Accelerating FPGA Development

Harnessing the Power of HDL Coder and Collaborative Integration with NanoXplore

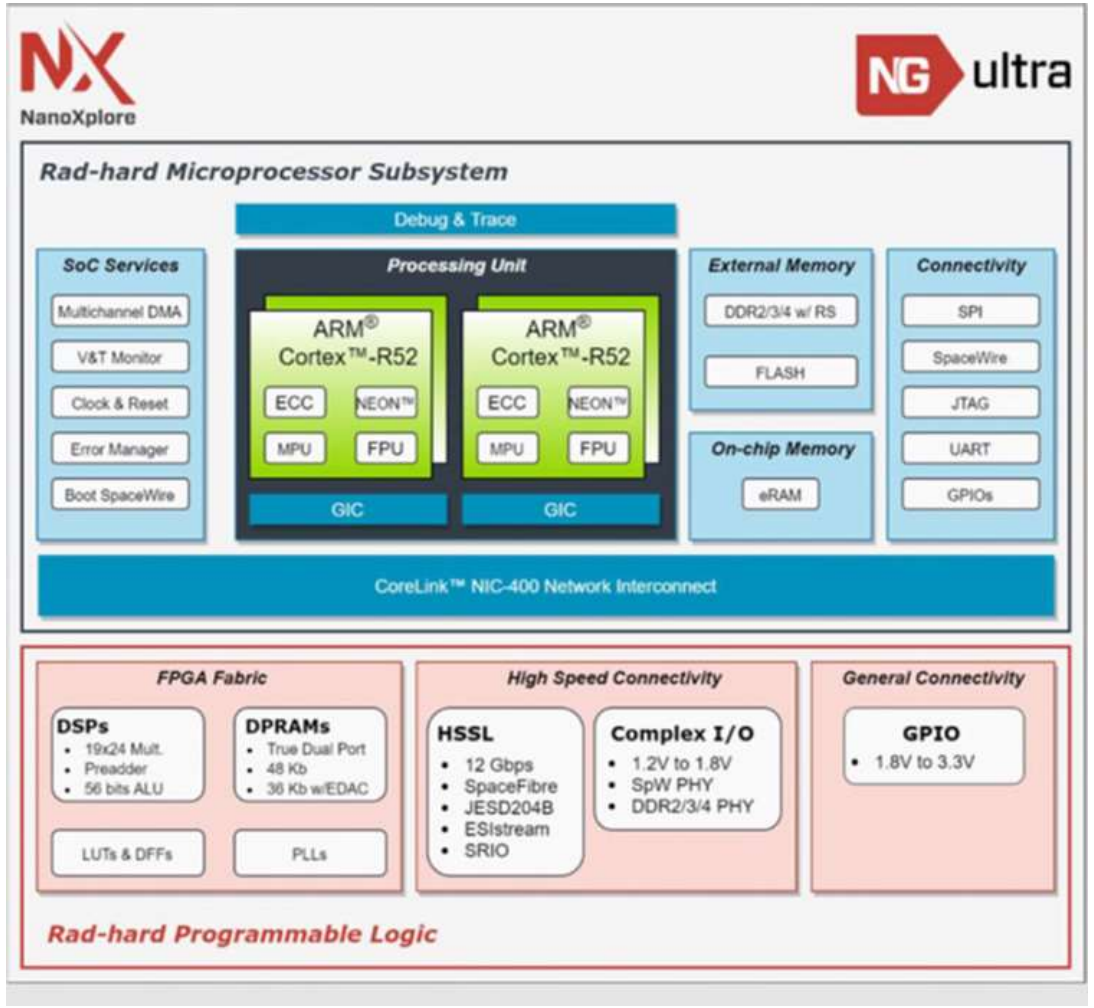
Yanitsa Stoyanova
Digital Design and Verification Engineer
Airbus Defence and Space

Stephan van Beek
European Technical Specialist SoC/FPGA Design Flows
MathWorks

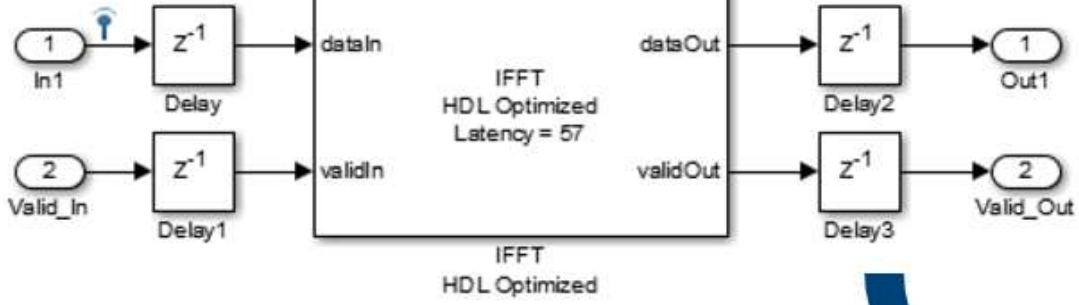
Project, feasibility studies

H2020 OPERA Project has received funding from the European Union's H2020 research and innovation program under grant agreement N°821969

Space Qualification and Validation of High Performance European Rad-Hard FPGA



Digital Signal Processing Algorithms Flow with HDL Coder




```
ENTITY IFFT IS
  PORT( clk                               :   IN    std_logic;
        reset                             :   IN    std_logic;
        clk_enable                        :   IN    std_logic;
        In1_re                            :   IN    std_logic_vector(15 DOWNTO 0);
        In1_im                            :   IN    std_logic_vector(15 DOWNTO 0);
        Valid_In                          :   IN    std_logic;
        ce_out                            :   OUT   std_logic;
        Out1_re                           :   OUT   std_logic_vector(15 DOWNTO 0);
        Out1_im                           :   OUT   std_logic_vector(15 DOWNTO 0);
        Valid_Out                         :   OUT   std_logic
        );
END IFFT;
```

