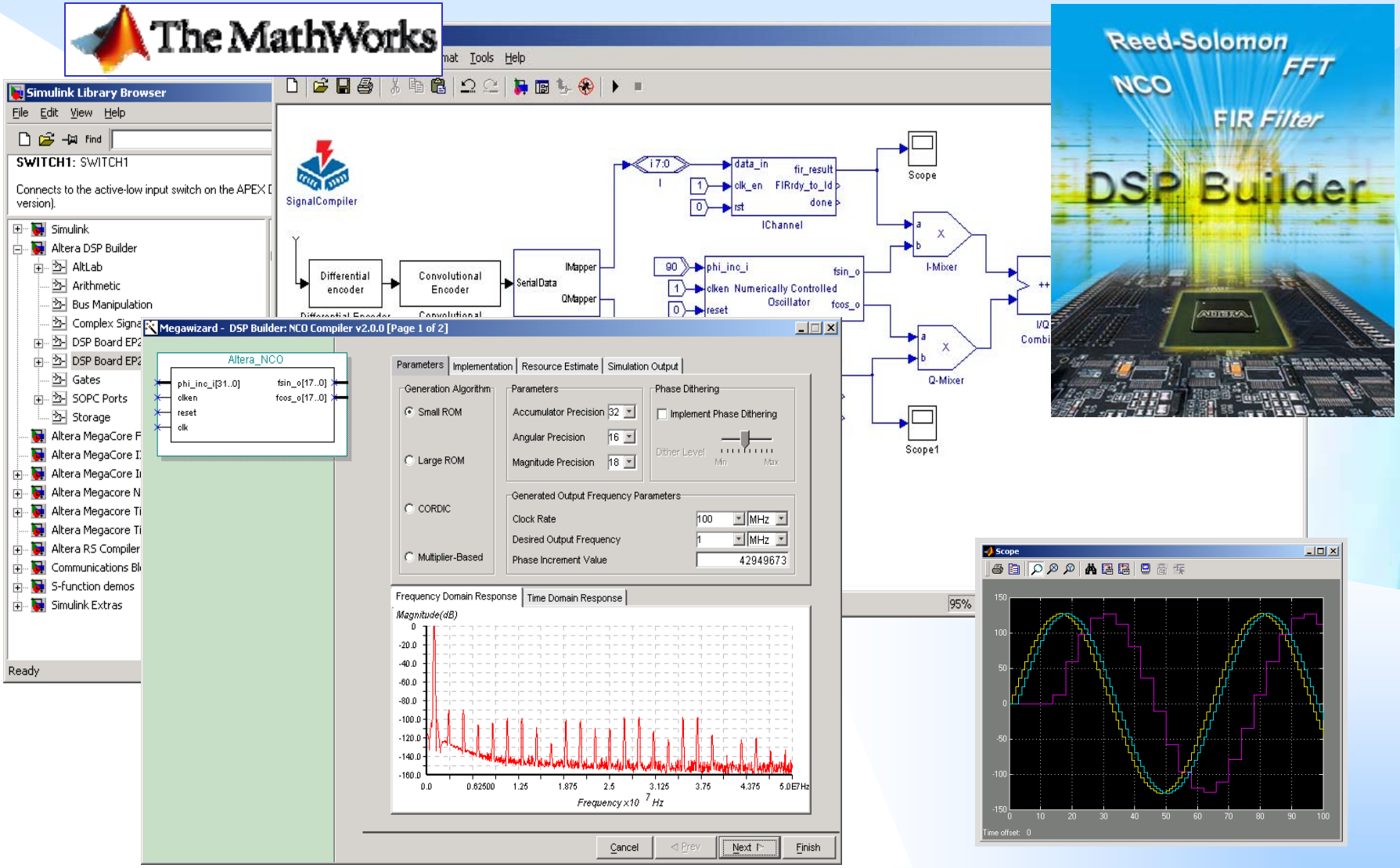
FPGA Design Using DSP Builder Advanced Blockset

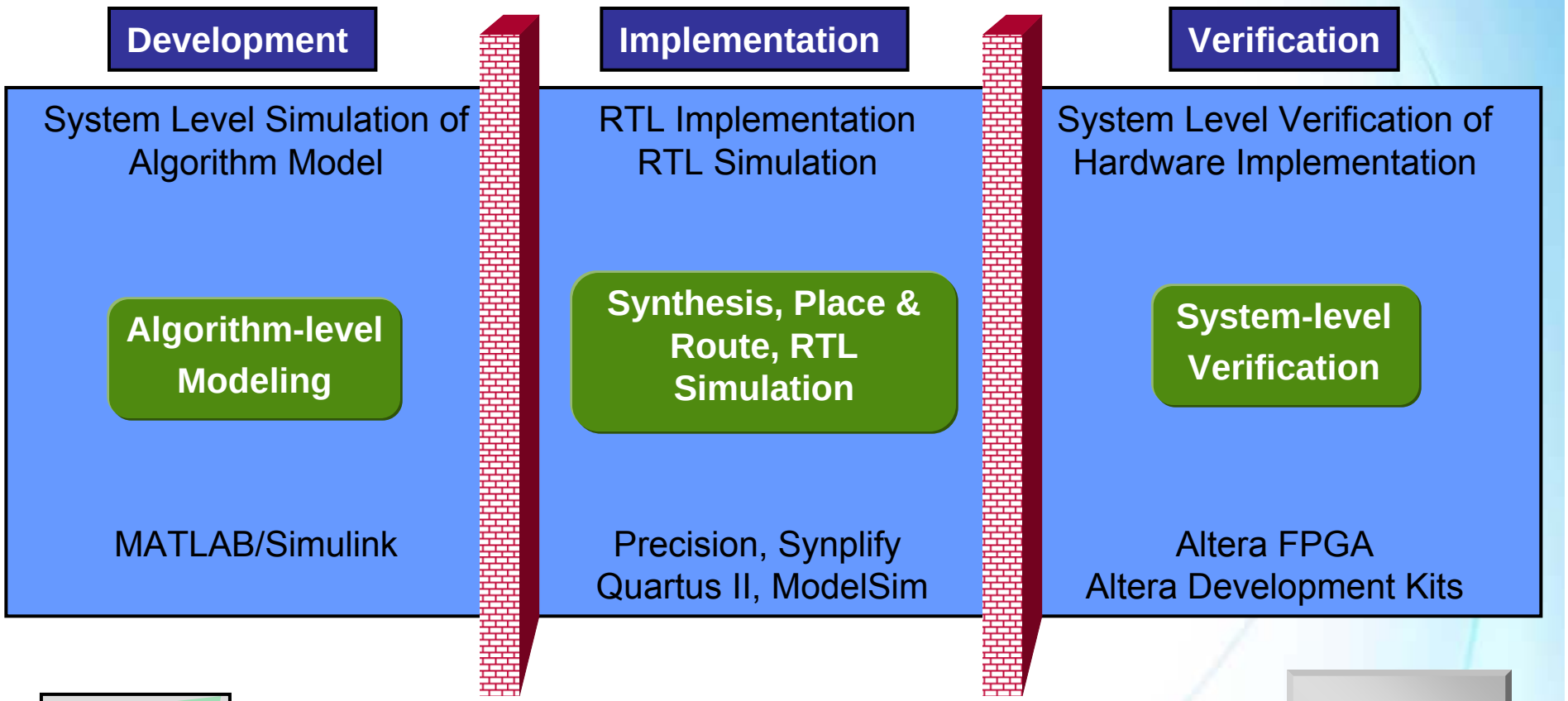
Agenda

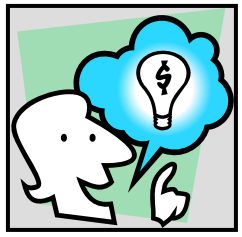
- Introduction
  - Model based design flow for Altera FPGAs
  - DSP Builder Features
- DSP Builder Advanced Blockset
  - Introduction to constraint driven model based design
- Advanced Blockset design examples
  - Radar front end processor: Interfacing to high speed A/Ds
  - Direct RF upconversion: Interfacing to multi-gigabit DACs
  - 8x8 Beam Former

