# High-Level Design Tools for Floating Point FPGAs

Tom Czajkowski



© 2015 Altera Corporation—**Public**

# The Importance of Tools

- Algorithm developers are not necessarily hardware designers
- The usual flow is to create an algorithm in software and then use tools to implement such algorithm on a target platform
  - For FPGAs this does require actually generating the hardware for a given algorithm
  - This means that the quality of the tool used directly impacts the quality of results a designer obtains
- The key question is, if the obtained result does not meet the required performance or area metric, is there anything the tool offers to aid the designer?


© 2015 Altera Corporation—**Public**

# Altera's Approach

- Two tools provide access to algorithmic development for FPGAs, including floating point arithmetic

- OpenCL SDK
  - The tool converts OpenCL source program (C-based) into a circuit that implements the given algorithm in hardware
  - Automatically connects the core algorithm to PCIe/DDR interfaces to build a complete system that can be immediately programmed onto an FPGA without any user intervention
  - No hardware design expertise is required to get a functional design

- DSP Builder
  - A tool for algorithmic development of an IP
  - The output is a block that matches user specifications


© 2015 Altera Corporation—**Public**

# Altera OpenCL SDK (Highlights)

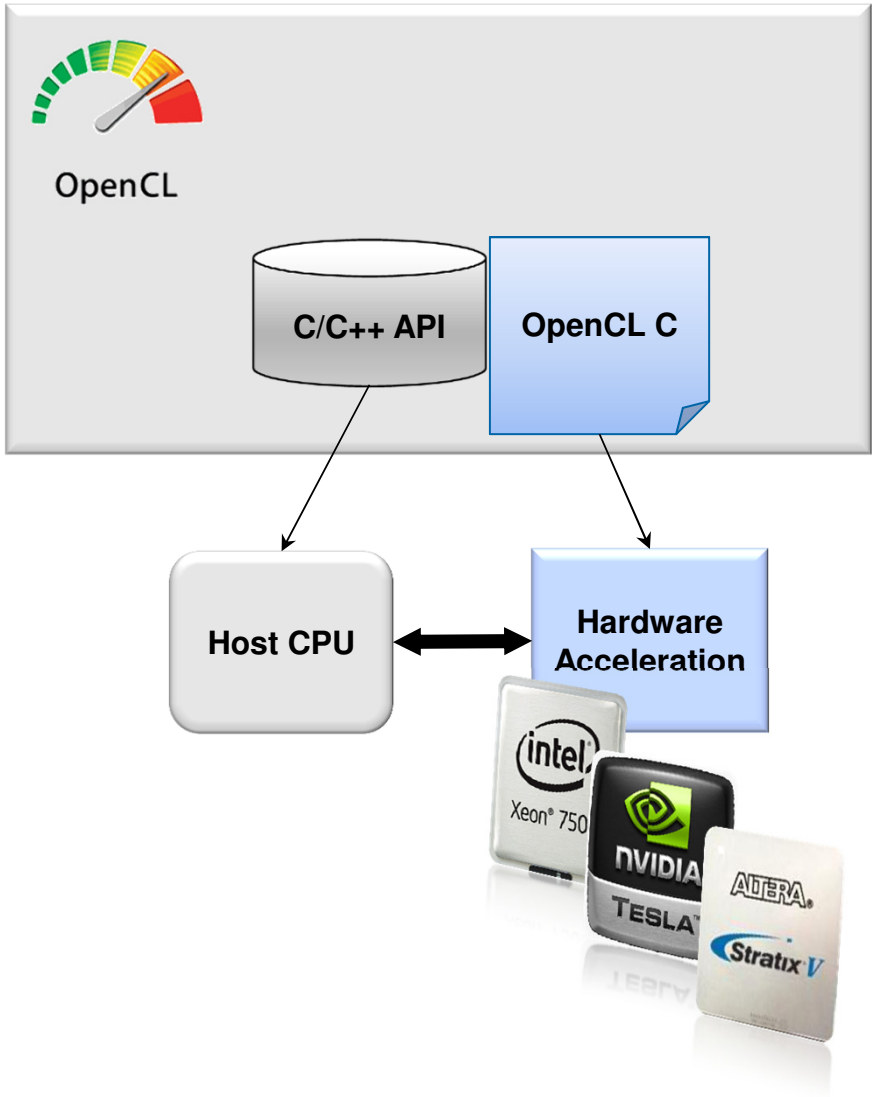




© 2015 Altera Corporation—**Public**

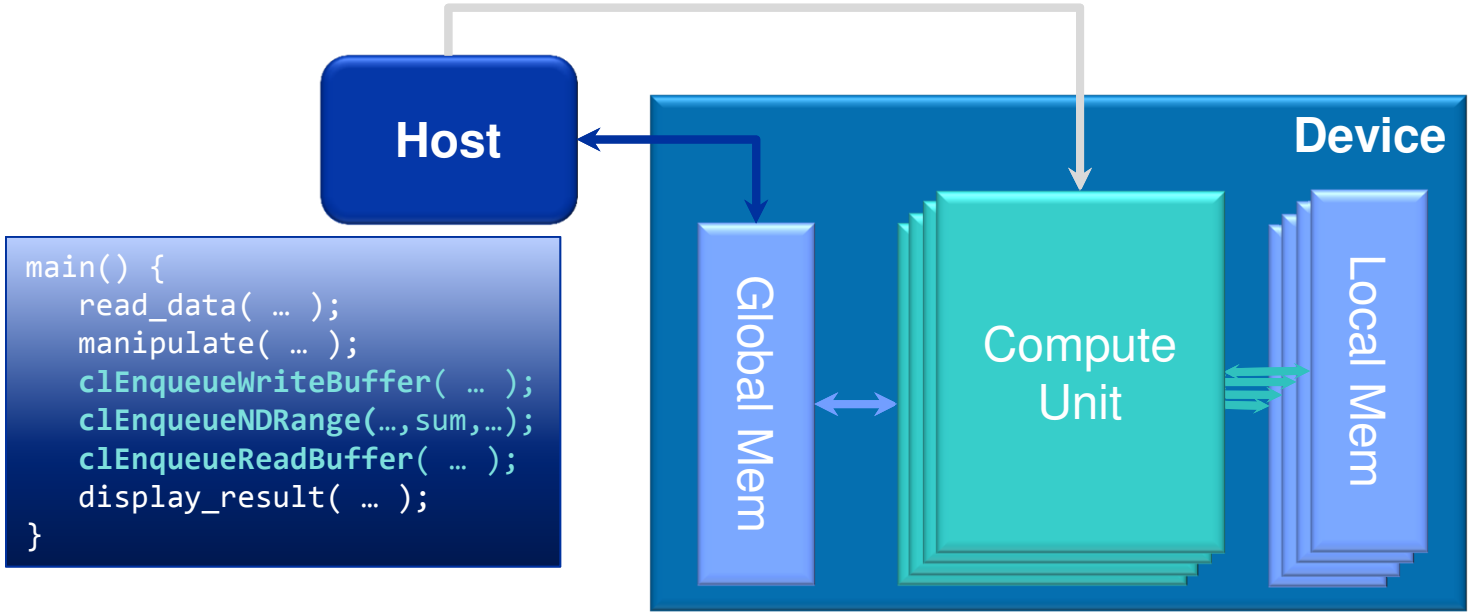
# OpenCL Overview

- Open Computing Language
  - Software-centric
    - C/C++ API for host program
    - OpenCL C (C99-based) for acceleration device
  - Unified design methodology
    - CPU offload
      - Memory Access
      - Parallelism
      - Vectorization




© 2015 Altera Corporation—Public

# OpenCL Abstract Programming Model




```
main() {
   read_data( … );
   manipulate( … );
   clEnqueueWriteBuffer( … );
   clEnqueueNDRange(…,sum,…);
   clEnqueueReadBuffer( … );
   display_result( … );
}
```

```
__kernel void
sum(__global float *a,
    __global float *b,
    __global float *y)
{
  int gid = get_global_id(0);
  y[gid] = a[gid] + b[gid];
}
```

- Explicit Data Storage
  - Hierarchical Memory Model
- Explicit Parallelism
  - Vectorization
  - Multi-threading