HDL Coder and the HDL Workflow Advisor

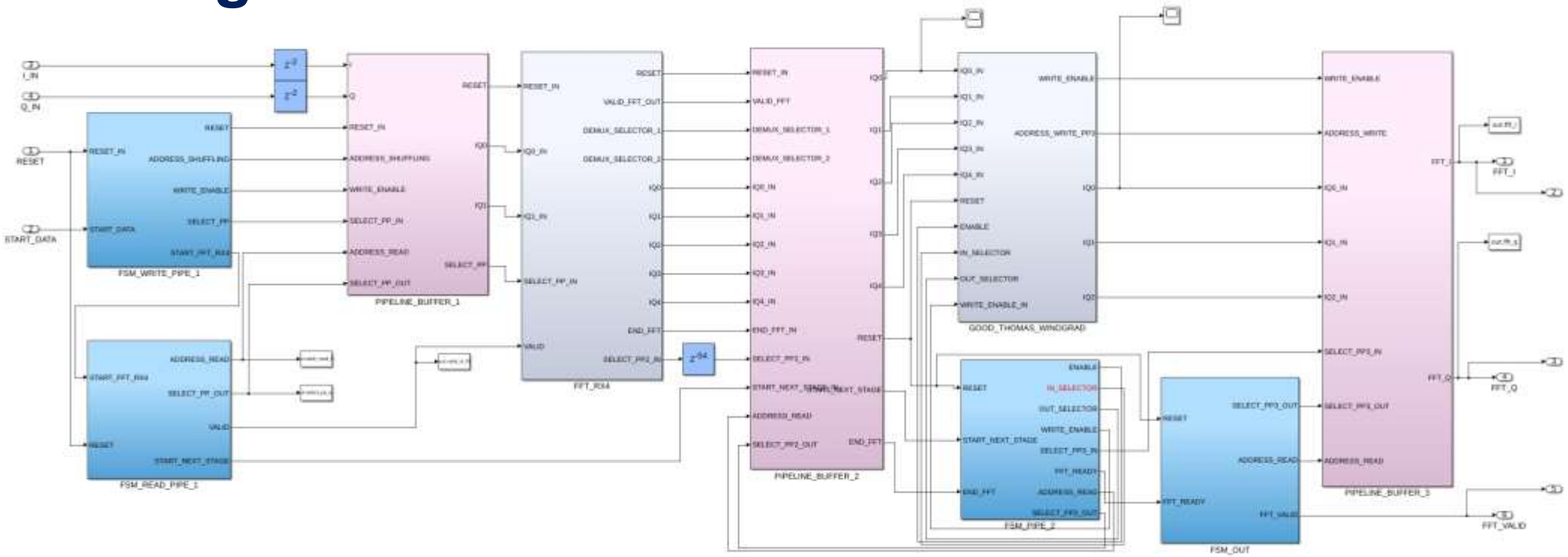
Workflow:

- User inputs: Simulink model
- Auto-Generated VHDL code
- Auto-Generated testbench plus stimuli vectors for RTL simulations

- Nanoxplore's Impulse Synthesis, Place and Route
- Netlist simulations

AIRBUS

FFT240 design model



Workflow:

- User inputs: Simulink model
- Auto-Generated VHDL code
- Auto-Generated testbench plus stimuli vectors for RTL simulations

- Nanoxplore's Impulse Synthesis, Place and Route
- Netlist simulations

AIRBUS

FFT240 design requirements

Required : 180 MHz

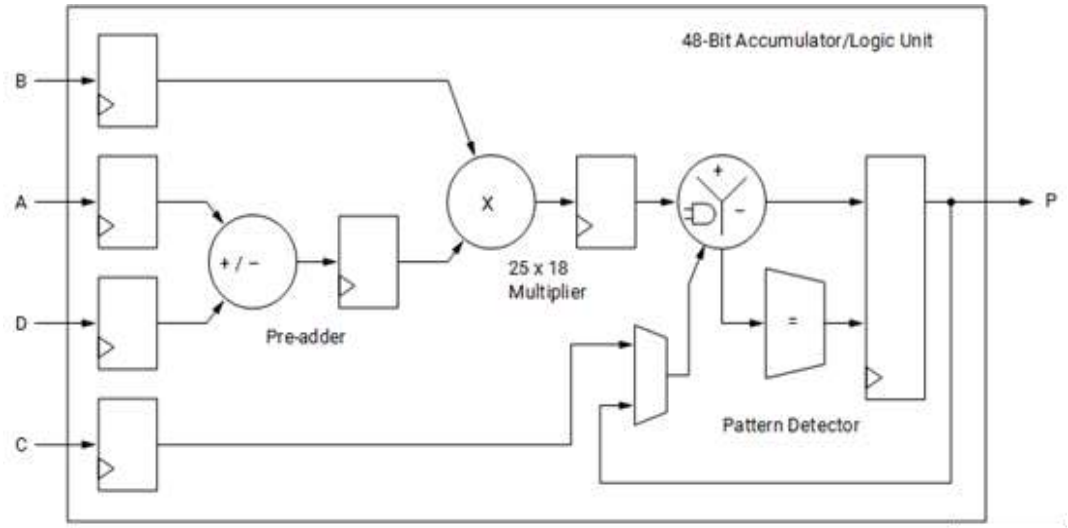
NxMap 22.2.0.4		fft240, no constraints		fft240, floorplanning, not timing driven		fft240, floorplanning, timing driven	
Ressource	total	abs	relative	abs	relative	abs	relative
4-LUT	505344	2988	0.59%	7959	1.57%	7959	1.57%
DFF	505344	5807	1.15%	7078	1.40%	7079	1.40%
Carry	126336	1374	1.09%	1120	0.89%	1120	0.89%
RFB	2632	64	2.43%	0	0.00%	0	0.00%
DSP	1344	22	1.64%	31	2.31%	31	2.31%
BRAM	672	20	2.98%	24	3.57%	24	3.57%
clk (post syn) in MHz		213,995		206,954		206,954	
clk (post rout) in MHz		131,199		178,285		183,993	

Garbage in = Garbage out

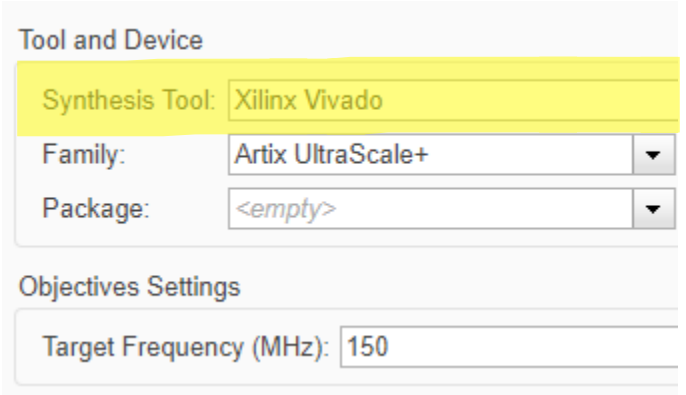
- Following good coding patterns is important
  (= good modelling patterns)