Simulink Model Based Algorithm Development



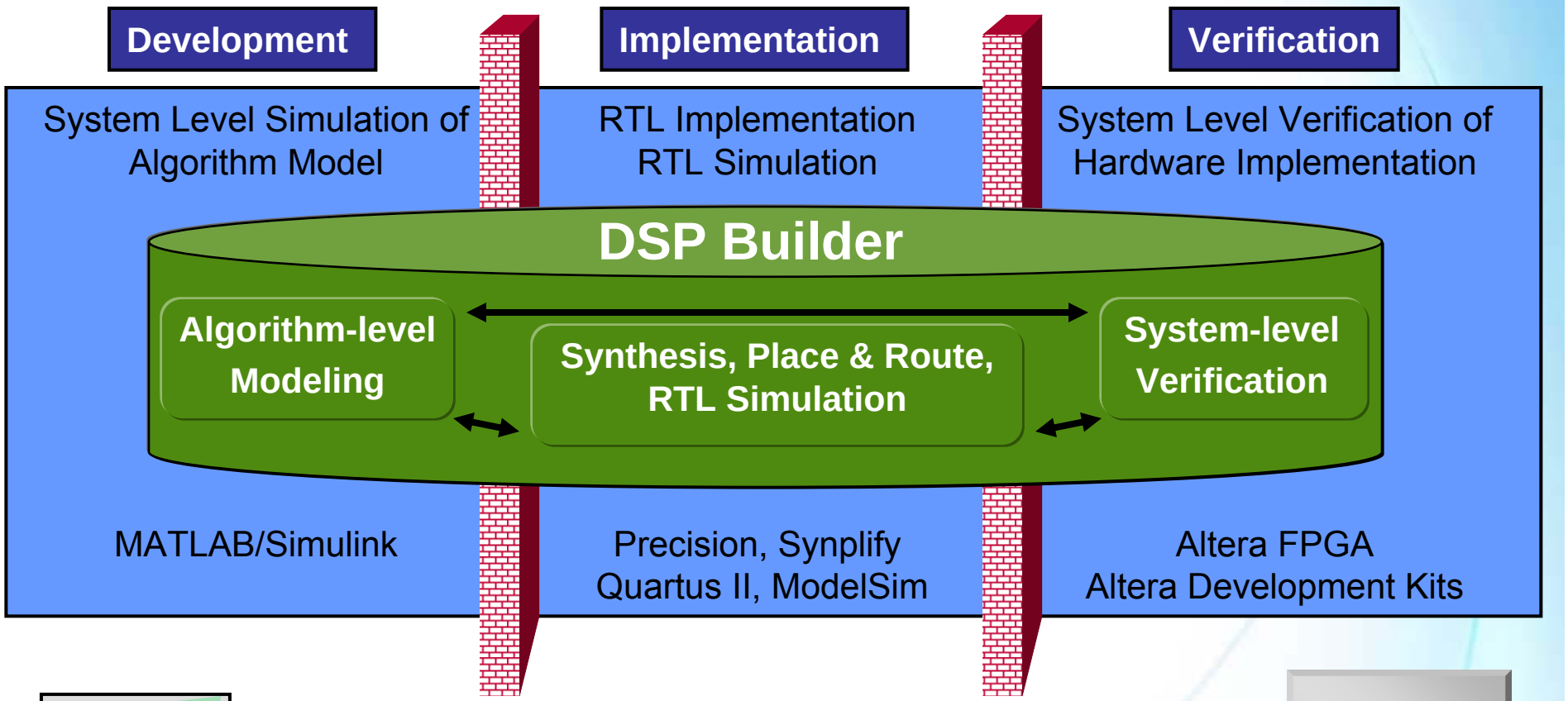
DSP Builder System Level Design Flow

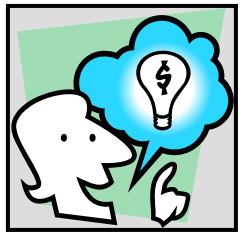




ALTERA

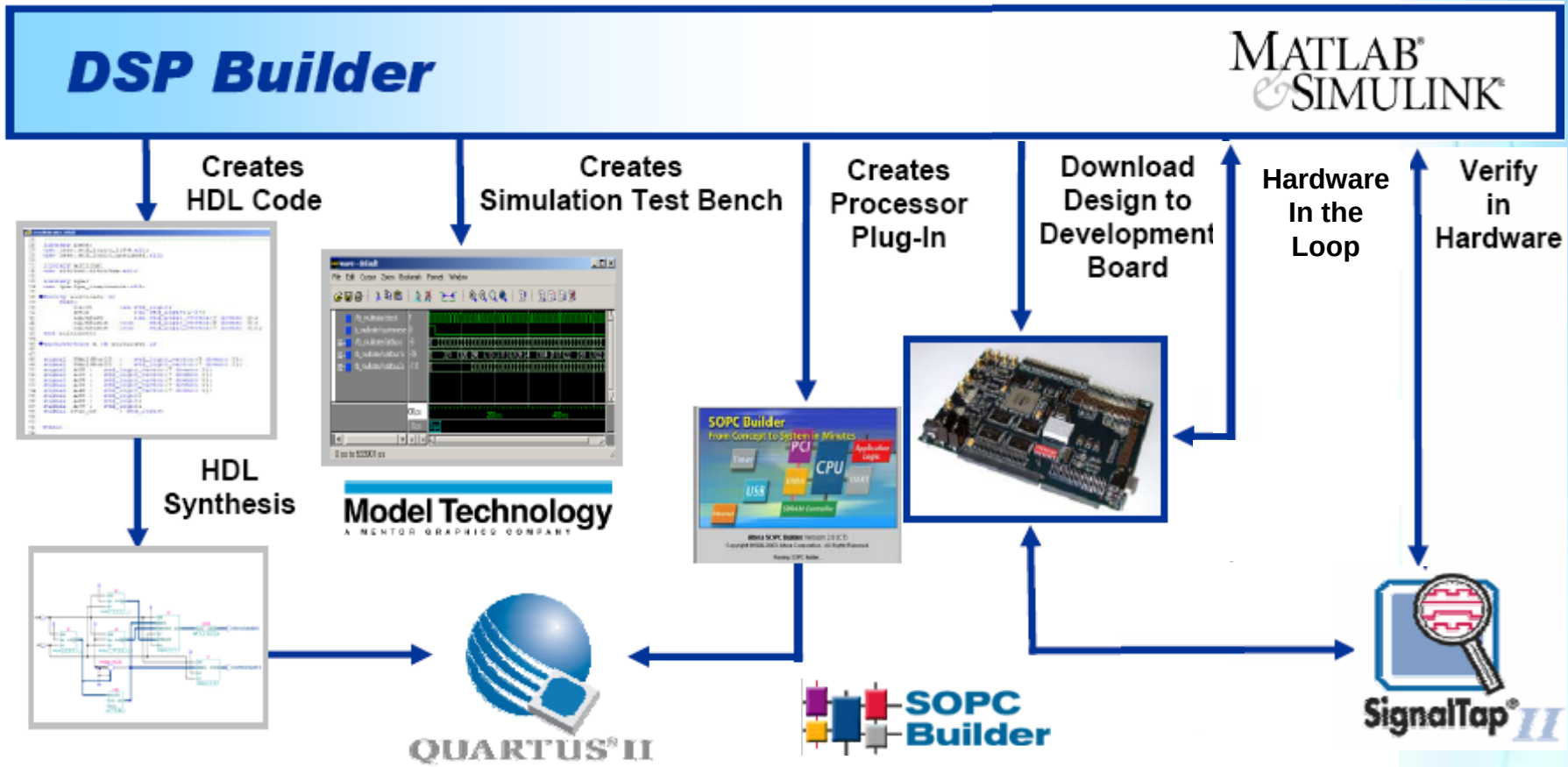
DSP Builder System Level Design Flow



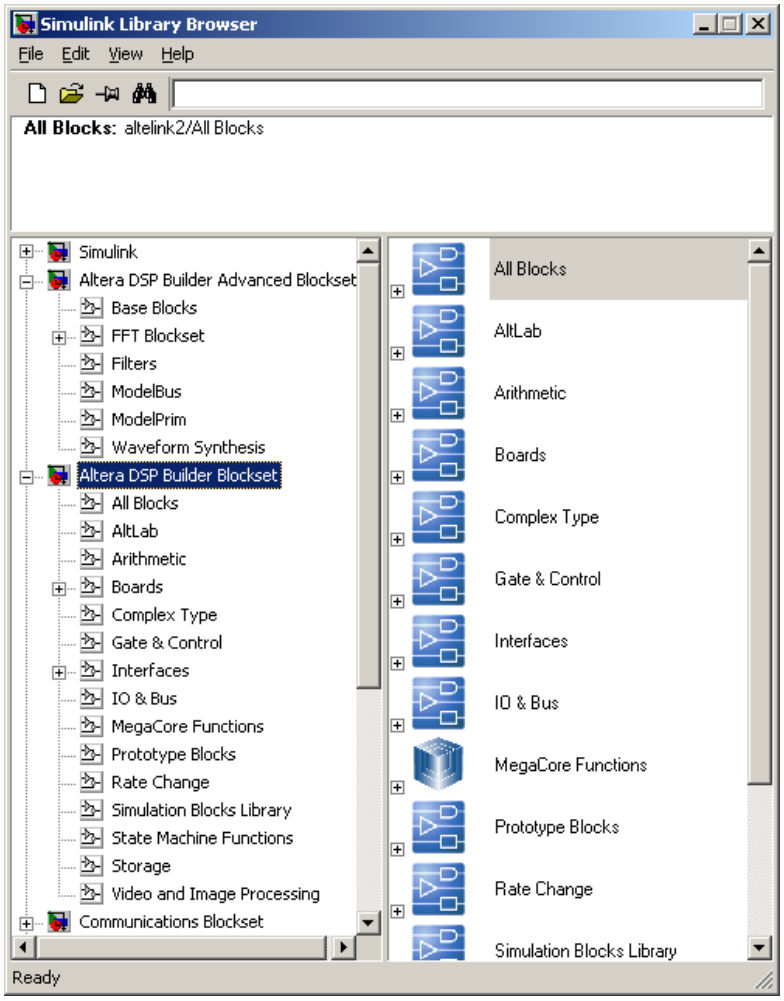


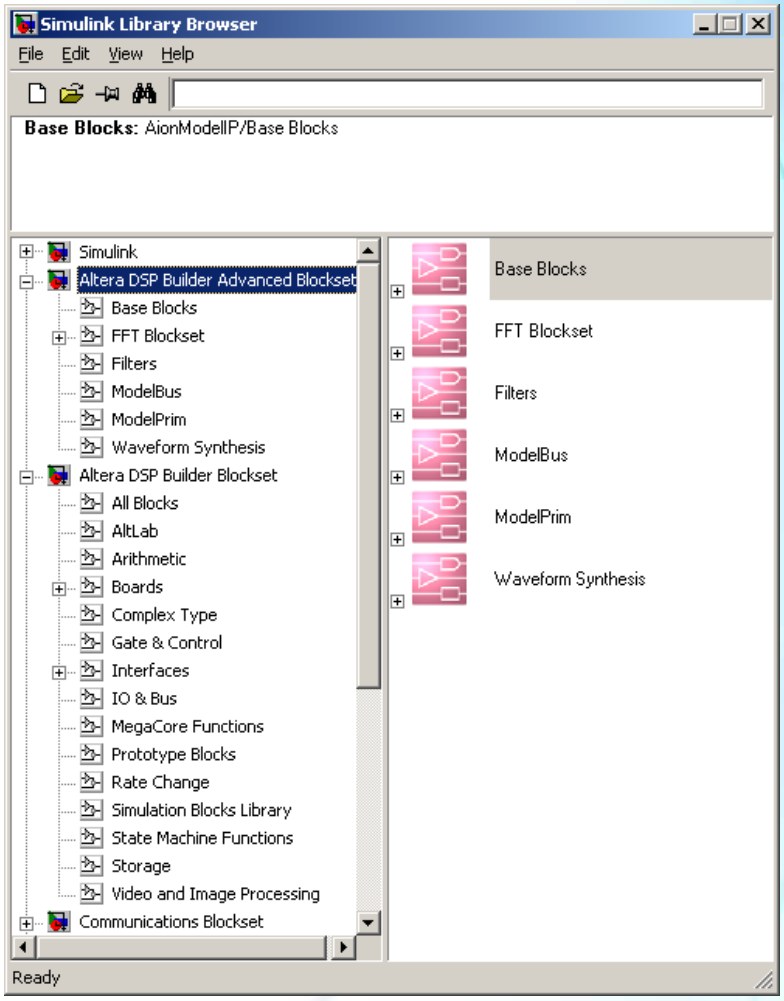
ALTERA

DSP Builder Overview



DSP Builder – Standard and Advanced Blocksets





DSP Builder Features

- Automatic Generation of VHDL design from a MATLAB/Simulink model
- Automatic Generation of Testbenchs
  - Captures Stimulus From Simulink, Writes Testbench
- HDL Import
  - Reads in Design : HDL: Verilog or VHDL, or Quartus Project
  - Creates Simulink Simulation Model
- Signal Tap: Embedded Logic Analyzer
  - Captures Internal Data And Brings It Into Matlab
- Hardware in Loop (HIL) Testing
  - Pass Vectors To / From Board
- Waveform Viewer
  - Visualize Waveforms as Digital Busses Using ModelSim