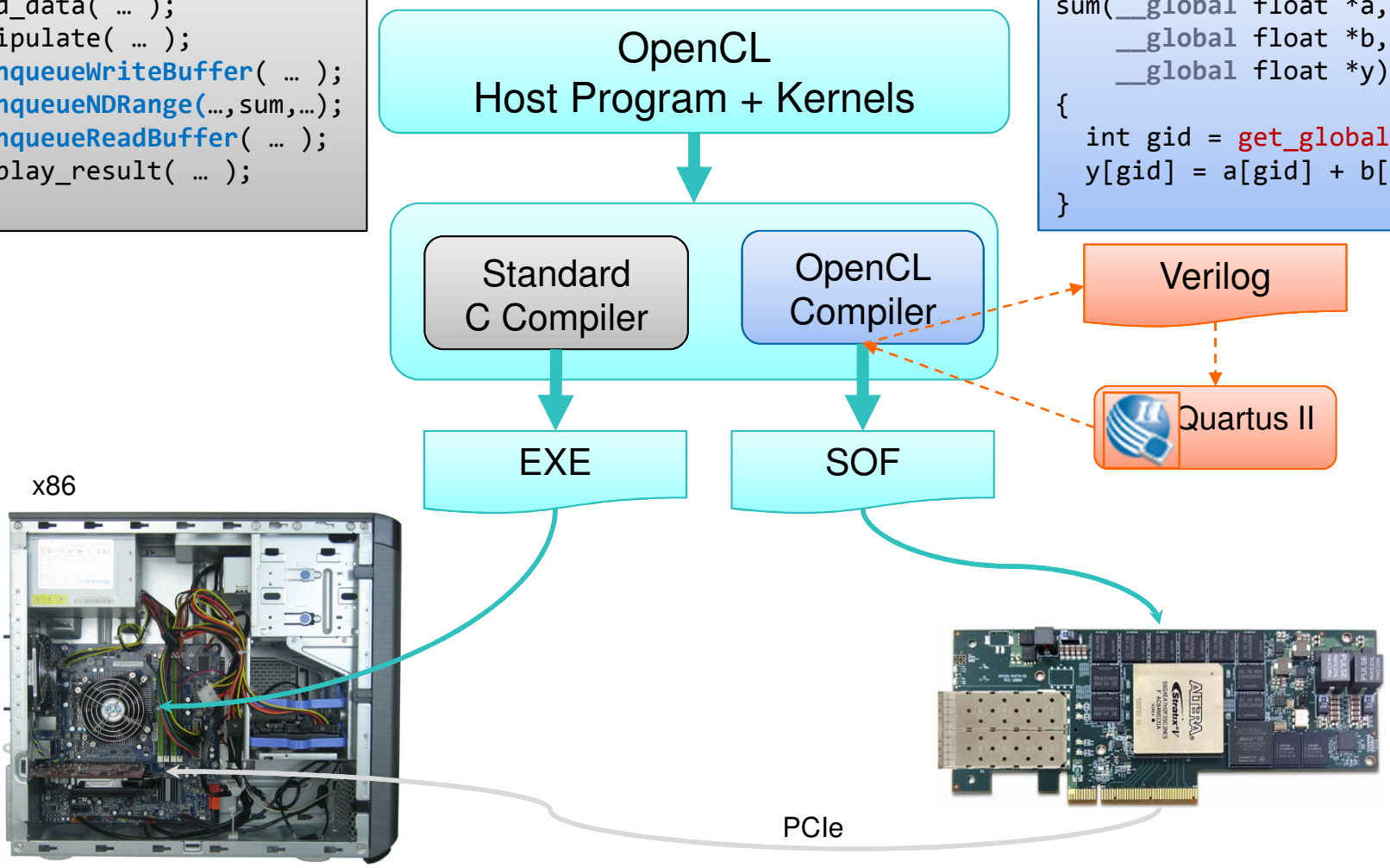
© 2015 Altera Corporation—Public

ALTERA.

# Flow

```
main() {
   read_data( … );
   manipulate( … );
   clEnqueueWriteBuffer( … );
   clEnqueueNDRange(…,sum,…);
   clEnqueueReadBuffer( … );
   display_result( … );
}
```

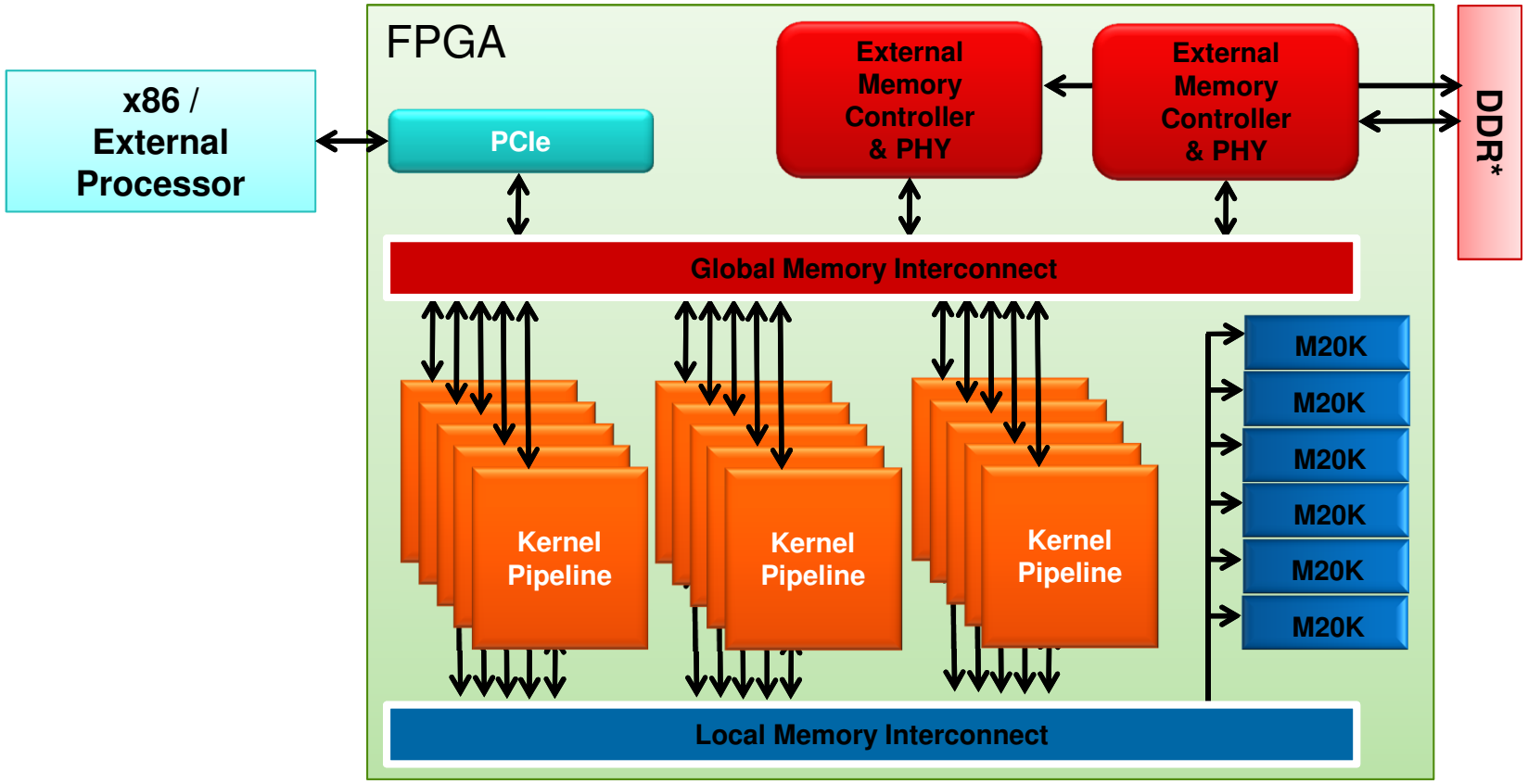
```
__kernel void
sum(__global float *a,
    __global float *b,
    __global float *y)
{
  int gid = get_global_id(0);
  y[gid] = a[gid] + b[gid];
}
```




© 2015 Altera Corporation—Public

ALTERA

# FPGA OpenCL Architecture




- Modest external memory bandwidth
- Extremely high internal memory bandwidth
- Highly customizable compute cores

© 2015 Altera Corporation—Public

# Compiling OpenCL to FPGAs

## Host Program

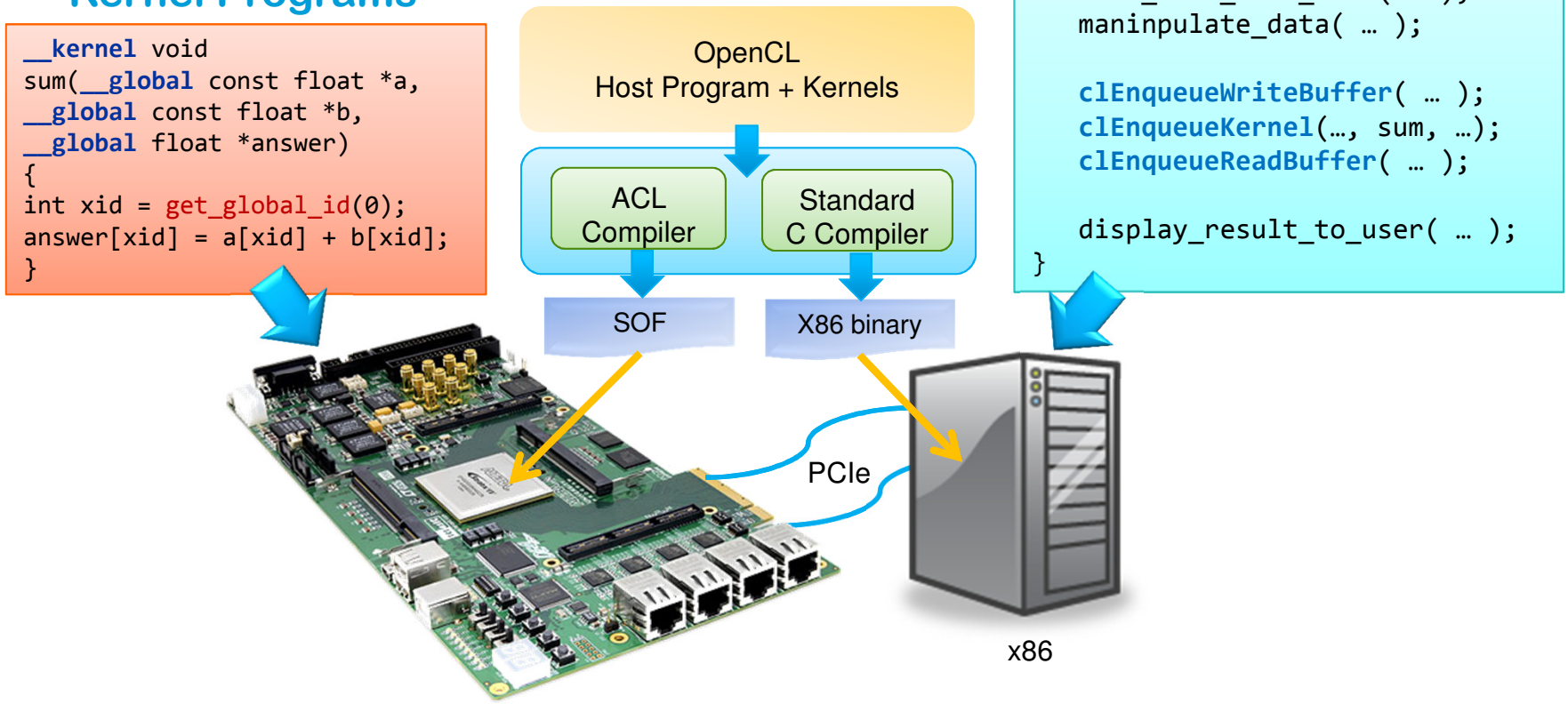
```
main()
{
   read_data_from_file( … );
   maninpulate_data( … );

   clEnqueueWriteBuffer( … );
   clEnqueueKernel(…, sum, …);
   clEnqueueReadBuffer( … );

   display_result_to_user( … );
}
```

