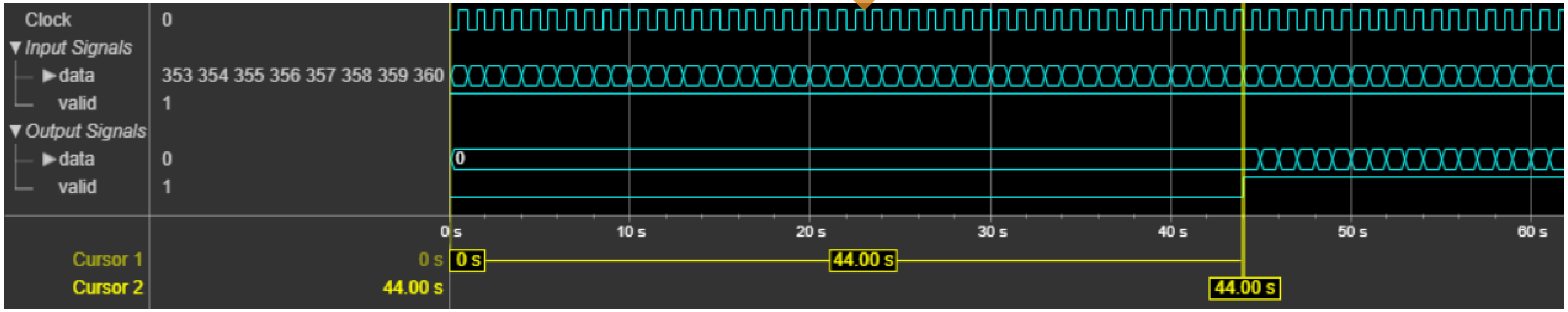
Input Data	Output Data	Decimation Type	Gain Correction	Latency in Clock Cycles
Scalar	Scalar	Fixed	`off`	$3 + N$. When $R = 1$, $2 + N$.
			`on`	$3 + N + 9$. When $R = 1$, $2 + N + 9$.
Scalar	Scalar	Variable	`off`	$4 + N$. When $R_{max} = 1$, $3 + N$.
			`on`	$4 + N + 9$. When $R_{max} = 1$, $3 + N + 9$.
Vector	Scalar	Fixed	`off`	`floor`$((vecLen - 1) \times (N/vecLen)) + 1 + N + (2 + (vecLen + 1) \times N$
			`on`	`floor`$((vecLen - 1) \times (N/vecLen)) + 1 + N + (2 + (vecLen + 1) \times N) + 9$
Vector	Vector	Fixed	`off`	`floor`$((vecLen - 1) \times (N/vecLen)) + 1 + N + (2 + (vecLen + 1) \times N$
			`on`	`floor`$((vecLen - 1) \times (N/vecLen)) + 1 + N + (2 + (vecLen + 1) \times N) + 9$

CIC Decimator Latency

Frame Size = 2,
Decimation factor: R= 2
Differential delay: M = 1
Number of sections N= 2

Frame Size = 8,
Decimation factor: R= 8
Differential delay: M = 1
Number of sections N= 3





Takeaway

DSP HDL Blocks:

- Support GSPS processing
- Seamless algorithm adaptation
- Simulate with hardware latency
- Facilitate design optimization for speed, area, and throughput

Next Steps

✓ Upcoming Training DSP for FPGAs

Contact Sales

Trial License

DSP for FPGA Training

This three-day course will review DSP fundamentals from the perspective of implementation within the FPGA fabric. Particular emphasis will be given to highlighting the cost, with respect to both resources and performance, associated with the implementation of various DSP techniques and algorithms.

Topics include:

- Introduction to FPGA hardware and technology for DSP applications
- DSP fixed-point arithmetic
- Signal flow graph techniques
- HDL code generation for FPGAs
- Fast Fourier Transform (FFT) Implementation
- Design and implementation of FIR, IIR and CIC filters
- CORDIC algorithm
- Design and implementation of adaptive algorithms such as LMS and QR algorithm
- Techniques for synchronization and digital communications timing recovery

https://www.mathworks.com/learn/training/dsp-for-fpgas.html