DSP Builder Features

- SOPC Bus Support
  - Avalon Masters & Slaves, Custom Instruction Set (Nios)
- IP Support
  - FIR, NCO, FFT, Reed Solomon, Viterbi, CIC
  - Other IP through HDL Import
- Data Width Propagation
  - Automatically Propagates Bus Widths Through Signal Path
- Multi Data Rate Support
  - PLLs or Clock Enables
  - Multi-rate FIFO
- Integration of DSP Boards
  - Customer board easily integrated
- Complex Data

DSP Builder
Advanced Blockset

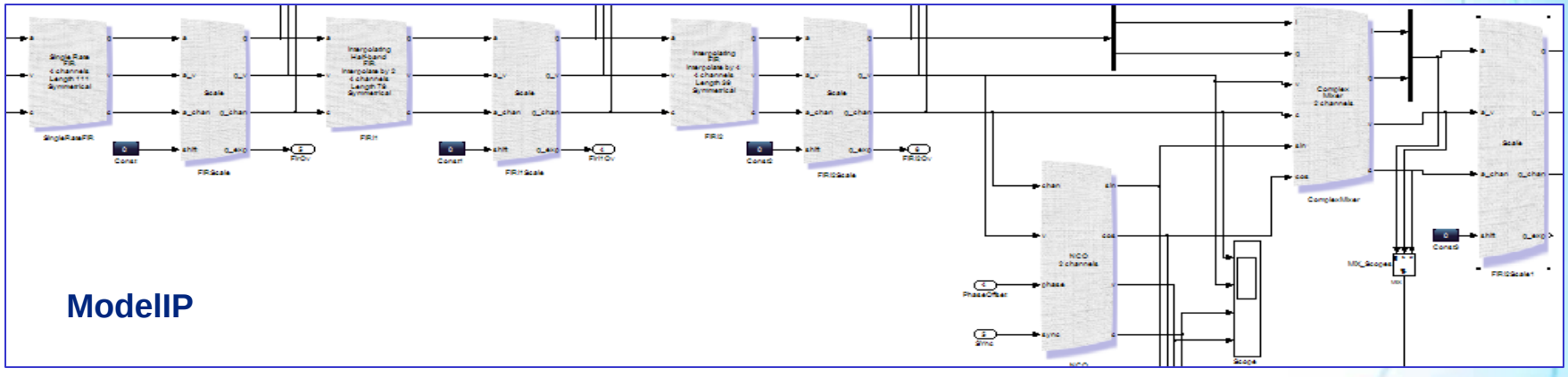
What's New in Advanced Blockset?

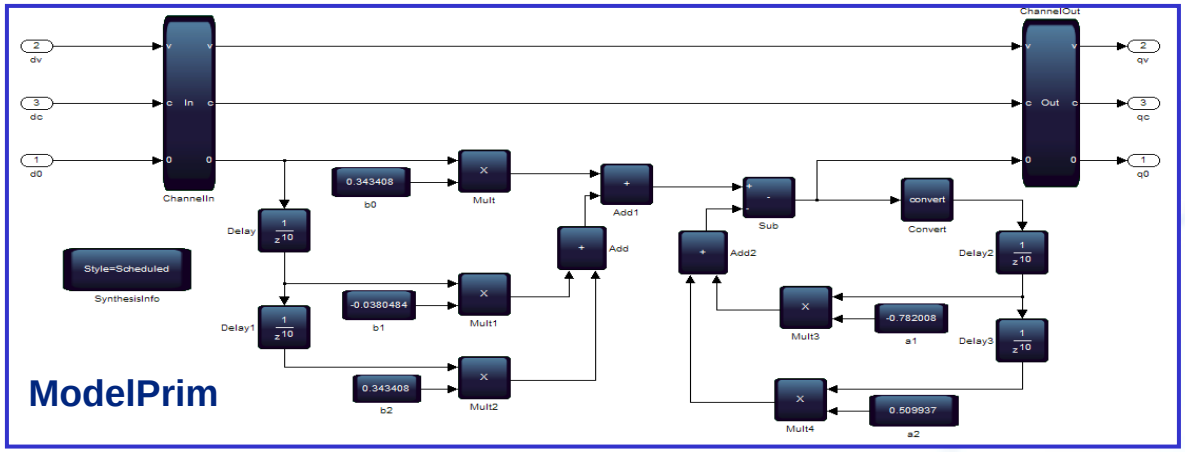
- Constraint-Driven Design
  - Automated pipelining
    - Meet desired clock rate
    - Enable timing closure at high clock rates of 400-500 MHz
- Automatic TDM Support for ModelIP
  - ModelIP reuses the resources efficiently
- "Textbook" Design with ModelPrim
- Multi-Channel Designs Made Easy
- Memory-map Register Generation
  - Allow easy configuration of coefficients and run-time parameters