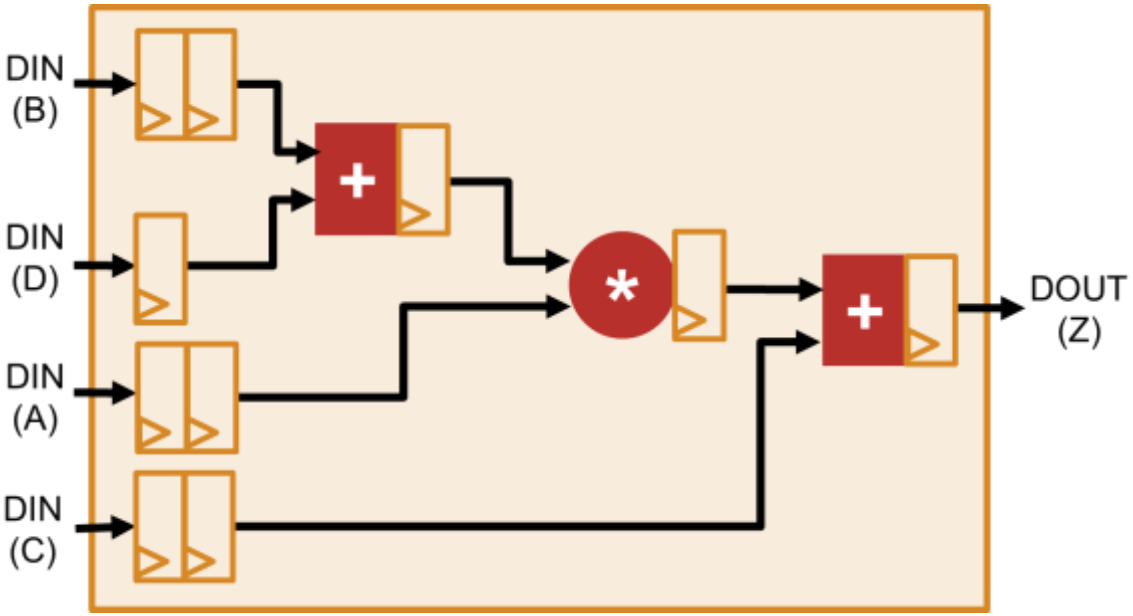
- Fortunately …. HDL Coder produces good code for NanoXplore
  Why is that??

DSP architecture NG Ultra and Xilinx are ‘similar’