## Kernel Programs

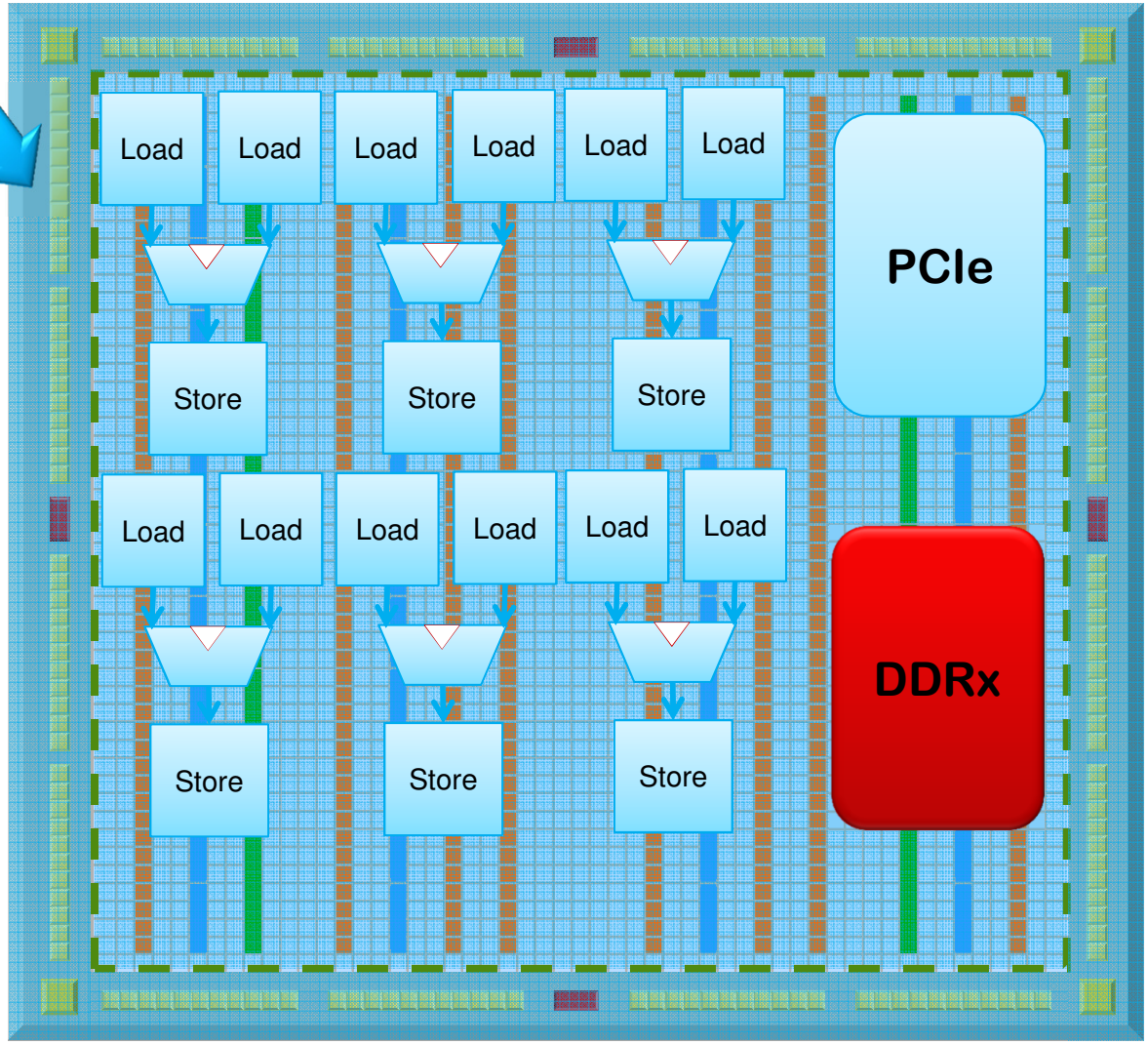
```
__kernel void
sum(__global const float *a,
__global const float *b,
__global float *answer)
{
int xid = get_global_id(0);
answer[xid] = a[xid] + b[xid];
}
```




© 2015 Altera Corporation—Public

# Compiling OpenCL to FPGAs

```
__kernel void
sum(__global const float *a,
__global const float *b,
__global float *answer)
{
int xid = get_global_id(0);
answer[xid] = a[xid] + b[xid];
}
```




© 2015 Altera Corporation—Public

ALTERA

# Floating-Point Support in OpenCL SDK





© 2015 Altera Corporation—**Public**

# Conformant SP FP Support

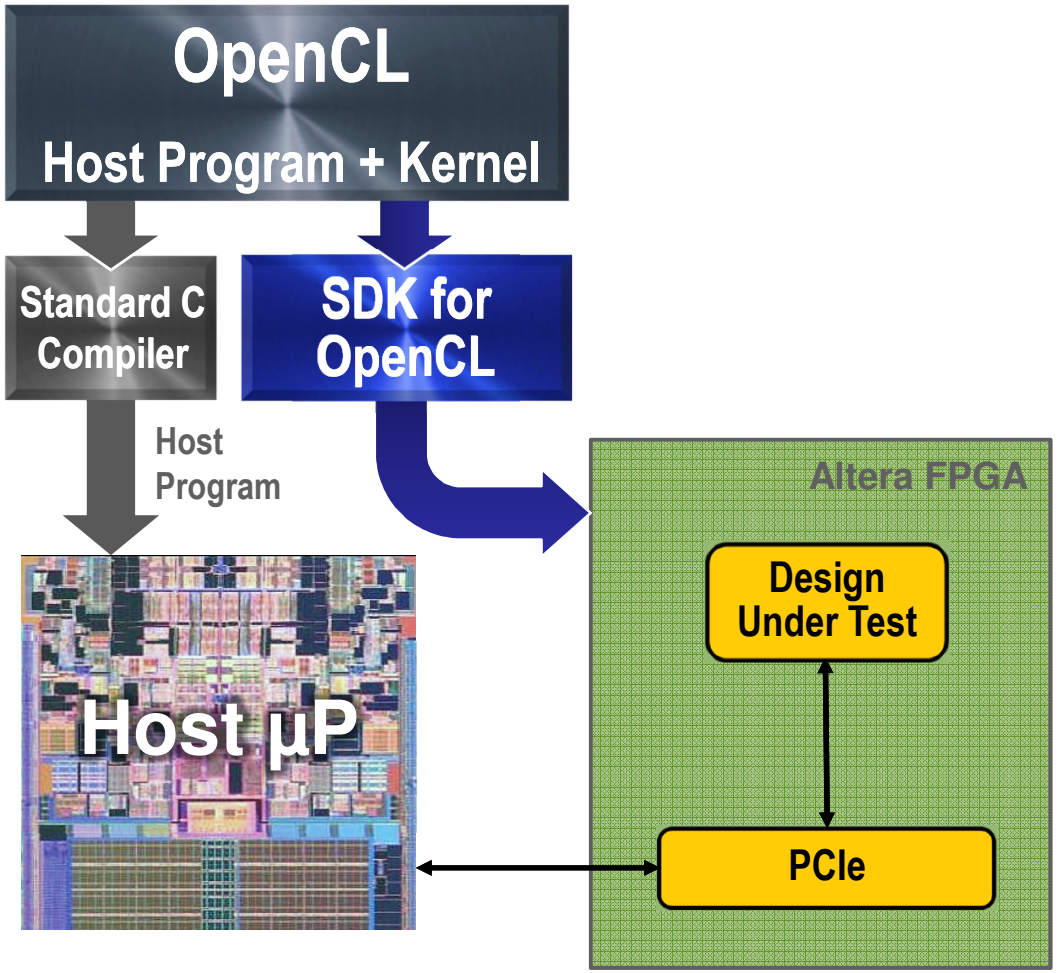



- OpenCL SDK features a complete single-precision FP support
- All operations are tested to be conformant within the specified ULP bound
- Use of a float type is sufficient to take advantage of this feature


© 2015 Altera Corporation—**Public**

# Verifying Conformance

- Each function is tested in hardware using ~20 Billion test vectors
- For single-input functions, this is an exhaustive test
  - For multi-input functions, this is a comprehensive test
- 94 FP functions
  - 1.88 Trillion test vectors to achieve FP conformance





© 2015 Altera Corporation—Public

# Additional Features

- The compiler enables the user to optimize FP operations to improve the quality of results

- --fp-relaxed
  - This flag enables the compiler to reorder FP operations to create a better implementation. It needs to be supplied by a user because it has an impact on the results produced by a given algorithm

- --fpc
  - This flag enables the user to inform the tool that the data being processed is both finite and the rounding mode can be round-to-zero, instead of round-to-nearest-even
    - This saves a significant amount of area in many applications

- Hardened Single-Precision FP operators
  - When Arria 10 is the target FPGA for an OpenCL design, hardened FP operators are used automatically – conformant to OpenCL standard