Increased Productivity by Closing Timing Faster

Constraint Driven Design: (1) Create Model

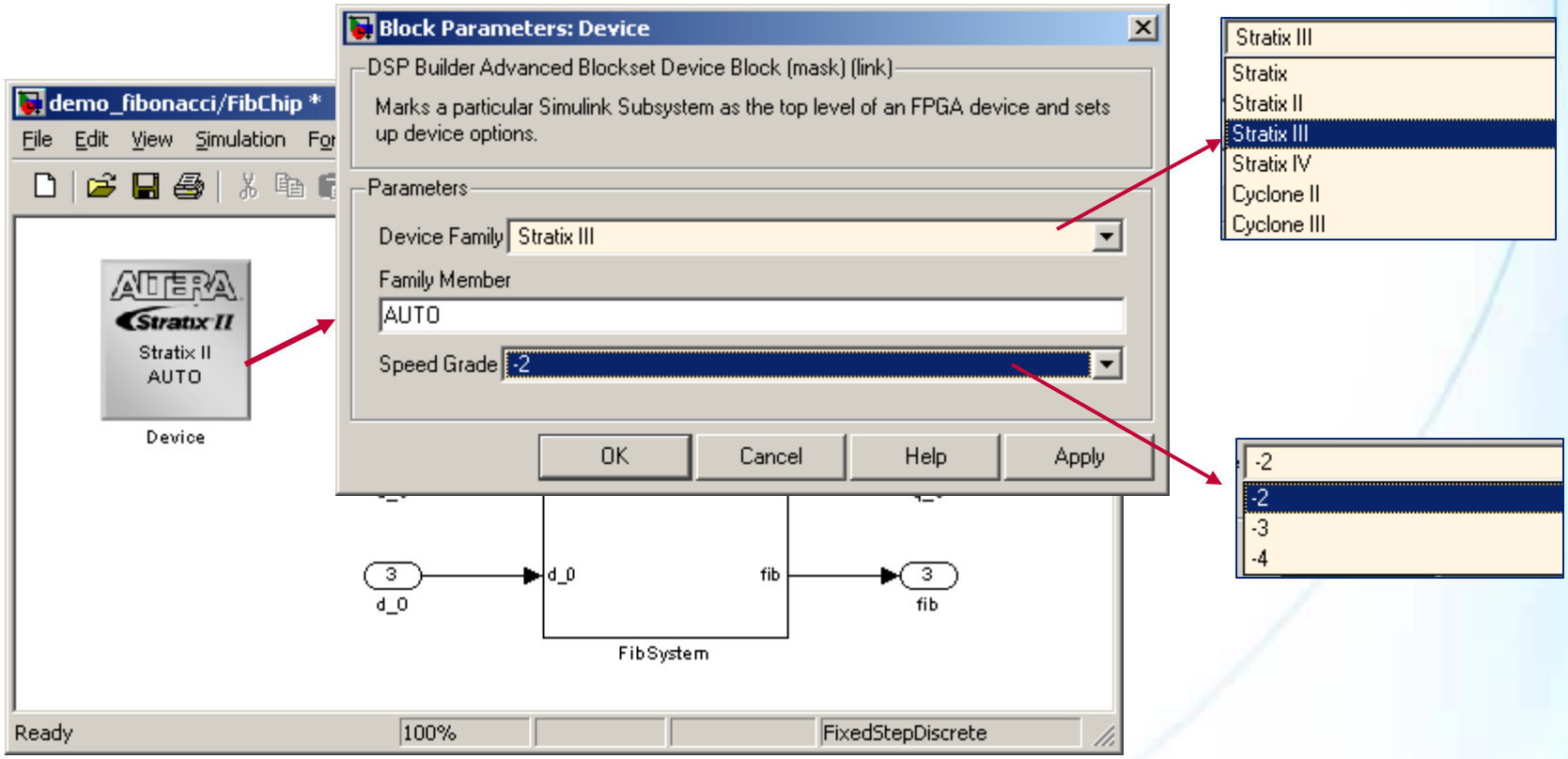
- Use ModelIP or ModelPrim Libraries





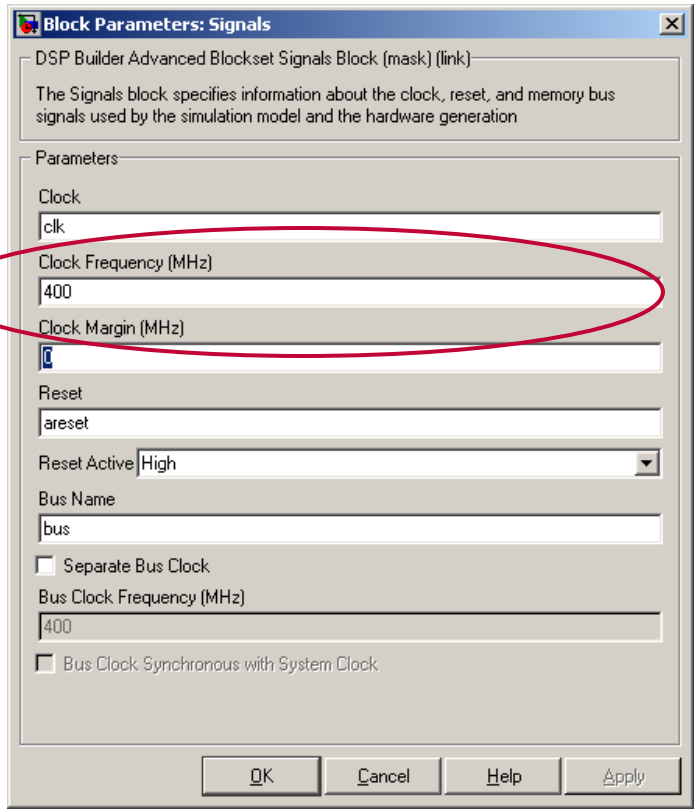
Constraint Driven Design: (2) Select Device

- Device independent modeling until this level

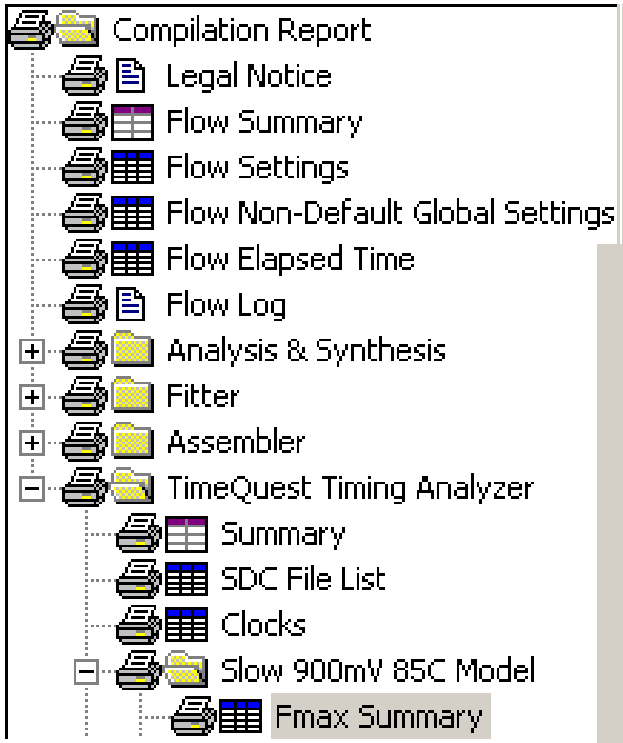


Constraint Driven Design: (3) Set Frequency

- Automatic Pipelining / Time Sharing (ModelIP)



Constraint Driven Design: (4) Compile

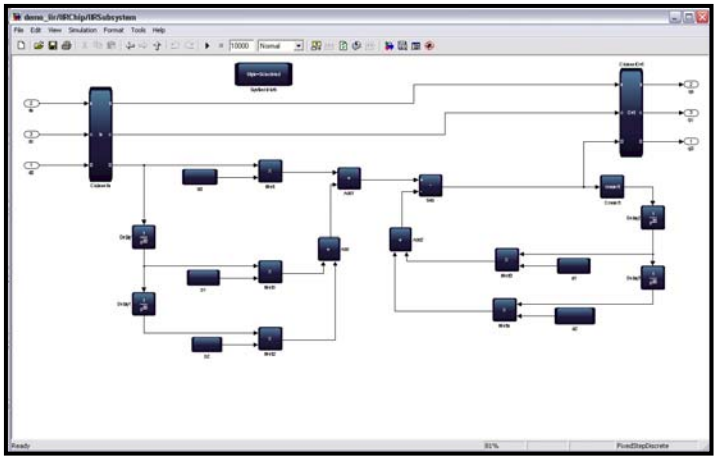


Slow 900mV 85C Model Fmax Summary

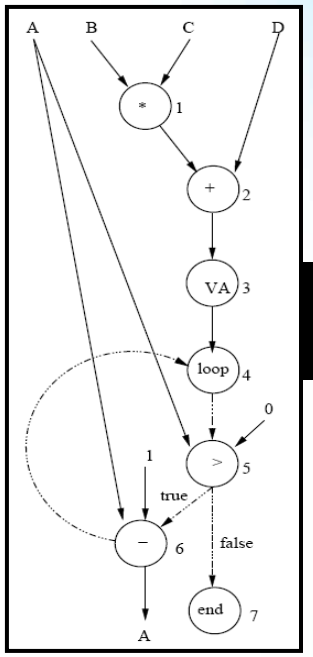
	Fmax	Restricted Fmax	Clock Name	Note
1	307.22 MHz	307.22 MHz	bus_clk	
2	408.66 MHz	408.66 MHz	clk	

Fitter Status	Successful - Fri Apr 11 11:35:26 2008
Quartus II Version	8.0 Internal Build 185 03/20/2008 SJ Full Version
Revision Name	FilterSystem
Top-level Entity Name	demo_firi_FilterSystem
Family	Stratix IV
Device	EP4SGX70DF29C2
Timing Models	Preliminary
Logic utilization	7 %
Combinational ALUTs	1,448 / 56,320 (3 %)
Memory ALUTs	770 / 28,160 (3 %)
Dedicated logic registers	3,825 / 56,320 (7 %)
Total registers	3825
Total pins	121 / 412 (29 %)
Total virtual pins	0
Total block memory bits	12,805 / 6,617,088 (< 1 %)
DSP block 18-bit elements	14 / 384 (4 %)
Total GXB Receiver Channels	0 / 16 (0 %)
Total GXB Transmitter Channels	0 / 16 (0 %)
Total PLLs	0 / 3 (0 %)
Total DLLs	0 / 4 (0 %)

Higher Level Synthesis



1



2

3

1. Covert the MDL schematic into an intermediate DFG representation

2. Apply transforms and analysis:

- Break apart carry chains
- DSP Block & Memory Timing
- Share multipliers
- Pipeline for:
  - required FMax performance
  - Balanced/matched delays
- ….

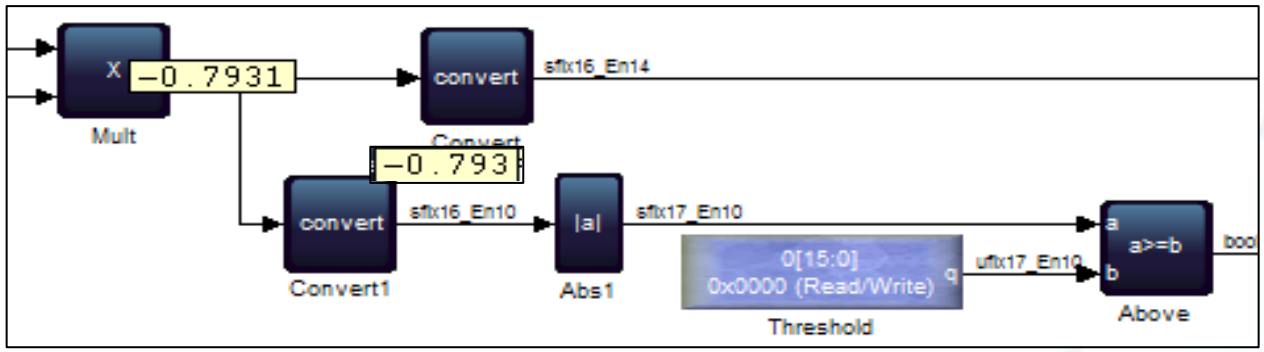
3. Generate RTL

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity DSPA is port (
SEL: in STD_LOGIC_VECTOR(15 downto 0);
C, D, B: in STD_LOGIC_VECTOR(15 downto 0);
A : out STD_LOGIC);
end;