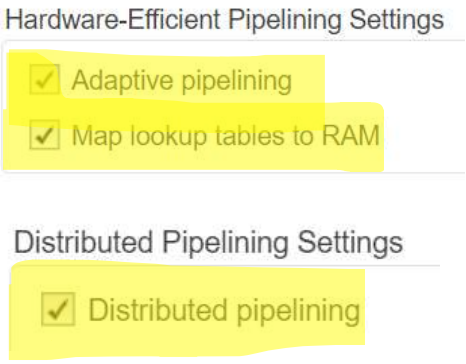


Xilinx

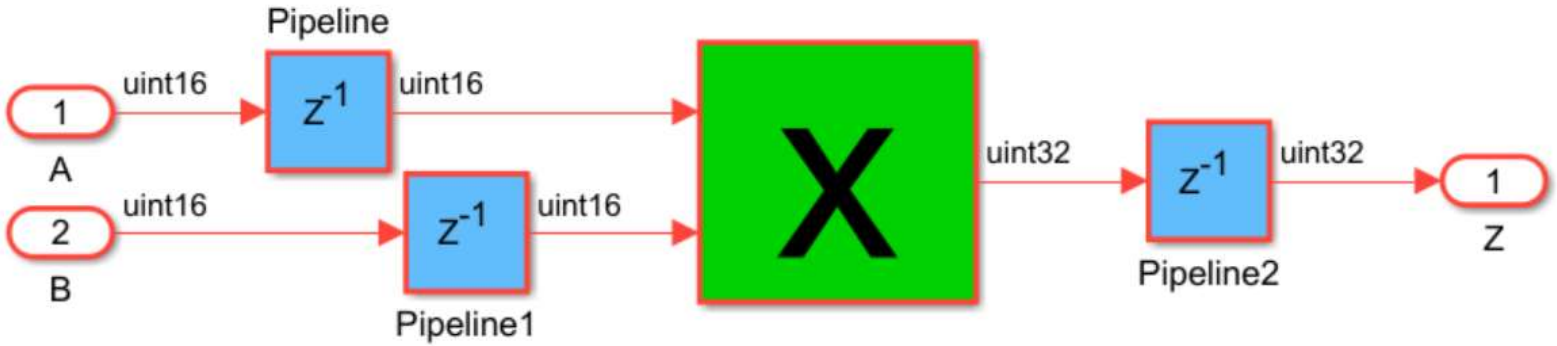


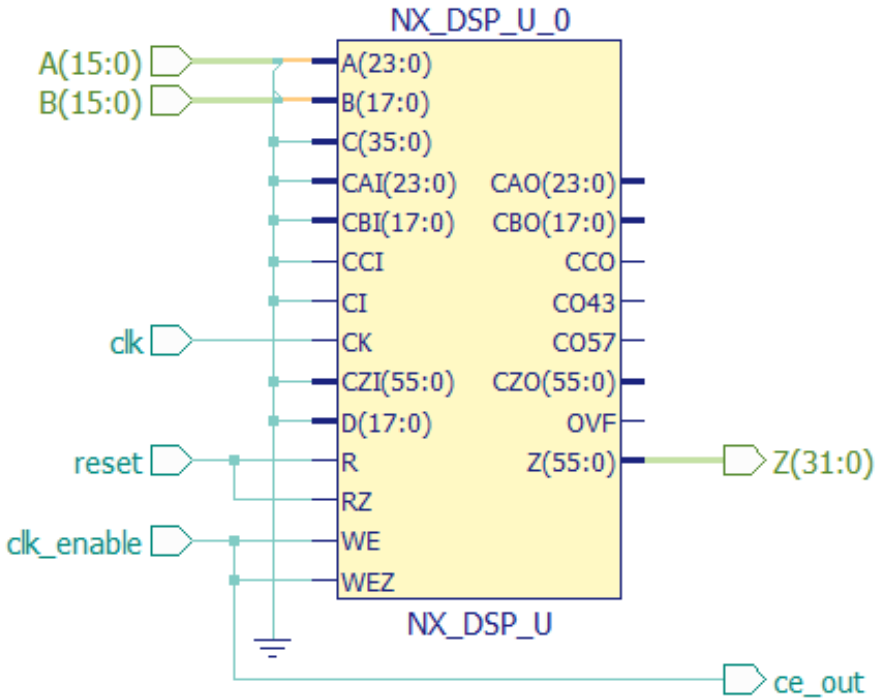


NanoXplore



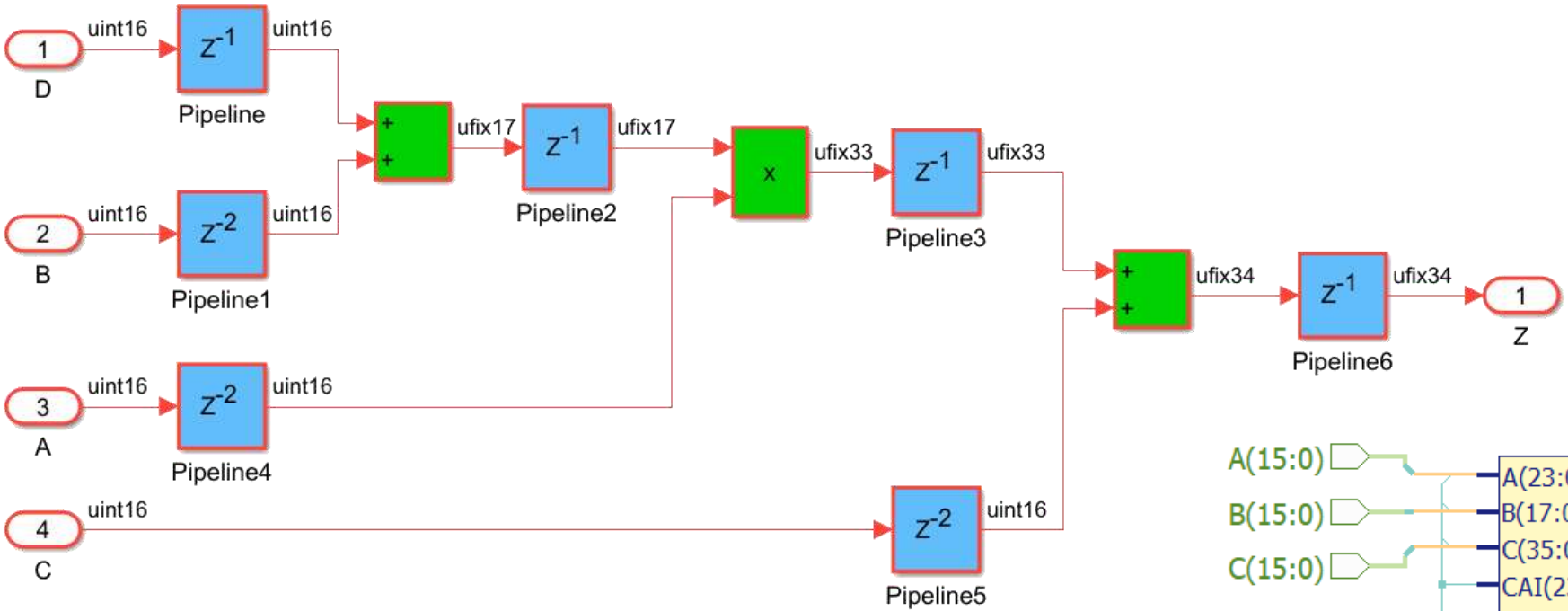
Pipelined Multiply

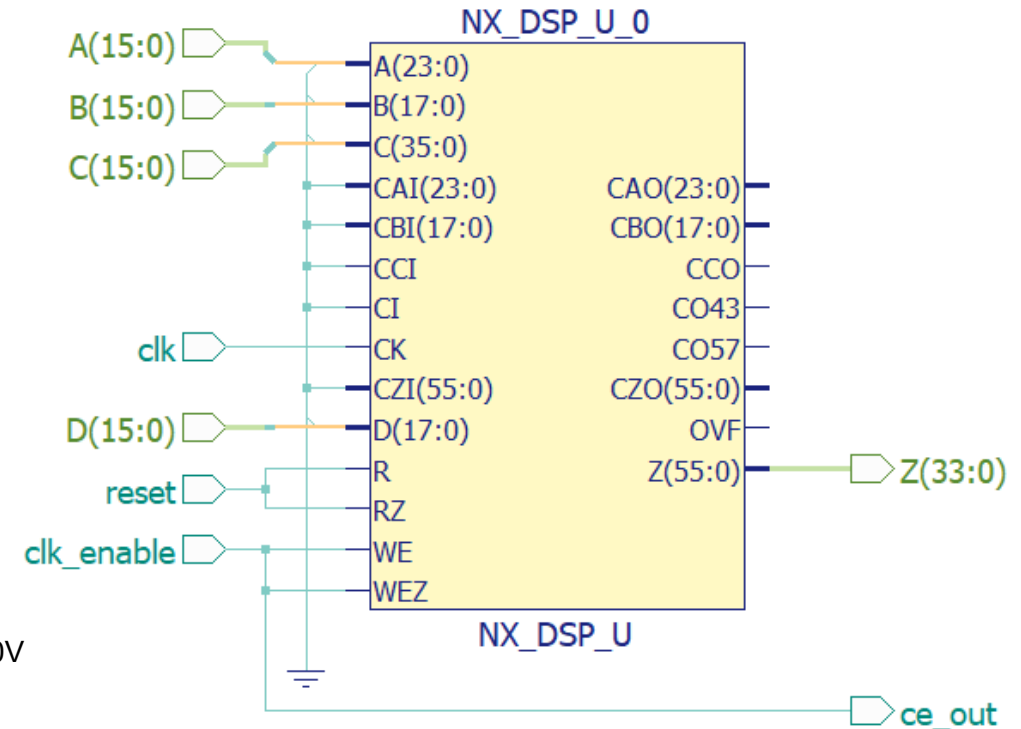




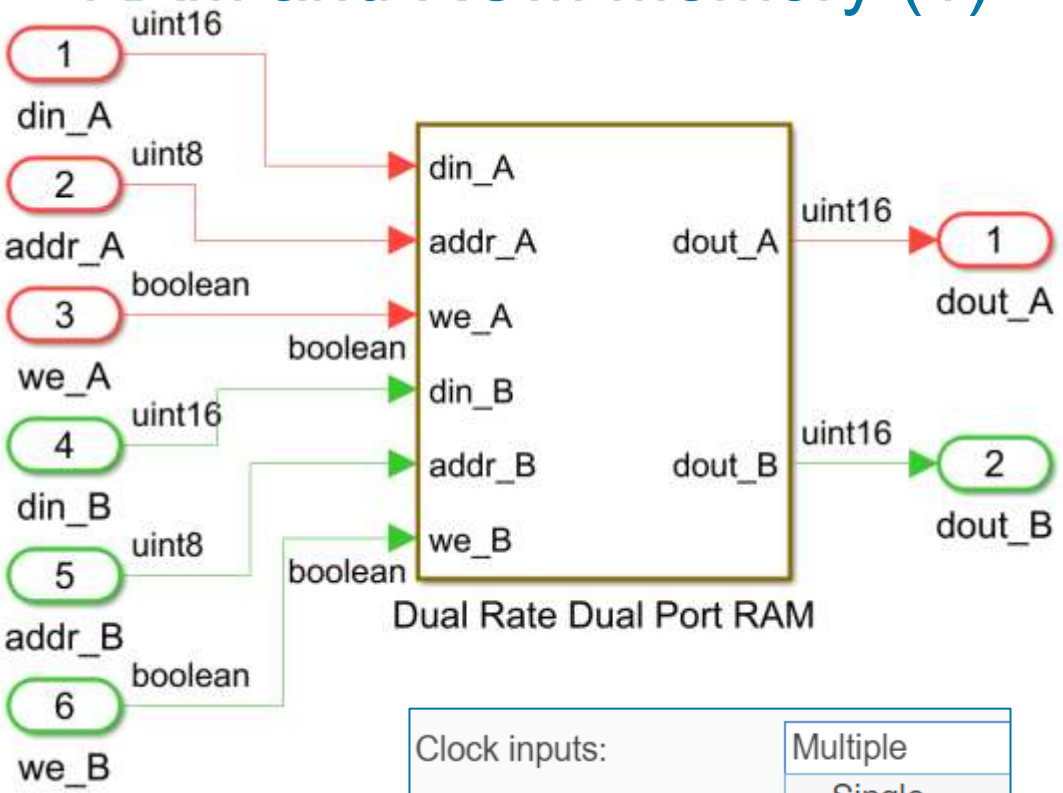

```
A_unsigned <= unsigned(A);

enb <= clk_enable;

Pipeline_process : PROCESS (clk)
BEGIN
  IF clk'EVENT AND clk = '1' THEN
    IF reset = '1' THEN
      Pipeline_out1 <= to_unsigned(16#0000#, 16);
    ELSIF enb = '1' THEN
      Pipeline_out1 <= A_unsigned;
    END IF;
  END IF;
END PROCESS Pipeline_process;


B_unsigned <= unsigned(B);

Pipeline1_process : PROCESS (clk)
BEGIN
  IF clk'EVENT AND clk = '1' THEN
    IF reset = '1' THEN
      Pipeline1_out1 <= to_unsigned(16#0000#, 16);
    ELSIF enb = '1' THEN
      Pipeline1_out1 <= B_unsigned;
    END IF;
  END IF;
END PROCESS Pipeline1_process;


Product_out1 <= Pipeline_out1 * Pipeline1_out1;

Pipeline2_process : PROCESS (clk)
BEGIN
  IF clk'EVENT AND clk = '1' THEN
    IF reset = '1' THEN
      Pipeline2_out1 <= to_unsigned(0, 32);
    ELSIF enb = '1' THEN
      Pipeline2_out1 <= Product_out1;
    END IF;
  END IF;
END PROCESS Pipeline2_process;


Z <= std_logic_vector(Pipeline2_out1);
```