© 2015 Altera Corporation—Public

# Optimization Examples

- Fast Fourier Transform
- Matrix Multiplication

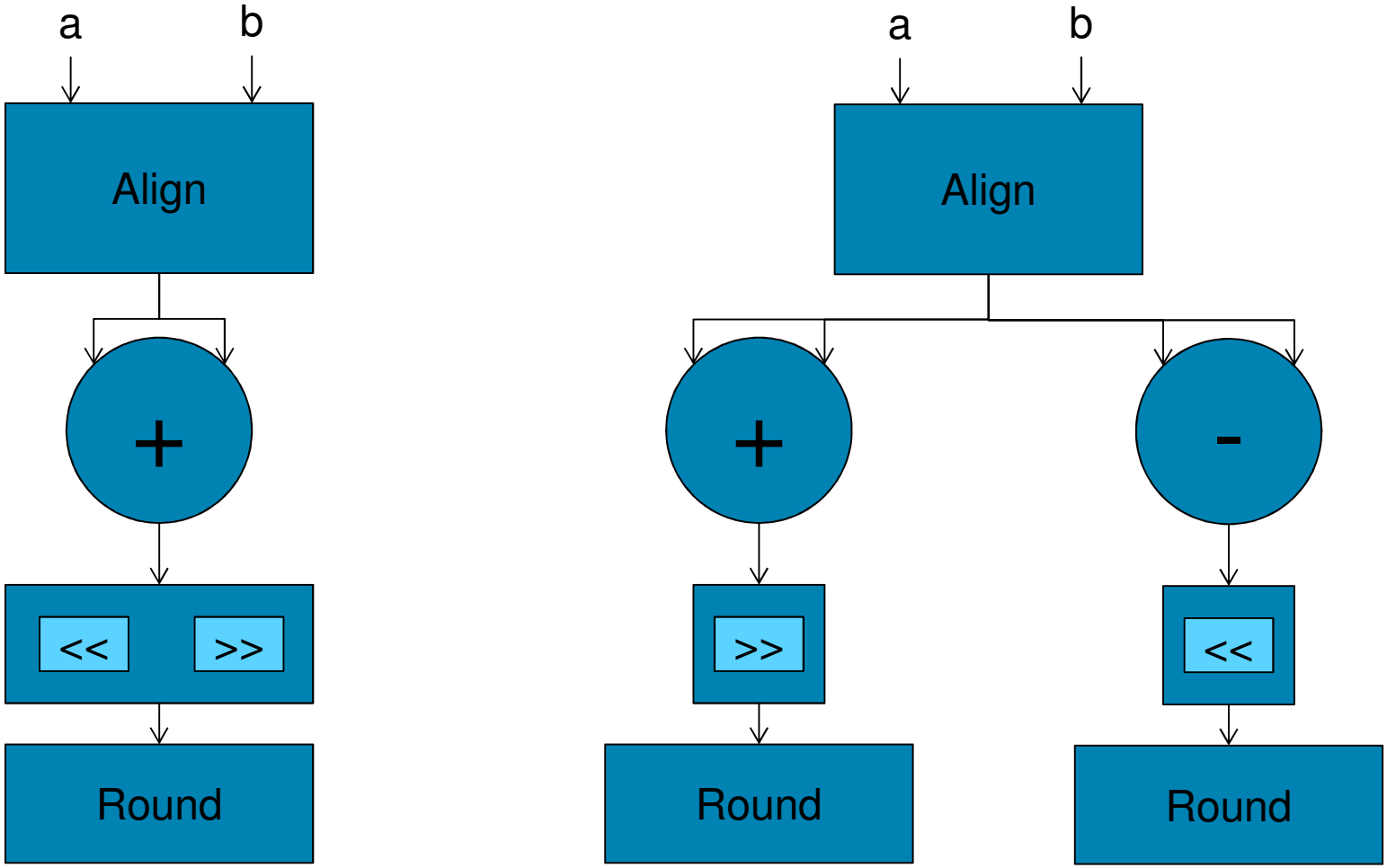

© 2015 Altera Corporation—**Public**

# FFT

- A classical signal processing application
- Frequently implemented using fixed-point arithmetic
  - This is ok for some applications, but not for all
- Key operations
  - add,
  - subtract
  - multiply


© 2015 Altera Corporation—Public

# Fused Add-Sub operation



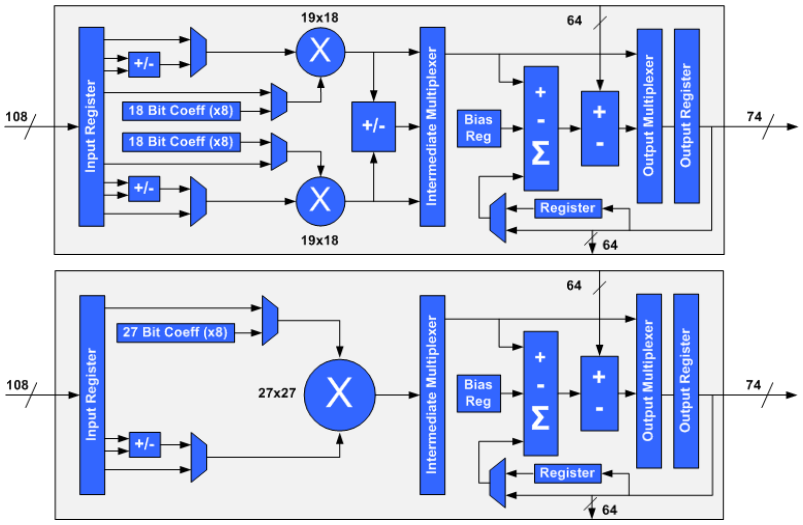


© 2015 Altera Corporation—Public

# Arria 10 Hardened FP DSP Blocks

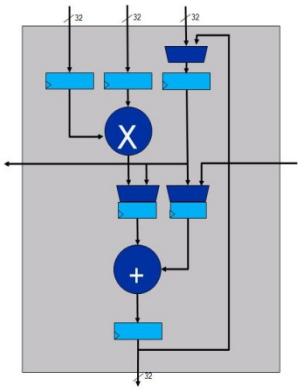
- Easily make tradeoffs during design cycle
  - Same DSP Block support Variable Precision integer math as well as single-precision floating point
  - **Precision flexibility throughout the DSP datapath**

***Variable Precision Fixed Point***




***Single-Precision Floating Point***

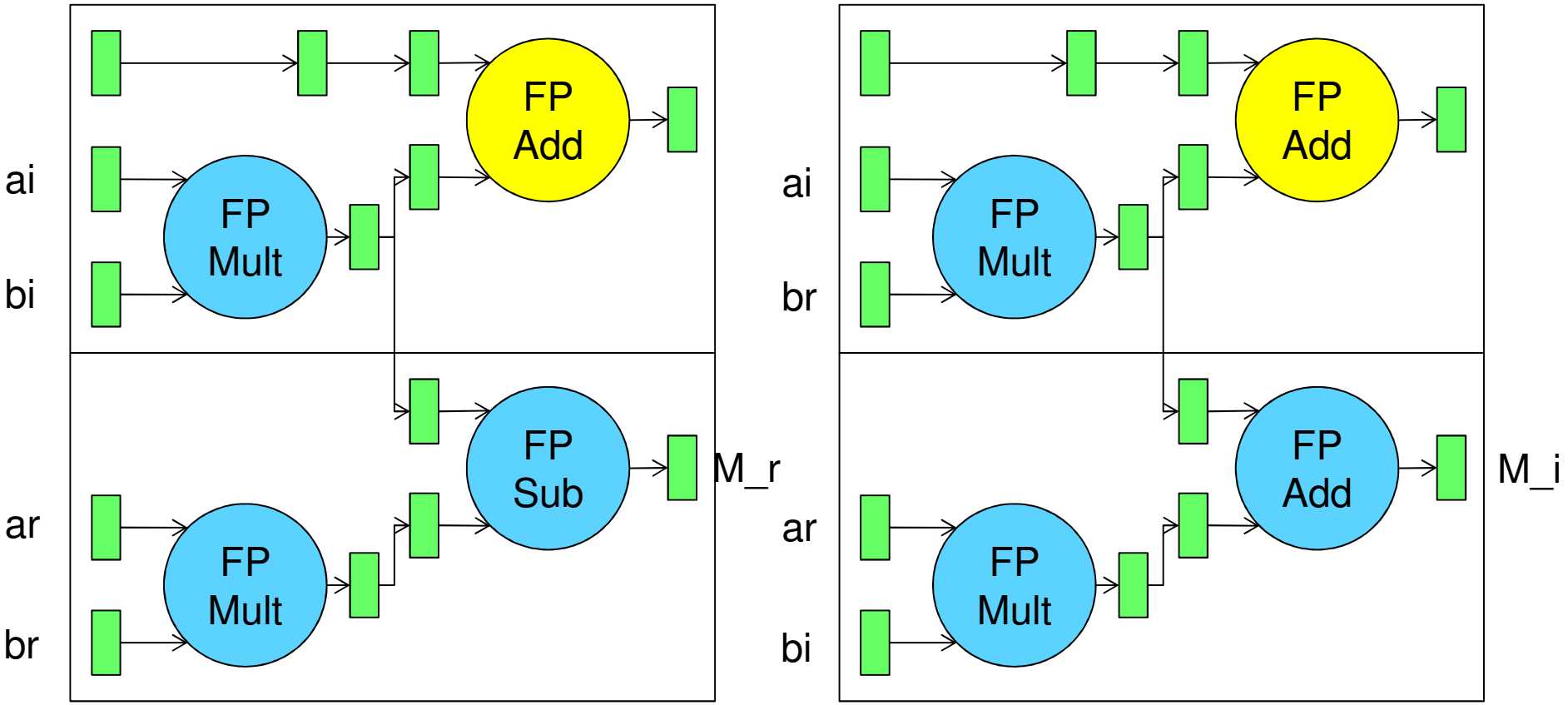
**A x B**
**A + C**
**A - C**
**A x B + C**
**A x B – C**
**Acc = A x B + Acc**
**Vector dot Products**
Complex Multiply, FFT Butterfly