architecture BEHAVIOUR of DSPA is
begin

A := B * C + D;
……

end;
```

ModelPrim: Zero Latency Blocks

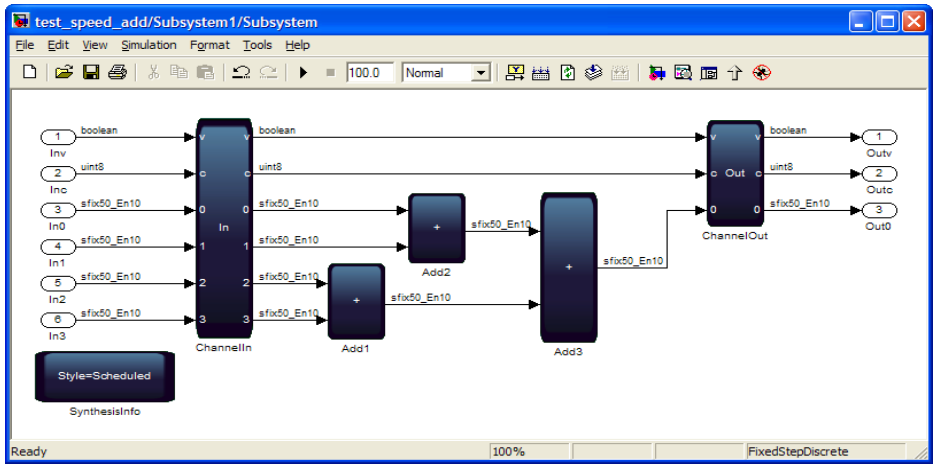
- Blocks are behavioural in nature
  - *What* to do, not *When* to do it
  - Focus on signal flow representation

- Much easier debug and modify without pipeline
- Design-once and retarget to different speed-grades and families



Behavioural input enables Optimizations

Performance Through Pipelining

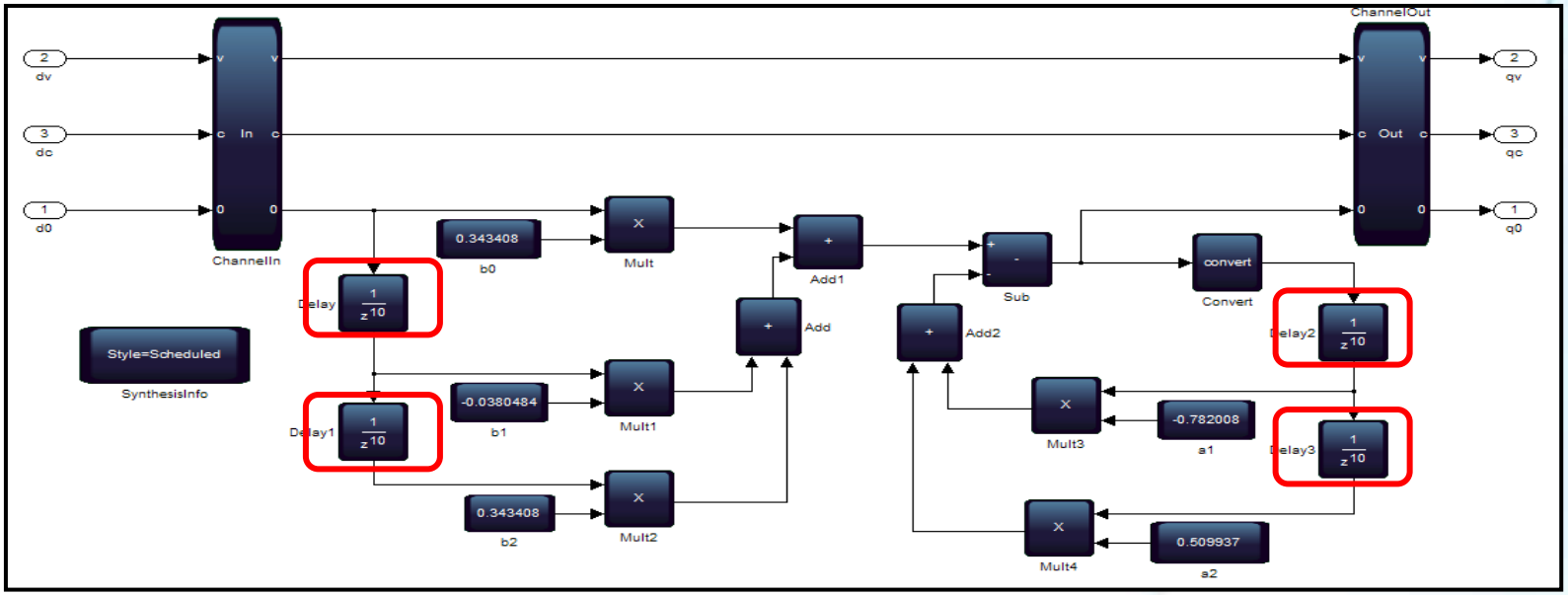


- Simply enter desired System Clock Frequency,
- No need to change model
- Simple 50-bit 4-input adder tree
  - 100 MHz Target => 118 LUT4s, 121 MHz, No pipeline
  - 200 MHz Target => 175 LUT4s, 286MHz, 1 stage pipeline
  - 400 MHz Target => 350 LUT4s, 581 MHz, 5 stage pipeline

Timing driven synthesis produces small or fast RTL from same model

Multi-channel Designs

- IIR example uses ‘textbook’ lumped delays
- Replace registers with number of channels
- Delays are distributed around logic to meet fmax goal
- Processes multiple channels simultaneously



Easy to enter Models produce high quality hardware

DSP Builder Advanced Blockset Design Examples

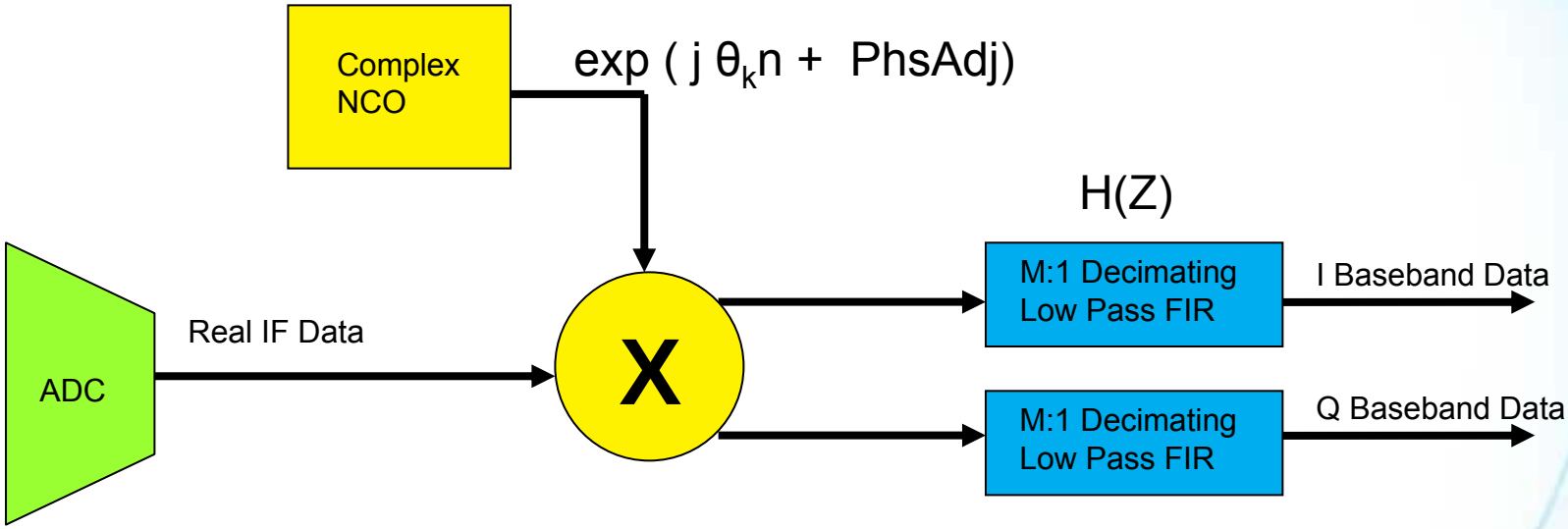
Design Examples

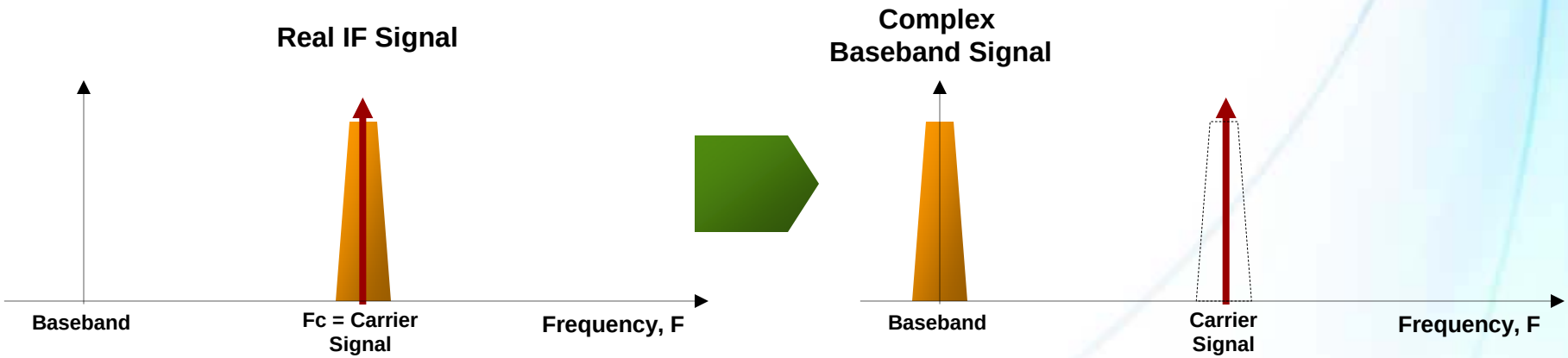
- Radar front end processor
  - Interfacing to high speed A/Ds