Pipelined Multiply-Add with Pre-Adder

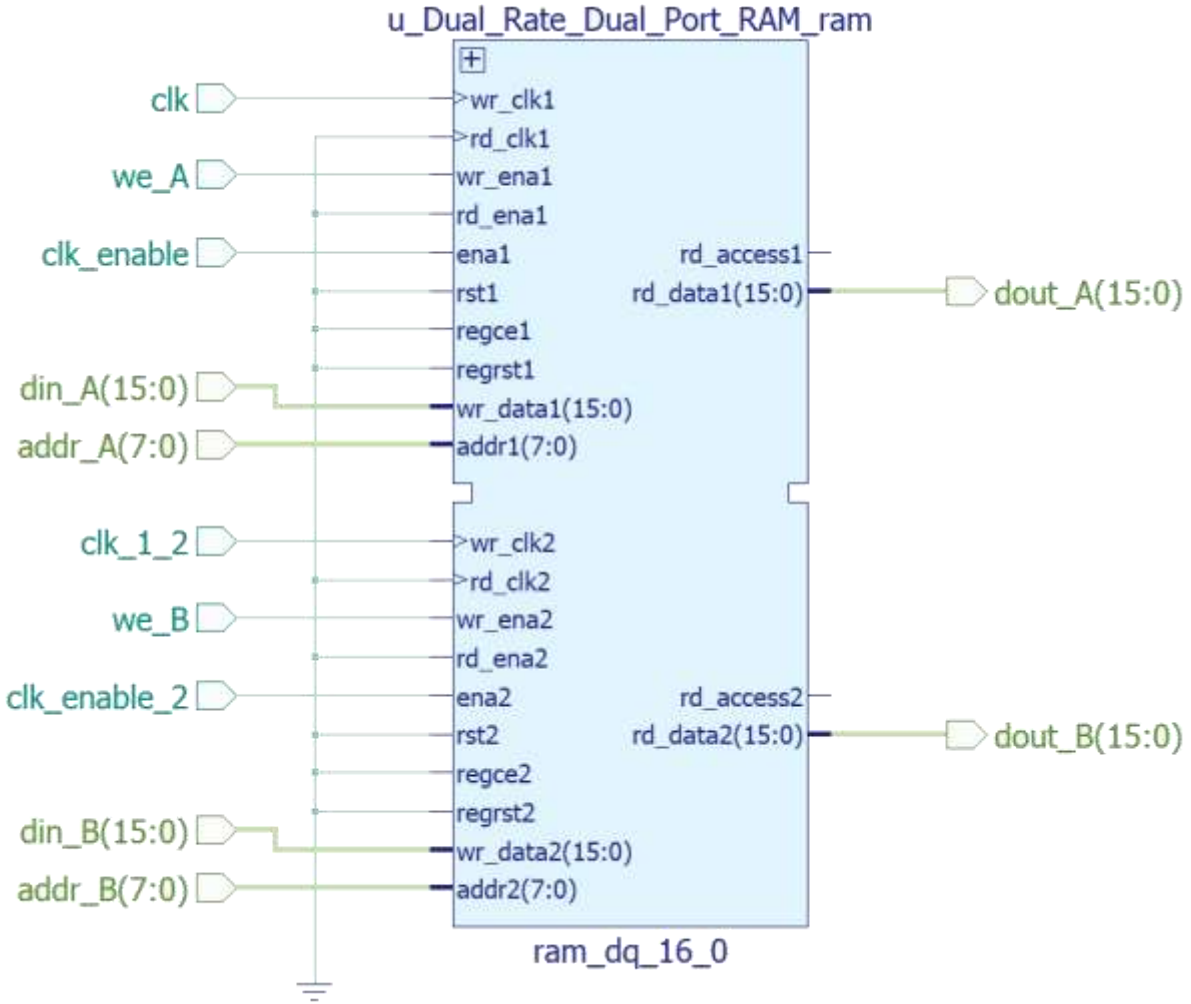




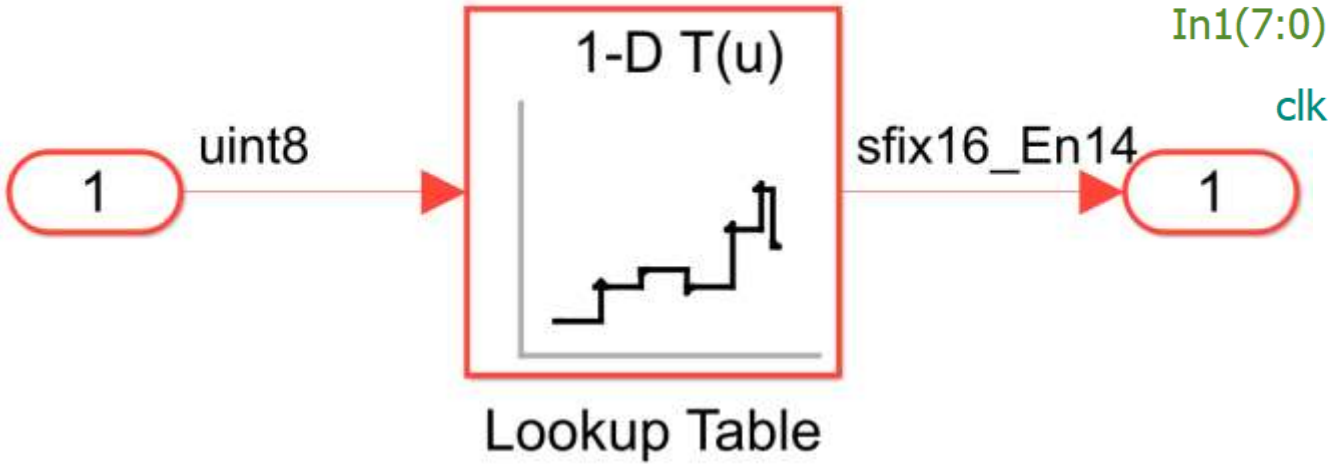
RAM and ROM Memory (1)




```
ENTITY FPGASubsystem IS
  PORT( clk
        clk_1_2
        clk_enable
        clk_enable_2
```



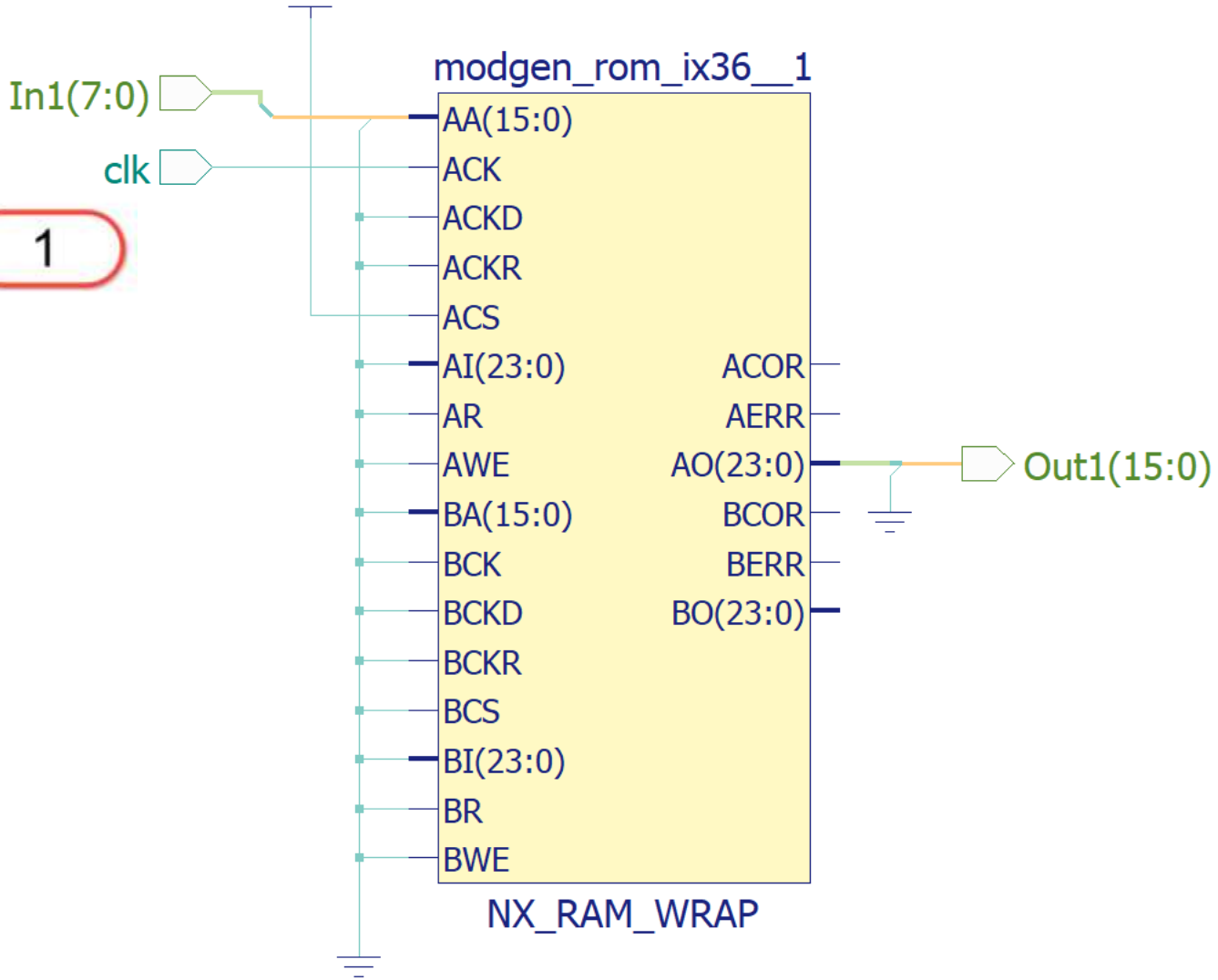
RAM and ROM Memory (2)