© 2015 Altera Corporation—Public

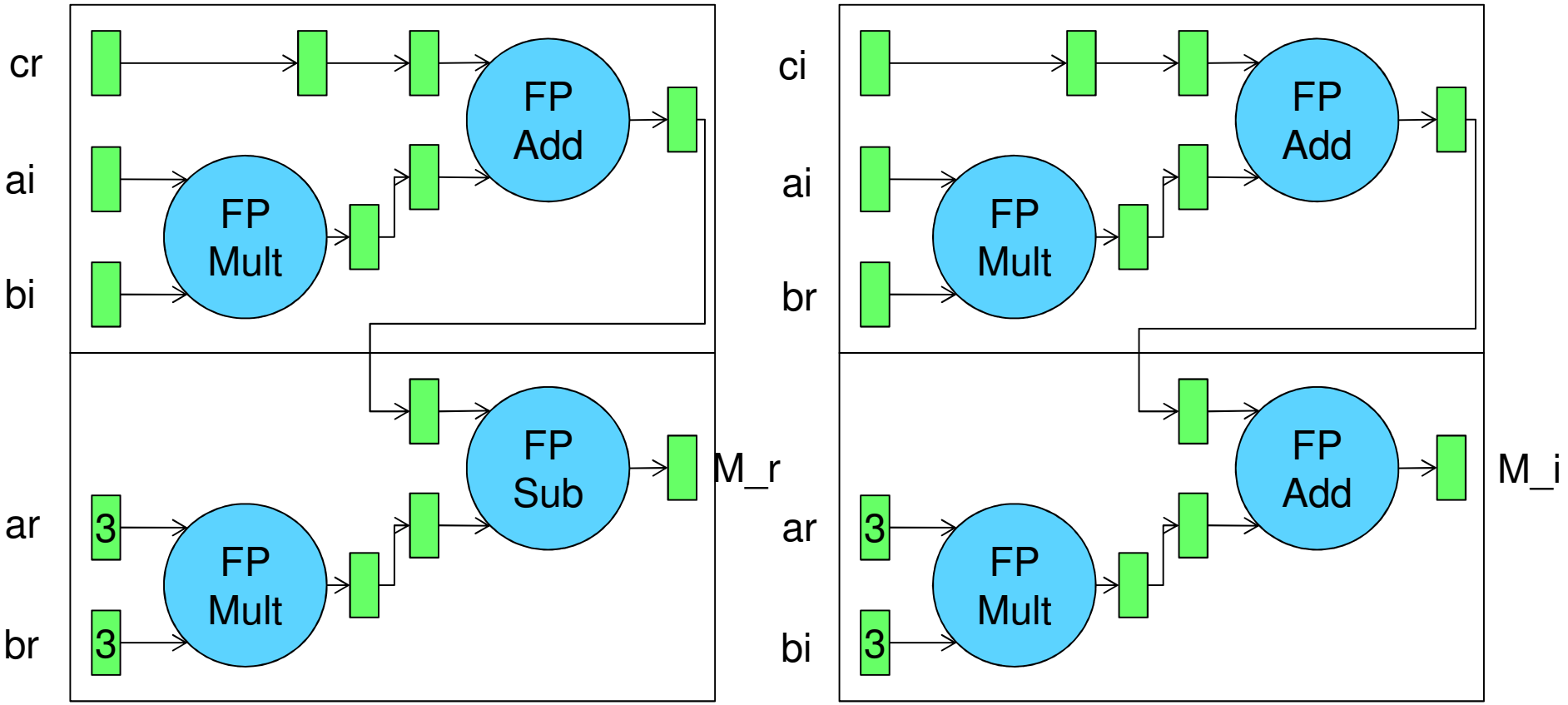
# Implementing Complex Math with Hardened FP





© 2015 Altera Corporation—Public

# Complex Multiplication and Addition





© 2015 Altera Corporation—Public

# Results

| Optimization | ALMs | DSPs |
|---|---|---|
| IEEE754 Conformant | 62126 | 60 |
| FPC | 39662 | 60 |
| Fused-Add-Sub + FPC | 34102 | 60 |
| Hardened FP | 6208 | 98 |

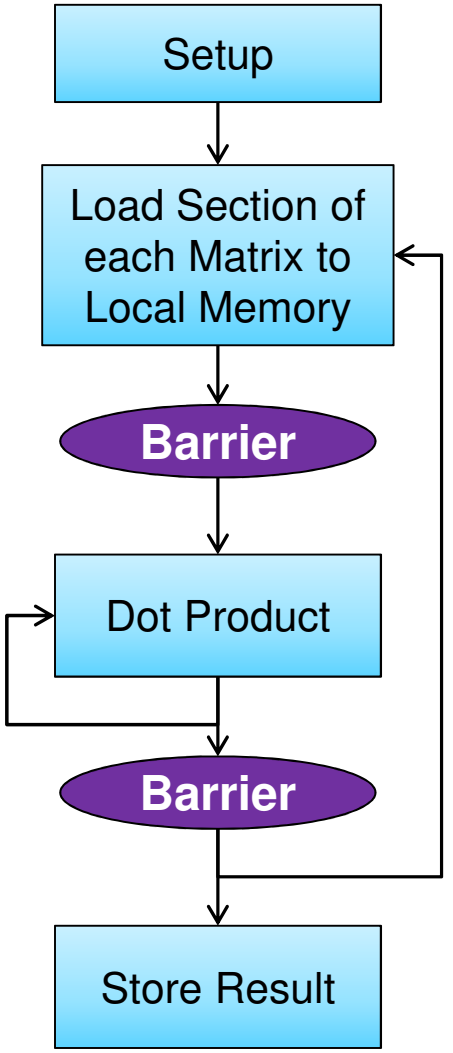

© 2015 Altera Corporation—Public

# Matrix Multiplication Source Code

```
#define BLOCK_SIZE 4
#define AS(i, j) A_local [j + i * BLOCK_SIZE ]
#define BS(i, j) B_local [j + i * BLOCK_SIZE ]

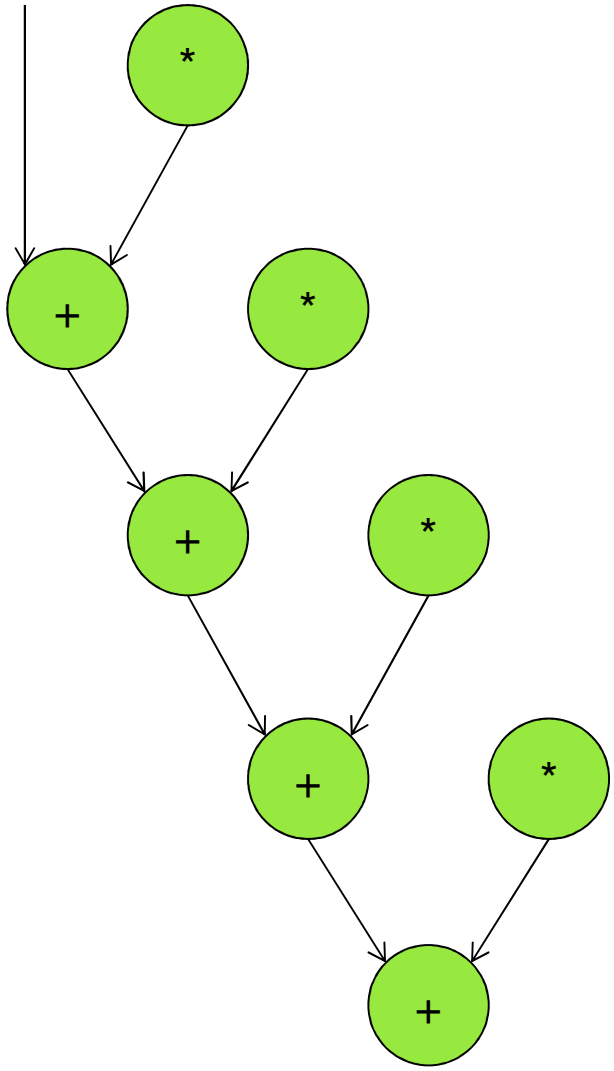
__kernel  void mm( __global float *restrict C, __global float *A, __global float *B,
                   __local float *restrict A_local, __local float *restrict B_local)
{
   int bx = get_group_id(0);
   int by = get_group_id(1);
   int tx = get_local_id(0);
   int ty = get_local_id(1);

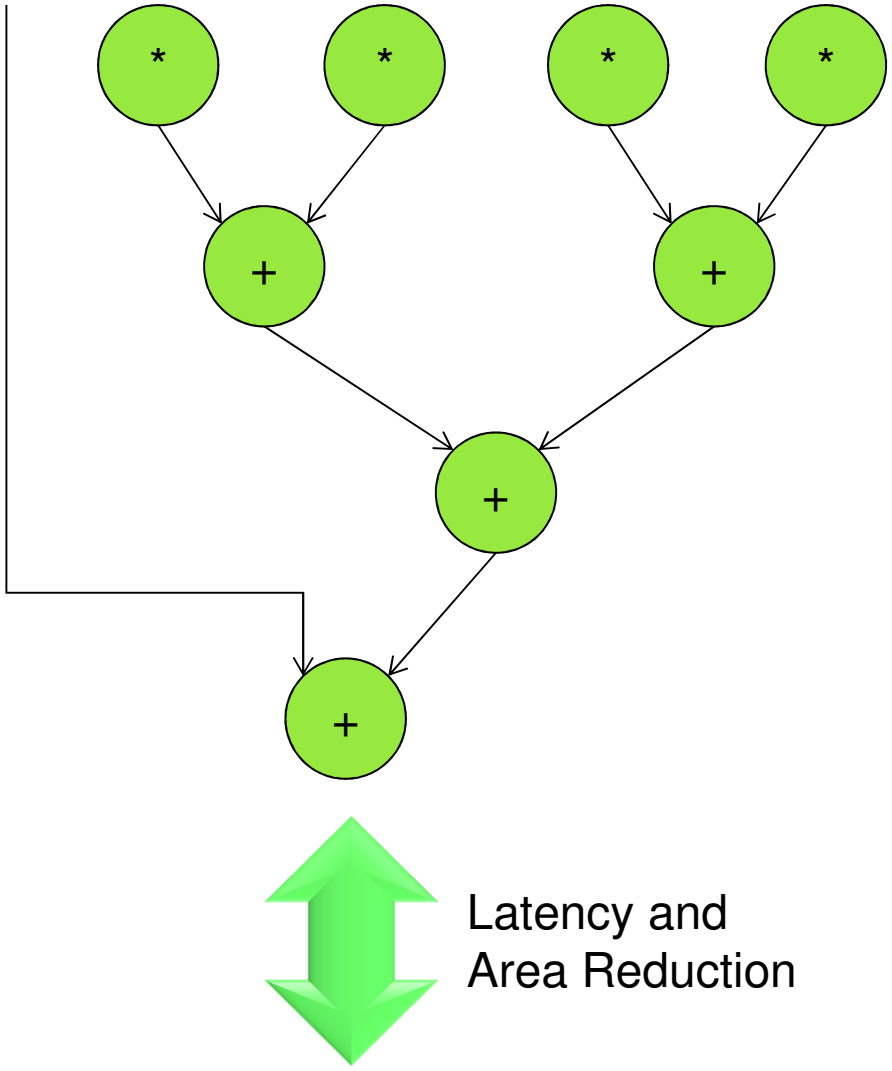
   float sum = 0.0f;
   for (int i = 1024* BLOCK_SIZE * by, j = BLOCK_SIZE * bx;
         i <= 1024* BLOCK_SIZE * (1+by)  - 1;
         i += BLOCK_SIZE, j += BLOCK_SIZE*1024) {
      AS(ty, tx) = A[i + 1024* ty + tx];
      BS(ty, tx) = B[j + 1024* ty + tx];
      barrier(CLK_LOCAL_MEM_FENCE);
        #pragma unroll
        for (int k = 0; k < BLOCK_SIZE; ++k) {
            sum += AS(ty, k) * BS(k, tx);
        }
      barrier(CLK_LOCAL_MEM_FENCE);
   }
   C[get_global_id(1) * get_global_size(0) + get_global_id(0)] = sum;
}
```