  - 2.8 Gsps A/D, efficient downconvert to 350 Msps
- Direct RF upconversion to multi-gigabit DACs
  - Interfacing to high speed DACs
  - 4.096 Gsps Digital Up Converter
- 8x8 Beam Former
  - Folding based upon clock rate

Design Examples

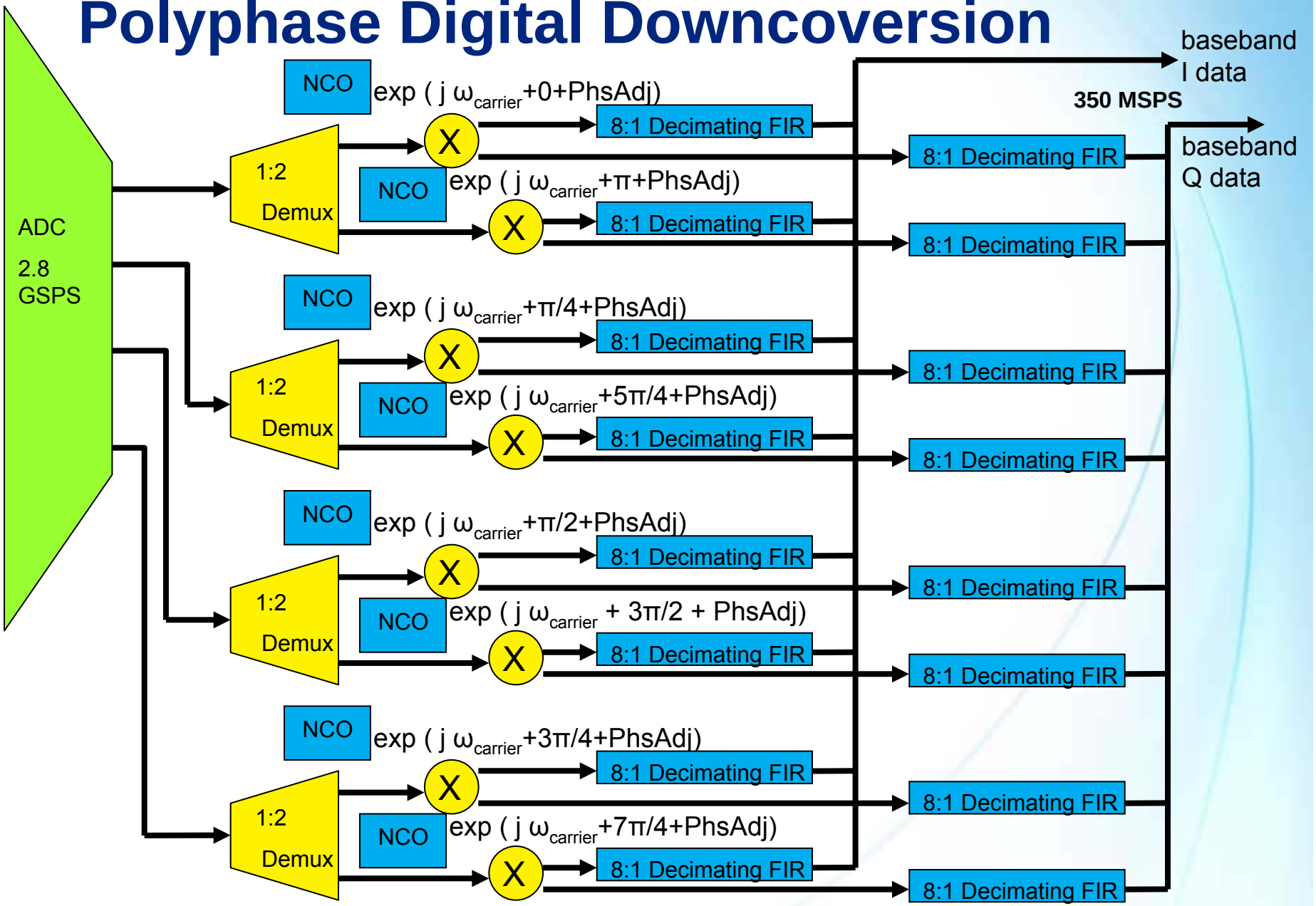
- Radar front end processor
  - Interfacing to high speed A/Ds
  - 2.8 Gsps A/D, efficient downconvert to 350 Msps
- Direct RF upconversion to multi-gigabit DACs
  - Interfacing to high speed DACs
  - 4.096 Gsps Digital Up Converter
- 8x8 Beam Former
  - Folding based upon clock rate

Traditional Digital Downconversion

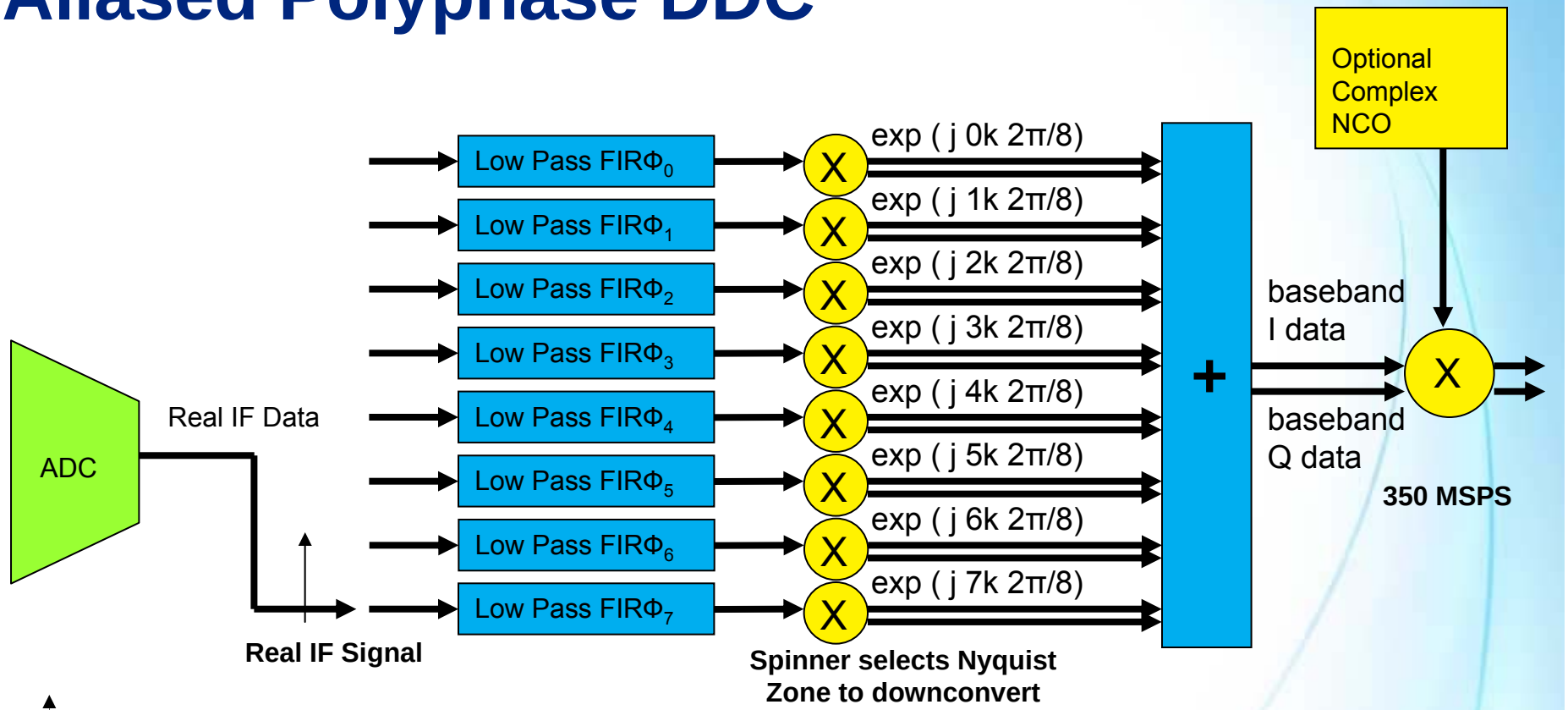


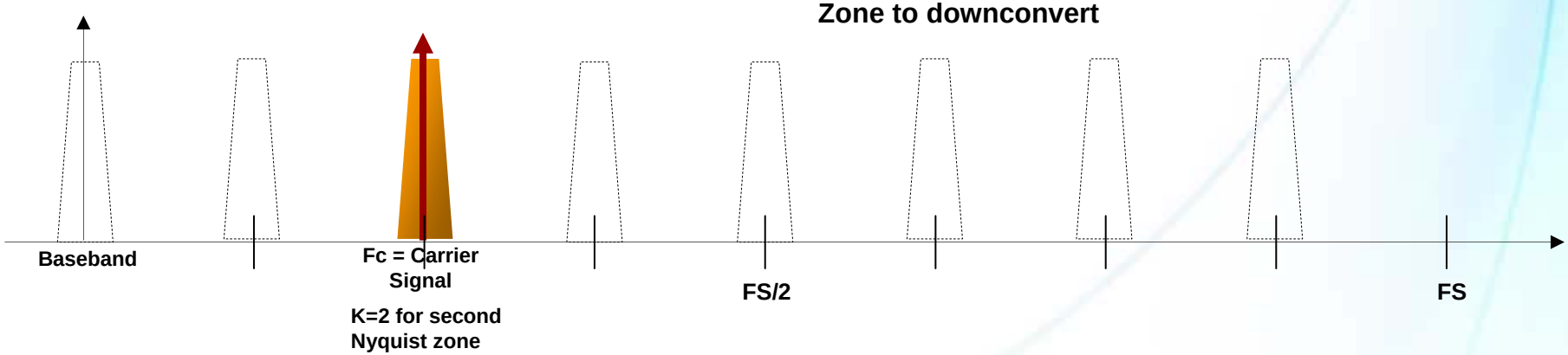


Polyphase Digital Downcoversion

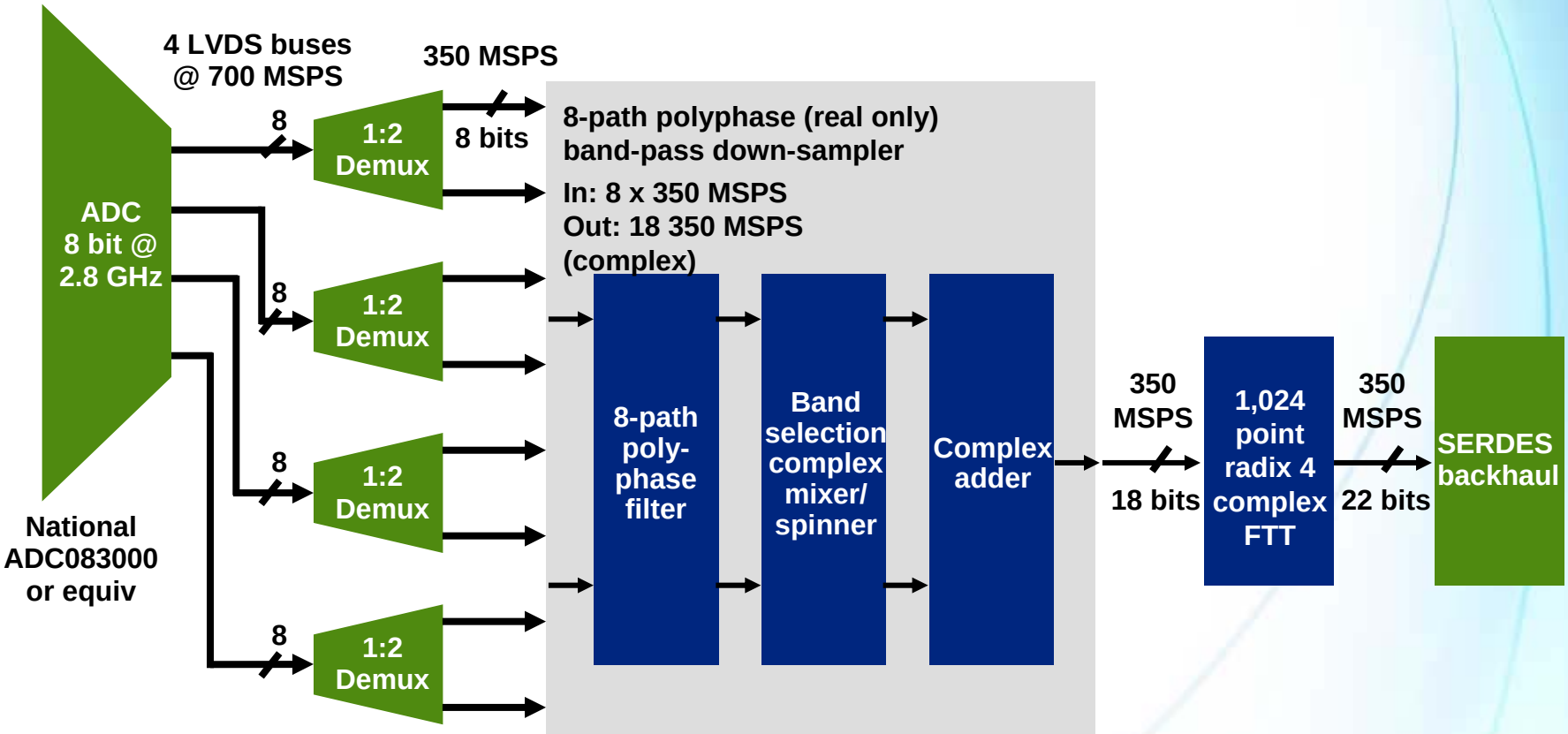


Aliased Polyphase DDC





Radar Front End



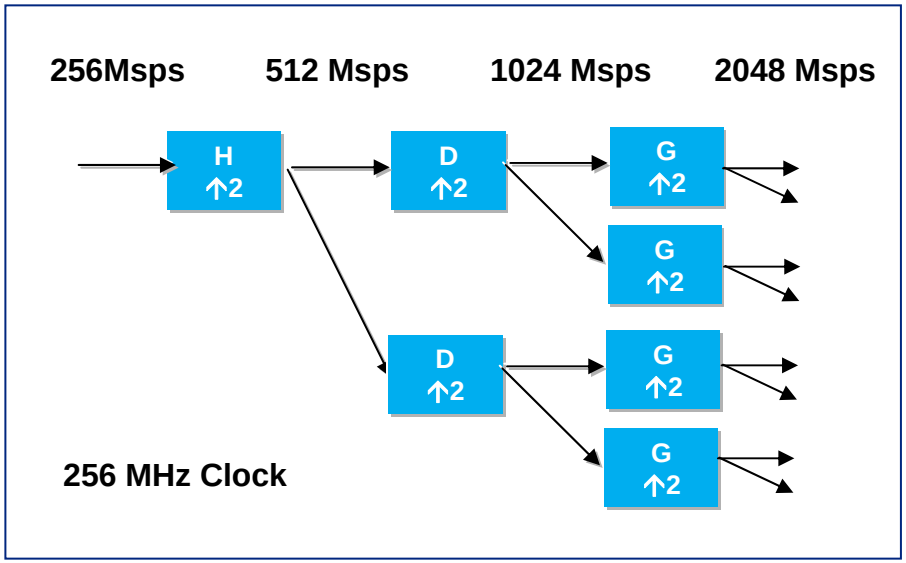
Implemented in DSPB-AB

Design Examples

- Radar front end processor
  - Interfacing to high speed A/Ds
  - 2.8 Gsps A/D, efficient downconvert to 350 Msps
- Direct RF upconversion to multi-gigabit DACs
  - Interfacing to high speed DACs
  - 4.096 Gsps Digital Up Converter
- 8x8 Beam Former
  - Folding based upon clock rate

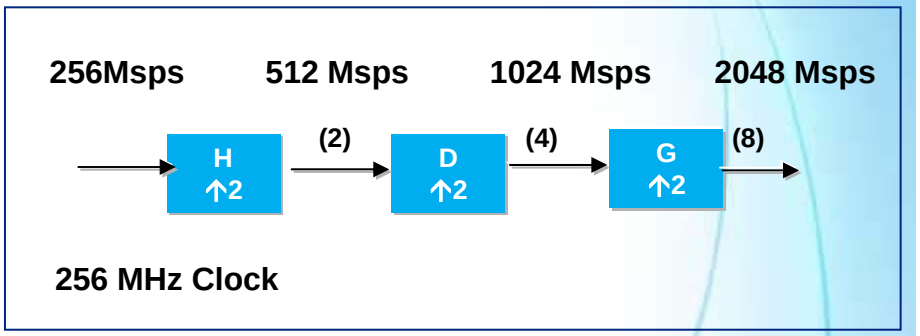
DirectRF: Data rate > Clock rate designs

Current Design Methodology



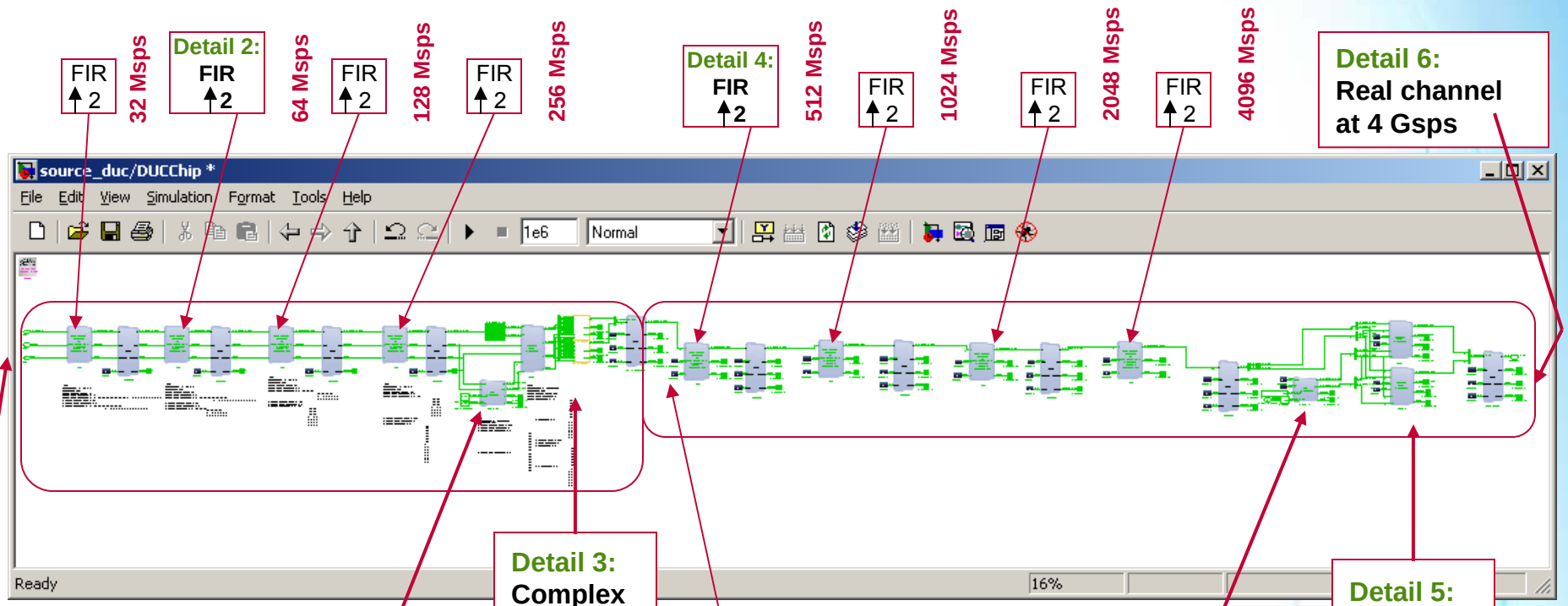
- Complex control required