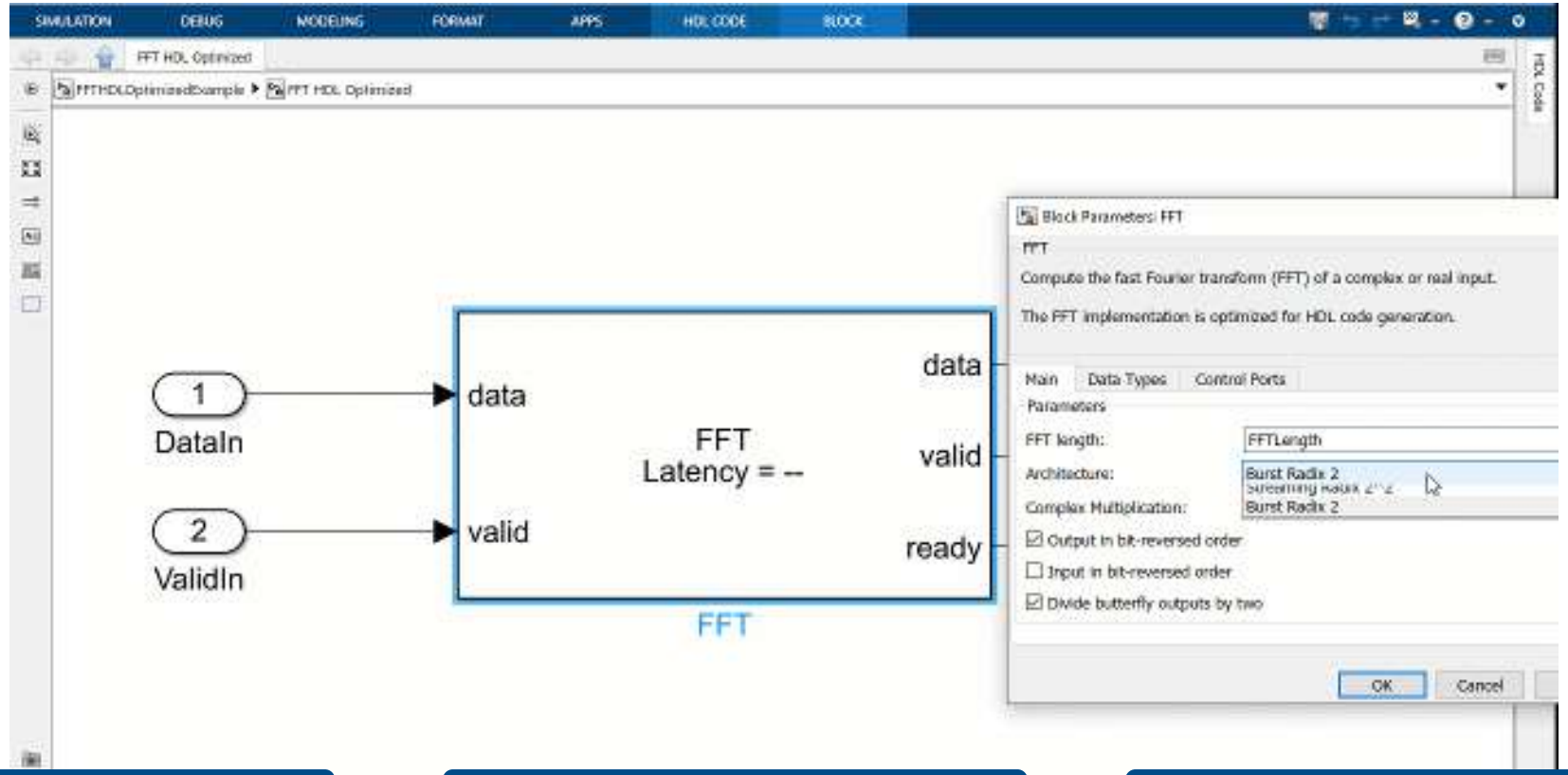
Hardware-Efficient Pipelining Settings

- [ ] Adaptive pipelining
- [x] Map lookup tables to RAM

```
PipelineRegister_process : PROCESS (clk)
BEGIN
  IF clk'EVENT AND clk = '1' THEN
    Lookup_Table_out1 <= out_rsvd;
  END IF;
END PROCESS PipelineRegister_process;
```



FFT Implementation Exploration



Minimize resources

- Burst Radix 2
- Latency = 2721 cycles
- 4 multipliers
- 6 RAMs

Low latency, Streaming

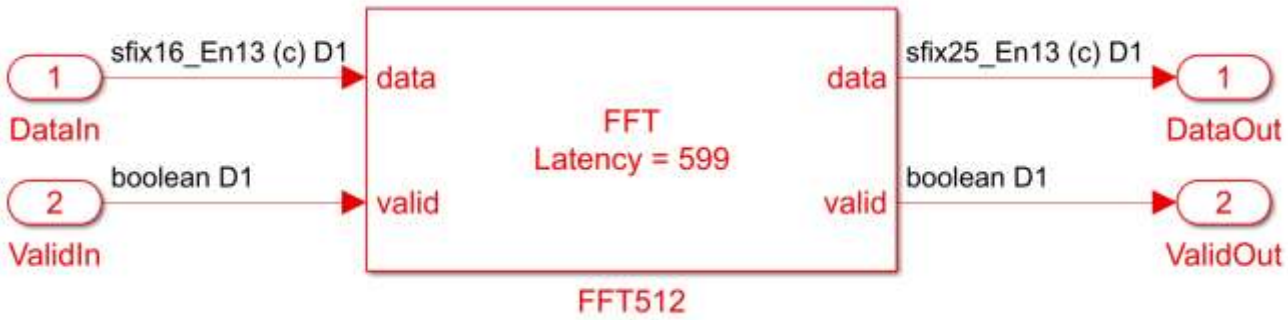
- Streaming Radix 2^2
- Latency = 599 cycles
- 16 multipliers
- 32 RAMs

High throughput, GSPS

- Streaming Radix 2^2
- Latency = 139 cycles
- Frame input size = 8
- 108 multipliers
- 160 RAMs