© 2015 Altera Corporation—Public

# What does --fp-relaxed do?




Latency and
Area Reduction


© 2015 Altera Corporation—**Public**

# Does --fpc do anything special for Matrix Multiplication?

- In addition to the regular benefits, --fpc flow recognizes adder trees as special constructs that can be further simplified

- Using the fact that one adder directly feeds another in a tree structure, enables us to reduce the adder-tree area by an additional ~27%


© 2015 Altera Corporation—**Public**

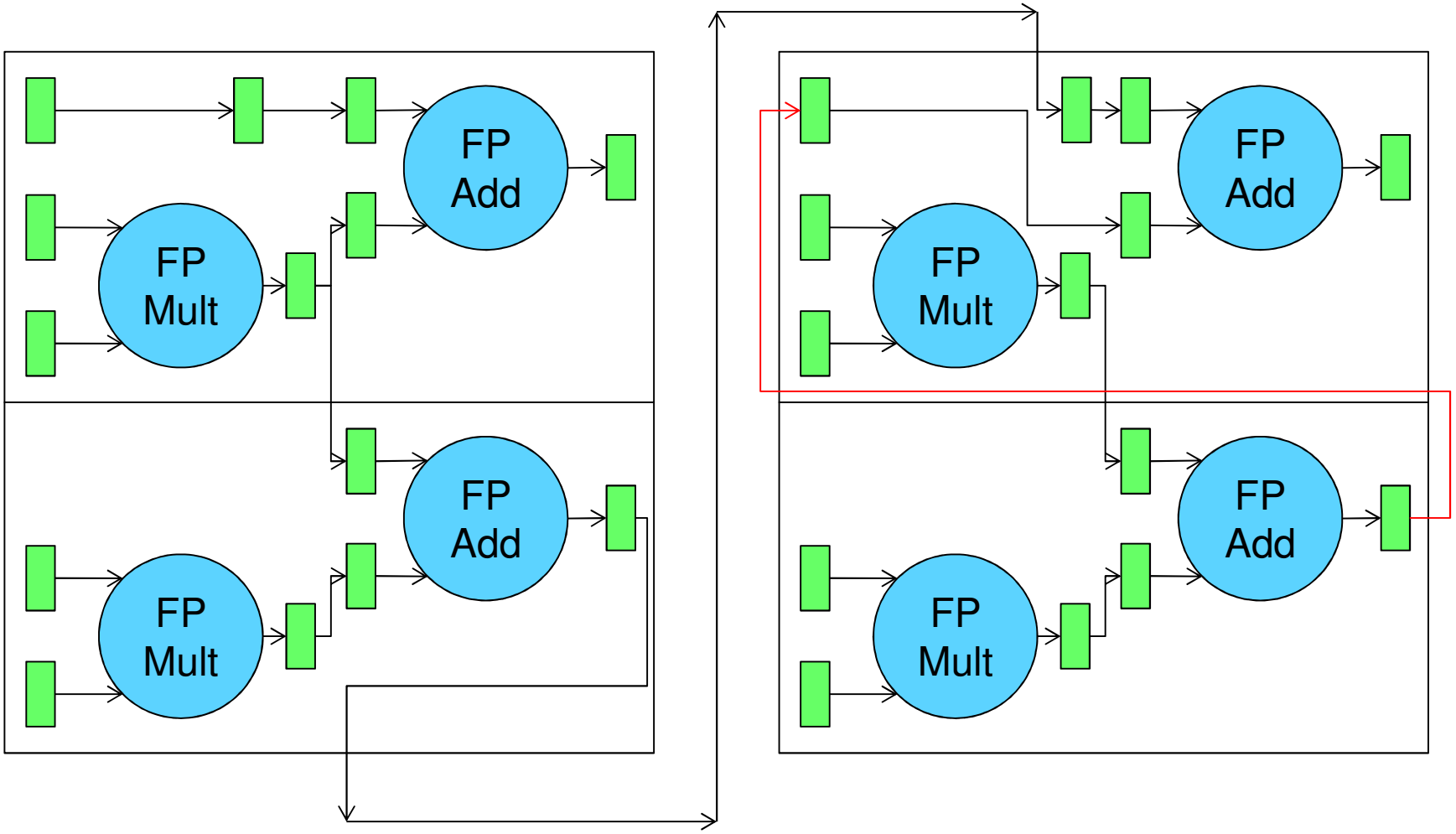
# How does Matrix Multiplication use Hardened FP Blocks?

- In matrix multiplication we can create balanced dot products that can be implemented using vector modes in Arria 10 Hardened FP DSP Block

- The dot products are entirely contained within the DSP blocks


© 2015 Altera Corporation—**Public**

# DOT Product Example





© 2015 Altera Corporation—Public

# Results

| Configuration | ALMs | DSPs |
|---|---|---|
| BLOCK=128, V=8, FPC | 315061 | 1034 |
| BLOCK=128, V=8, HFP | 61293 | 1034 |





© 2015 Altera Corporation—Public

# DSP Builder (Highlights)

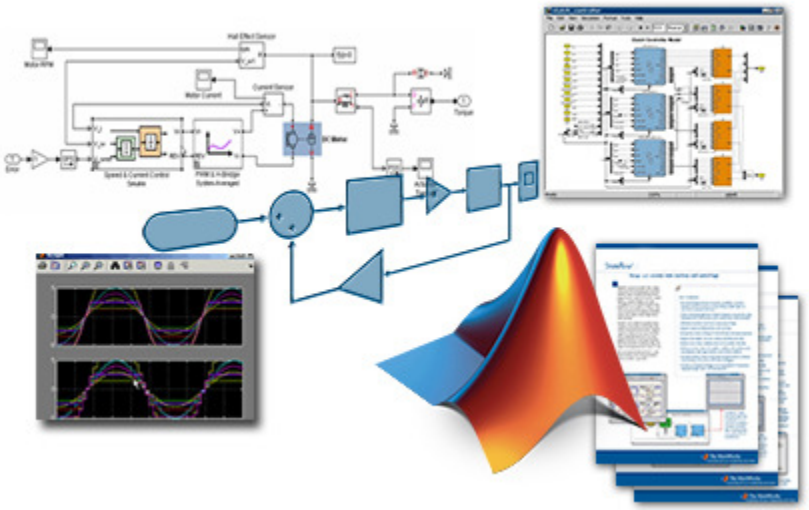




© 2015 Altera Corporation—**Public**

# Simulink® - Simulation and Model-Based Design

- Dynamic graphical modeling environment
- Simulink®
  - Dynamically **develop** entire systems
  - **Simulate** and interact with the system
  - **Explore** different architectures
  - **Analyze** results
- Libraries of predefined blocks
  - **Core Libraries**: Sources, Sinks, Continuous, Math, etc.
  - **Domain-specific Libraries**: DSP System Toolbox, etc.
  - **Altera Additions**
    - *Altera DSP Builder Advanced Blockset*
    - *Altera DSP Builder Standard Blockset*