- User has to manage reordering of Odd / Even phases of half band filters

Advanced Blockset Methodology



- Significantly simplifies high speed design
- The tool automatically duplicates necessary hardware to generate parallel outputs
- Handles polyphase reordering
- Design flow is simplified

Direct RF Design: Overview



Detail 1:
8 Channels of I&Q at 16 Msps Time Shared on the same bus at 256 MHz

Detail 3:
8 NCOs with 6 MHz channel spacing = 48 MHz band

Up converted 256 MHz IF (Separate I /Q) Eight 6 MHz Signals in 48 MHz Band

Detail 5:
Polyphase NCO: 16 Phases in parallel

DirectRF

eq2x4

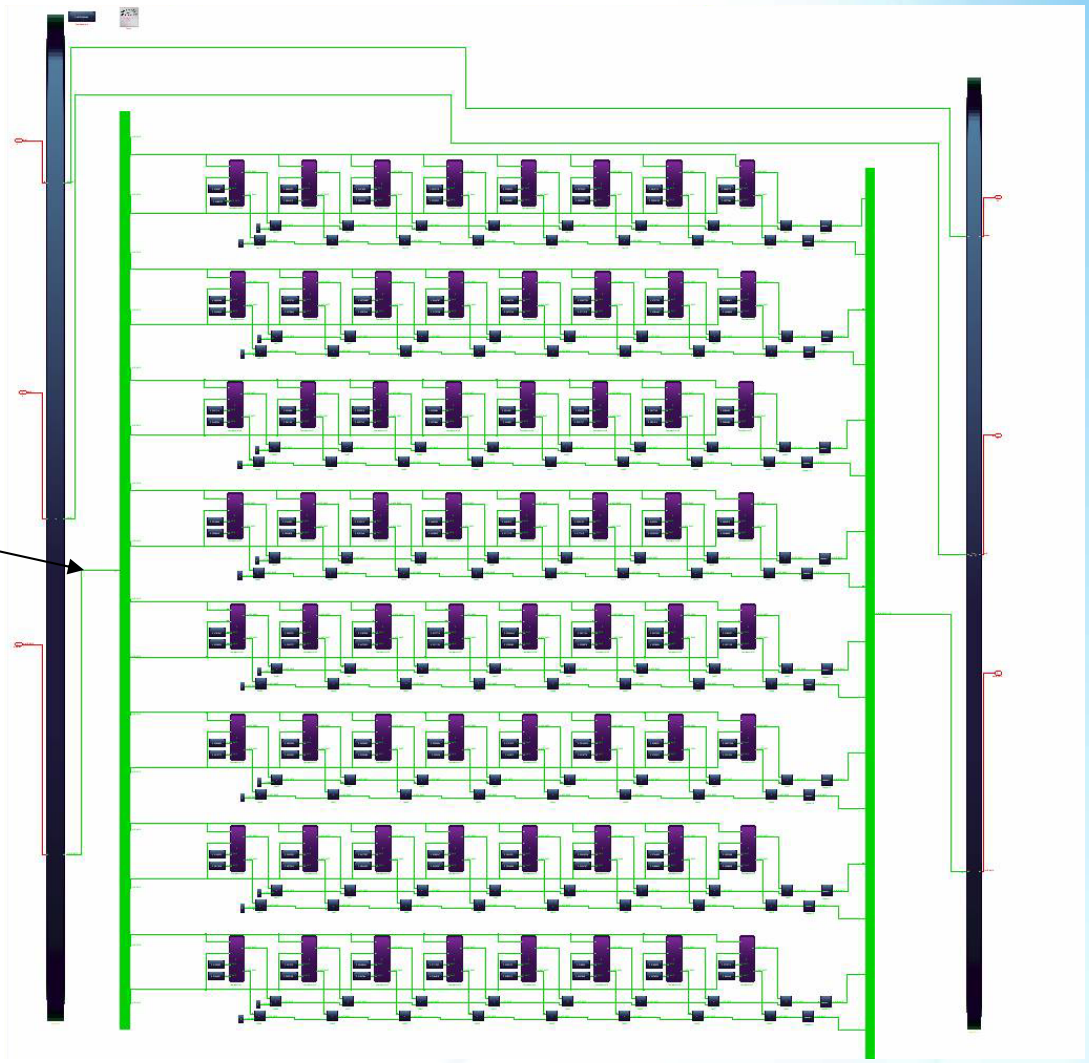
Design Examples

- Radar front end processor
  - Interfacing to high speed A/Ds
  - 2.8 Gsps A/D, efficient downconvert to 350 Msps
- Direct RF upconversion to multi-gigabit DACs
  - Interfacing to high speed DACs
  - 4.096 Gsps Digital Up Converter
- 8x8 Beam Former
  - Folding based upon clock rate

Folding: 8x8 Beam Former

- RADAR beam former
  - Multiply 8x1 incoming data vector by 8x8 matrix of weights
- Input data
  - 16-bit Complex data in rectangular form
  - Rate: 16-bits per sample @ 80MHz => 1280Mb/s
  - Format: I1,Q1,I2,Q2,…I8,Q8,I1,Q1,…
- Weights
  - 16-bit Complex weights
  - 8x8 matrix
- Output data
  - 16-bits per sample @ 80MHz
  - Format: I1,Q1,I2,Q2,…I8,Q8,I1,Q1,…