3.75 GSPS

Collaboration with NanoXplore



Multipliers	16
Adders/Subtractors	191
Registers	647
Total 1-Bit Registers	8504
RAMs	32
Multiplexers	461
I/O Bits	86
Static Shift operators	8
Dynamic Shift operators	0

```
 Precision DSP Inference Report For NX2H540TSC-LF1760V
************************************************************
+-------------------------------------------------------------------------------------------------------------------------------------------------+
|    | Instance Name                          | Out Net                                  | Mode      | AREG | BREG | CREG | ADREG | MREG | PREG |
+-------------------------------------------------------------------------------------------------------------------------------------------------+
|  1.| u_FFT.u_SDF1_3_1.u_MUL4.NX_DSP_U_0     | RETIMED_rtlcGen22                        | B*A       | 2    | 2    | 0    | -     | 0    | 0    |
|  2.| u_FFT.u_SDF1_3_1.u_MUL4.NX_DSP_U_1     | RETIMED_rtlcGen18                        | B*A       | 2    | 2    | 0    | -     | 0    | 0    |
|  3.| u_FFT.u_SDF1_3_1.u_MUL4.NX_DSP_U_2     | dinXTwdl_re                              | A*B-C     | 0    | 0    | 1    | -     | 1    | 1    |
|  4.| u_FFT.u_SDF1_3_1.u_MUL4.NX_DSP_U_3     | dinXTwdl_im                              | A*B+C     | 0    | 0    | 1    | -     | 1    | 1    |
|  5.| u_FFT.u_SDF1_5_1.u_MUL4.NX_DSP_U_0     | RETIMED_rtlcGen48                        | B*A       | 2    | 2    | 0    | -     | 0    | 0    |
|  6.| u_FFT.u_SDF1_5_1.u_MUL4.NX_DSP_U_1     | RETIMED_rtlcGen44                        | B*A       | 2    | 2    | 0    | -     | 0    | 0    |
|  7.| u_FFT.u_SDF1_5_1.u_MUL4.NX_DSP_U_2     | dinXTwdl_re                              | A*B-C     | 0    | 0    | 1    | -     | 1    | 1    |
|  8.| u_FFT.u_SDF1_5_1.u_MUL4.NX_DSP_U_3     | dinXTwdl_im                              | A*B+C     | 0    | 0    | 1    | -     | 1    | 1    |
|  9.| u_FFT.u_SDF1_7_1.u_MUL4.NX_DSP_U_0     | Complex4Multiply_mult2_im_pipel_5n0rl    | A*B       | 1    | 1    | -    | -     | 1    | 1    |
| 10.| u_FFT.u_SDF1_7_1.u_MUL4.NX_DSP_U_1     | Complex4Multiply_mult2_re_pipel_5n0rl    | A*B       | 1    | 1    | -    | -     | 1    | 1    |
| 11.| u_FFT.u_SDF1_7_1.u_MUL4.NX_DSP_U_2     | Complex4Multiply_mult2_im_pipel_5n0rl    | A*B       | 1    | 1    | -    | -     | 1    | 1    |
| 12.| u_FFT.u_SDF1_7_1.u_MUL4.NX_DSP_U_3     | Complex4Multiply_mult2_re_pipel_5n0rl    | A*B       | 1    | 1    | -    | -     | 1    | 1    |
| 13.| u_FFT.u_SDF1_9_1.u_MUL4.NX_DSP_U_0     | RETIMED_rtlcGen74                        | B*A       | 2    | 2    | 0    | -     | 0    | 0    |
| 14.| u_FFT.u_SDF1_9_1.u_MUL4.NX_DSP_U_1     | RETIMED_rtlcGen70                        | B*A       | 2    | 2    | 0    | -     | 0    | 0    |
| 15.| u_FFT.u_SDF1_9_1.u_MUL4.NX_DSP_U_2     | dinXTwdl_re                              | A*B-C     | 0    | 0    | 1    | -     | 1    | 1    |
| 16.| u_FFT.u_SDF1_9_1.u_MUL4.NX_DSP_U_3     | dinXTwdl_im                              | A*B+C     | 0    | 0    | 1    | -     | 1    | 1    |
+-------------------------------------------------------------------------------------------------------------------------------------------------+
```

```
( 23.95% cell delay, 76.05% net delay )

Min Period (Freq)          Required Period (Freq)
-----------------          ----------------------
5.898 (169.549 MHz)        6.667 (149.999 MHz)
```

FFT HDL - Optimized

Frequency without constraints :

- 135MHz @ worstcase

Frequency with floorplanning:

- 136MHz @ worstcase

Frequency with advanced floorplanning:

- 146MHz @ worstcase
- 150MHz can easily be reached
- 155-160MHz will be the limit

FFT HDL - Optimized

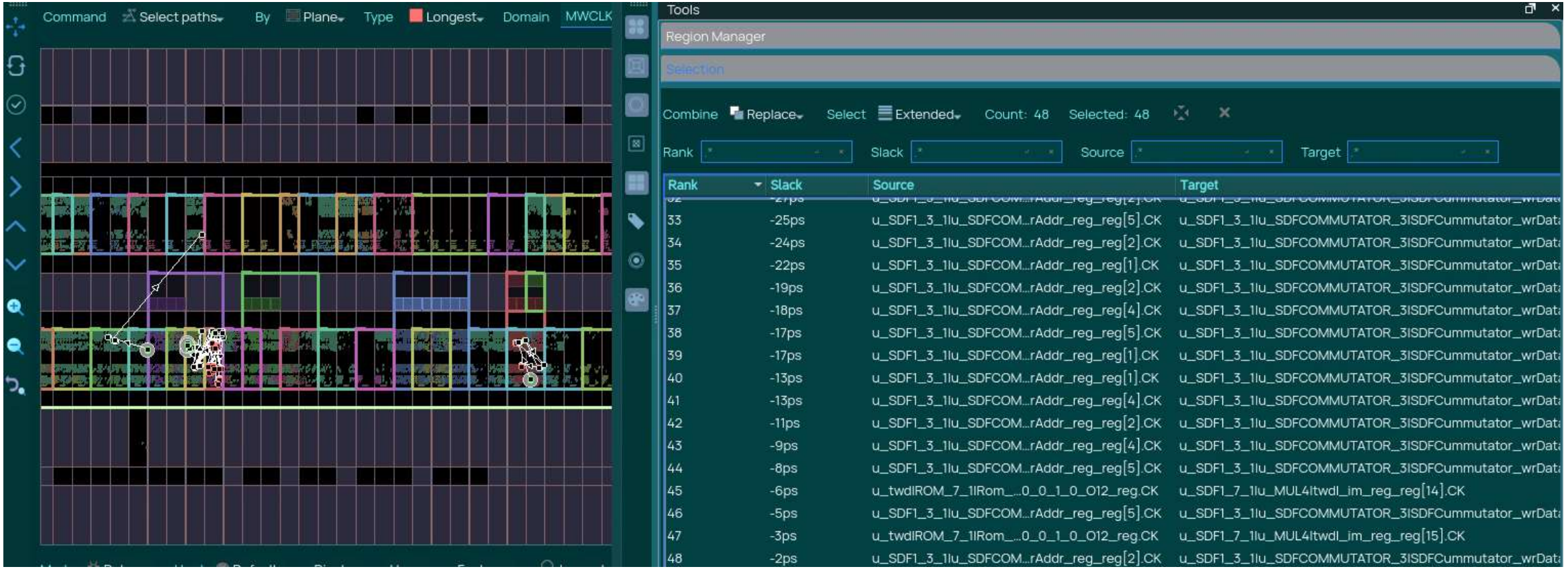
150MHz

Reporting domains

Domain		Frequency		Hold/Removal Summary			Setup/Recovery Summary			Status		
Source	Target	Required	Actual	Slack	Minimum Data Arrival Time	Minimum Required Relationship	Slack	Maximum Data Arrival Time	Maximum Required Relationship	Slack	Delay Path	Total
# Input	MWCLK (Rising)	-	-	-	-1.434ns	-	-	6.682ns	-	-	-	- #
# MWCLK (Rising)	Output	-	-	-	12.713ns	-	-	19.301ns	-	-	-	- #
MWCLK (Rising)	MWCLK (Rising)	150.015 MHz	146.135 MHz	297ps	297ps	0ps	-177ps	6.843ns	6.666ns	KO	-	KO
Total									3	KO	-	KO

FFT HDL - Optimized

150MHz



Concluding remarks

AIRBUS + NanoXplore +  =