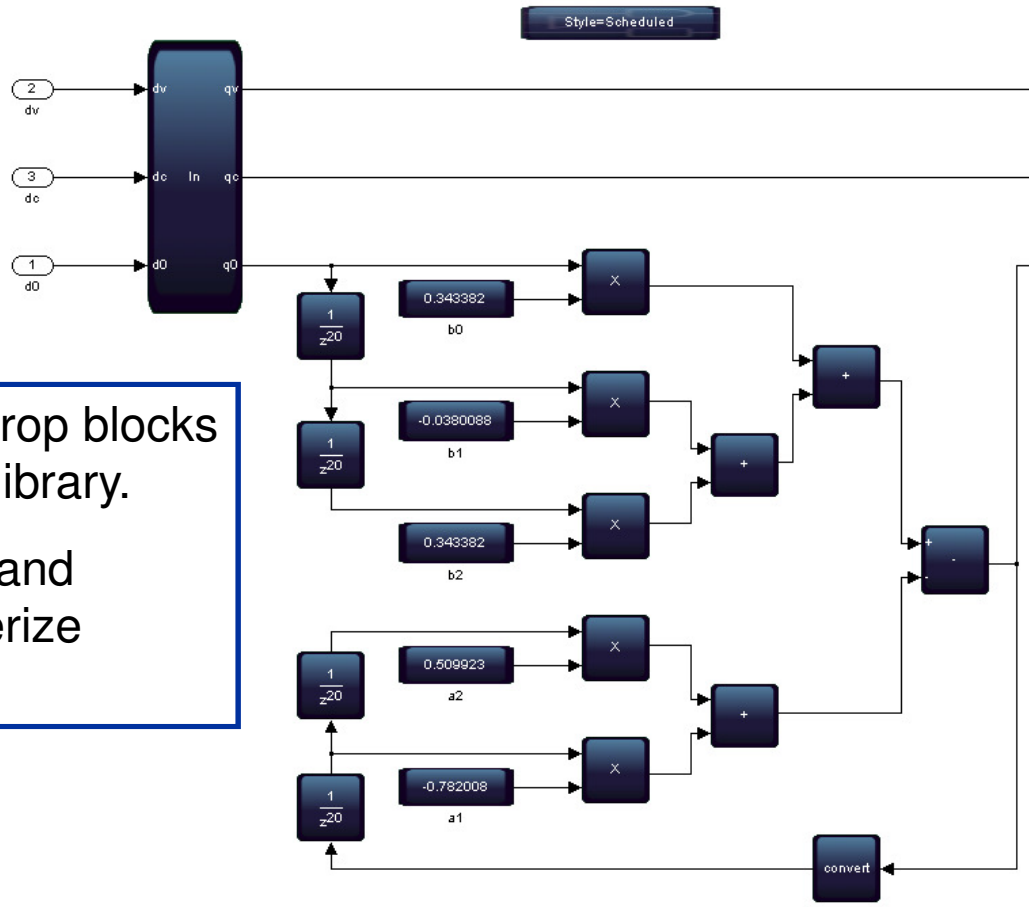

© 2015 Altera Corporation—**Public**

# DSP Builder Advanced

- **Schematic Entry**
  - Draw your algorithm, simulate & debug in Simulink
- **Constraint-driven design**
  - Set desired data rate, clock frequency & device
- **Automated pipelining**
  - Timing closure at high clock rates of 300-400+ MHz
- **Models silicon speeds**
  - Different devices families & speed grades result in different HDL
- **Automated resource sharing**
  - Marshall data though limited resources on consecutive clock cycles
- **High performance floating-point design**
  - Synthesize floating-point optimized HDL


© 2015 Altera Corporation—**Public**

# DSP Builder Schematic Entry




Drag & Drop blocks from the library.

Connect and parameterize

- Valid
- Channel
- Data

Protocol

Enables synchronisation without cycle counting.

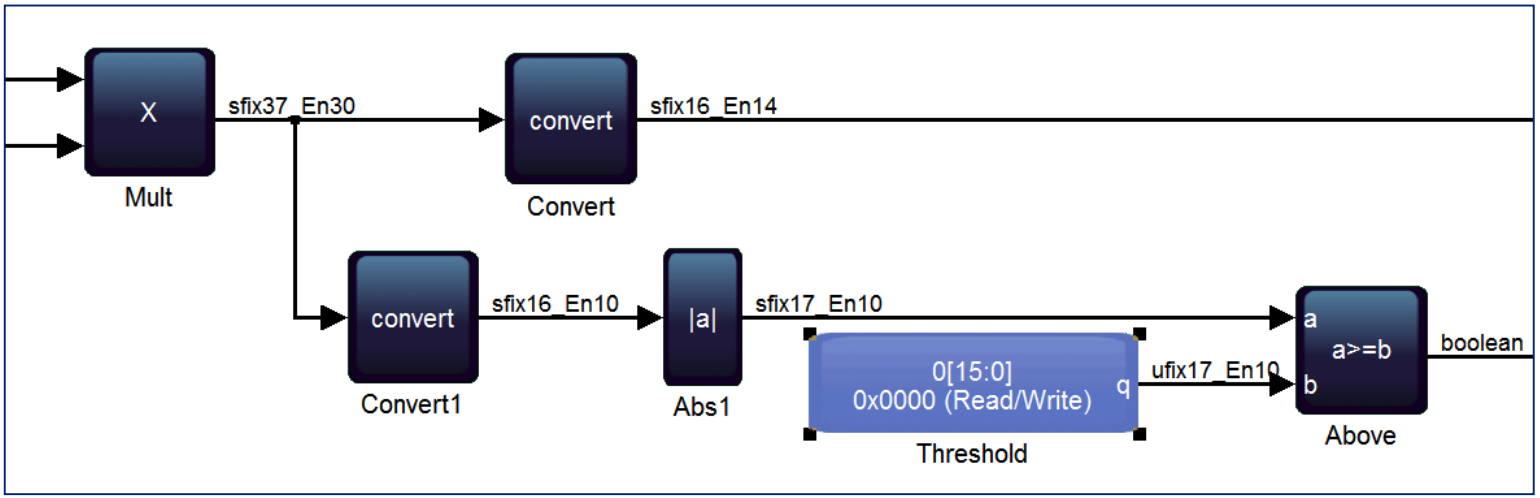
Data-type propagation simplifies schematic design. Blocks have simple type propagation rules.

The picture shows such a schematic for an infinite impulse response (IIR) filter.


© 2015 Altera Corporation—Public

# 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 modification without pipeline



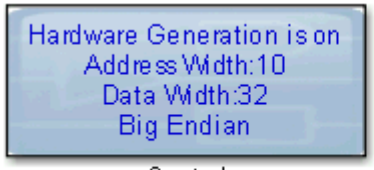

***Behavioural input enables automatic pipelining and optimization***


© 2015 Altera Corporation—**Public**

# Constraint-driven design

- Specify target device or device family and speed grade
  - Design remains generic
- Specify target clock frequency
- Specify Avalon-MM characteristics
- Locally guide pipelining effort and multiplier and memory implementation, if necessary




Control




Signals

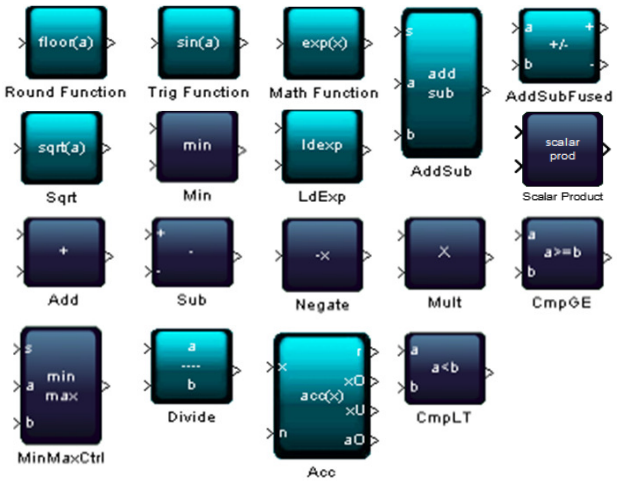



LocalThreshold


© 2015 Altera Corporation—**Public**

# Elementary math functions supporting floating point

- Coverage of ~70 elementary math functions
- Patented & published efficient mapping to FPGA hardware
  - Polynomial approximation, Horner's method, truncated multipliers, …
- Compliant to OpenCL & IEEE754 accuracy standards
- Rounding mode options for fundamental operators
- Half- to Double-precision




Fixed and floating point

Floating point only

**Trigonometrics of pi*x**

FPSinPiX
FPCosPiX
FPTanPiX
FPCotPiX

**Exp, Log and Power**

FPLn
FPLn1px
FPLog10
FPLog2
FPExp
FPExpFPC
FPExpM1
FPExp2
FPExp10
FPPowr

**Trig with argument reduction**

FPSinX
FPCosX
FPSinCosX
FPTanX
FPCotX

**Inverse trigonometric functions**

FPArcsinX\
FPArcsinPi
FPArccosX
FPArccosPi
FPArctanX
FPArctanPi
FPArctan2

**Conversion**

FXPToFP
FPToFXP
FPToFXPExpert
FPToFXPFused
FPToFP

**Macro Operators**

FPFusedHorner
FPFusedHornerExpert
FPFusedHornerMulti
FPFusedMultiFunction

**Trigonometrics misc**

FPHypot
FPRangeReduction

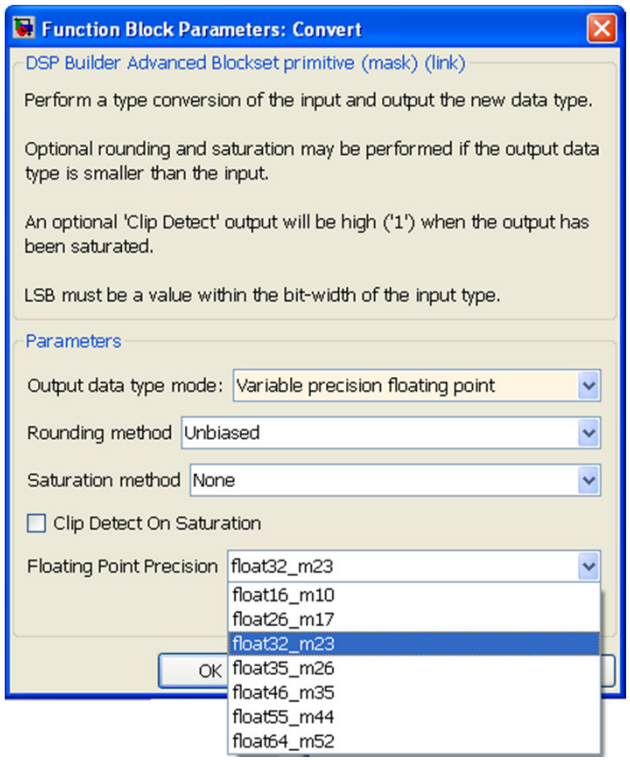
**Basic Floating Point**

FPAdd
FPAddExpert
FPAddN
FPSubExpert
FPAddSub \
FPAddSubExpert
FPFusedAddSub
FPMul
FPMulExpert
FPConstMul
FPAcc
FPSqrt
FPDivSqrt
FPRecipSqrt
FPCbrt
FPDiv
FPInverse
FPFloor
FPCeil
FPRound
FPRint
FPFrac
FPMod
FPDim
FPAbs
FPMin
FPMax
FPMinAbs
FPMaxAbs
FPMinMaxFused
FPMinMaxAbsFused
FPCompare
FPCompareFused