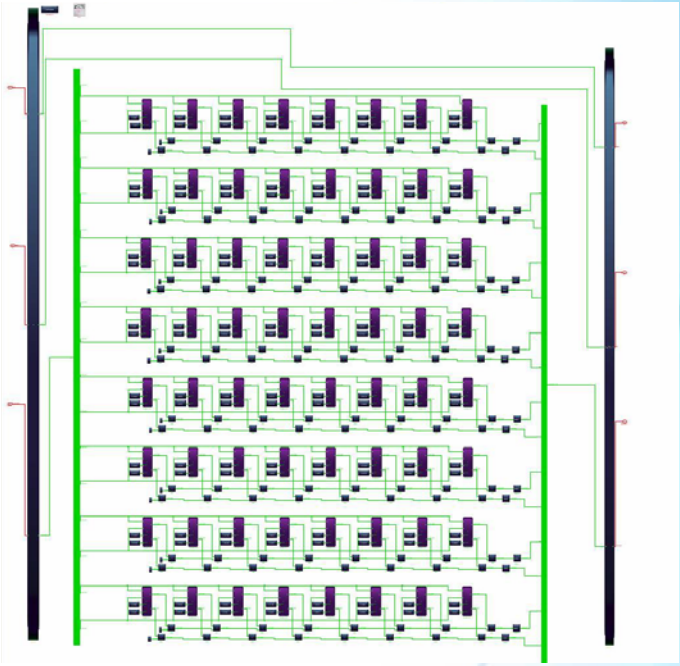
Real & Imaginary interleaved inputs/outputs on single bus



- 1 interleaved complex (I&Q) channel in and out

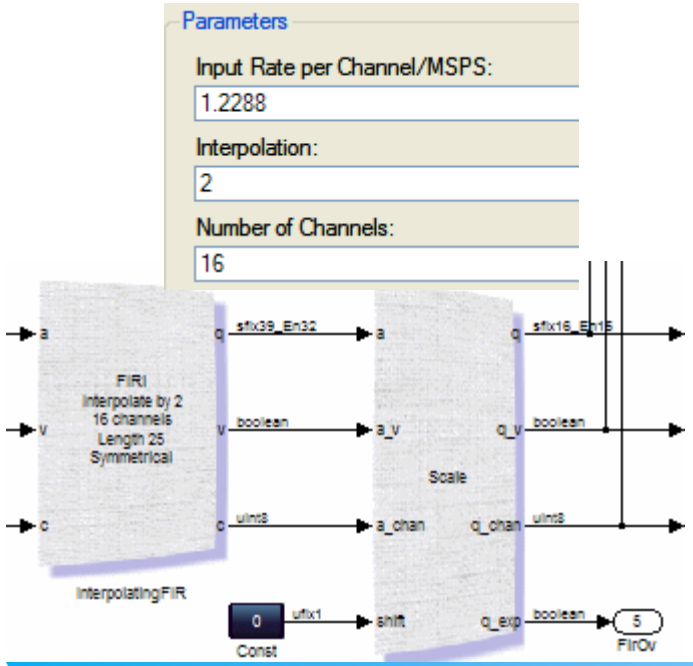
Advanced Blockset Folding

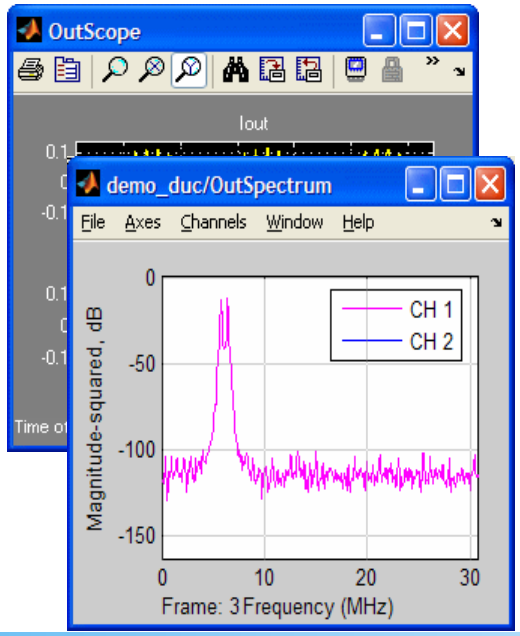
- Note that 64 Complex Multipliers are needed (or 256 multipliers)
- If clock rate = data rate then circuit requires 256 multipliers
- If clock rate > data rate then tool automatically time shares resources by putting samples into the memory and scheduling them accordingly

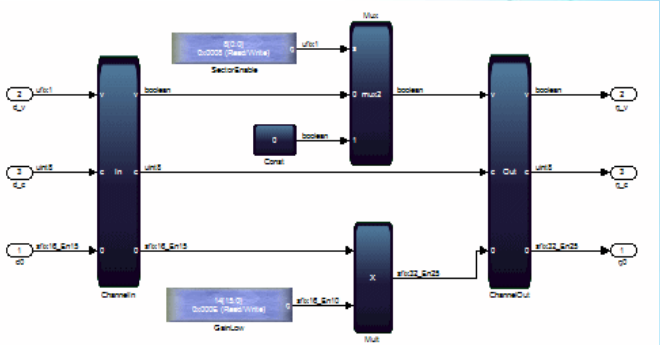


  - Example1 : Data Rate = 10 MHz, Clock = 200 MHz
    - Multipliers required = 256 multipliers / (200/10) or 25.6 multipliers (26 multipliers
  - Example2: Data Rate = 4 MHz, Clock = 256 MHz
    - Multipliers required = 256 multipliers / (256/4) or 4 multipliers

Summary: DSP Builder Advanced Blockset







Effortless FPGA Implementation

- Automatic pipelining to meet required Fmax
- Similar performance as optimized HDL
- Easy timing closure
- Fewer compile iterations

FAST Design Space Exploration

- Fast multi-channel design implementation
- Automatic generation of control plane logic
- Efficient pipelining for multi-channel data paths
- Ability to update design by editing system level parameters
- Effortless FPGA device family retargeting