© 2015 Altera Corporation—Public

# Floating Point: Variable precision

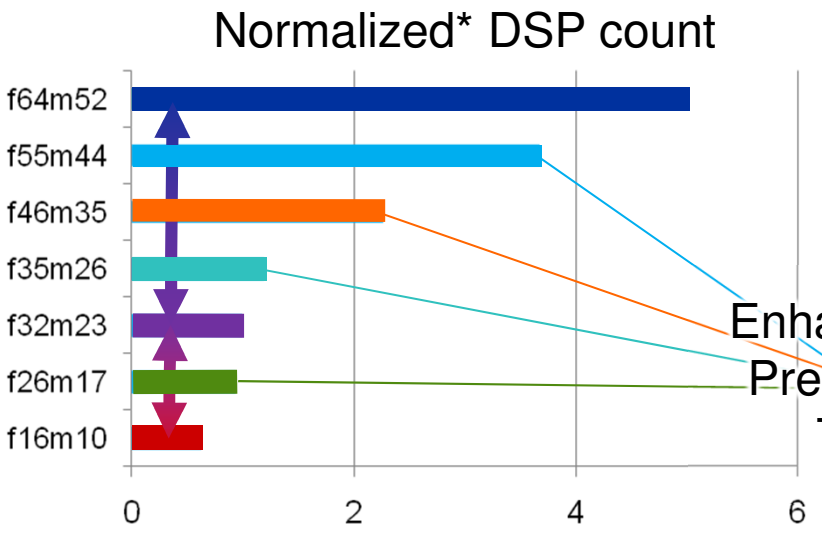
- IEEE format
- 7 precisions (including double and single)
  - float16_m10 (IEEE half)
  - float26_m17
  - float32_m23 (IEEE single)
  - float35_m26
  - float46_m35
  - float55_m44
  - float64_m52 (IEEE double)
  - where float***N***_m***M*** means total length of ***N*** bits, of which ***M*** are mantissa
- Format widths specifically chosen for efficient FPGA implementation
- Specified on Convert block
- Supported blocks are accurate within tolerances of given precision



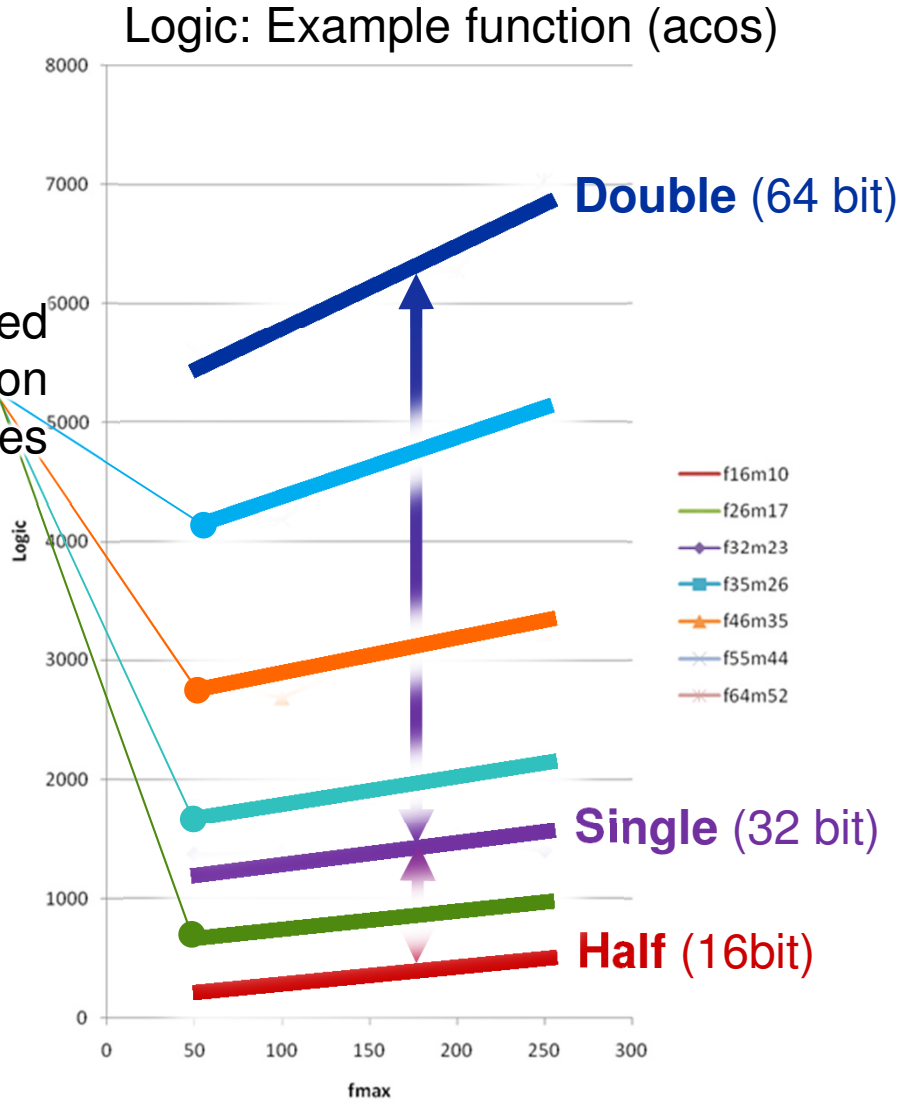

***Choose the right precision for your application***


© 2015 Altera Corporation—**Public**

# Soft Floating Point: Size vs. Precision Trade-off




Enhanced Precision Types

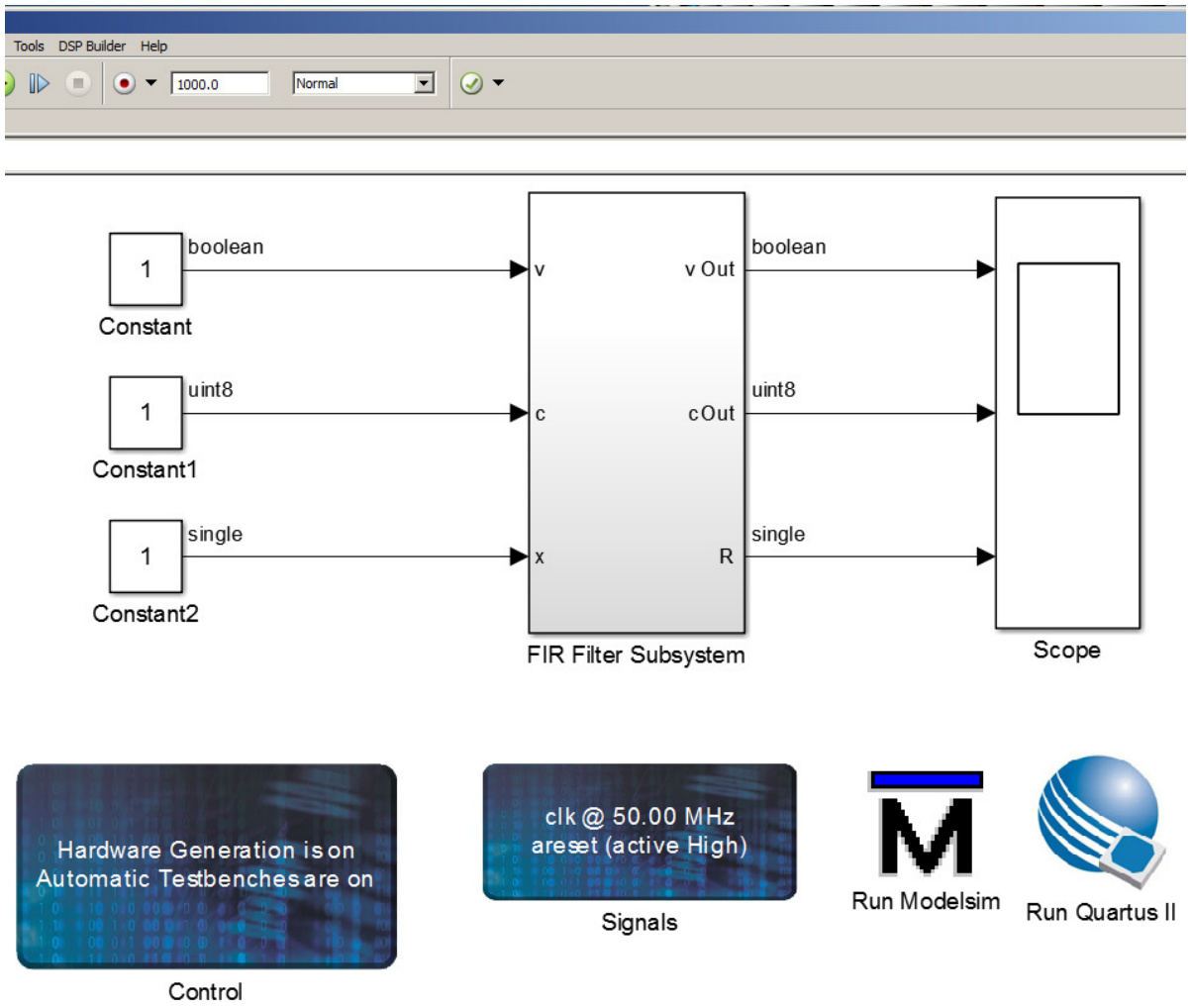



| Averaged logic & DSP use across maths functions, relative to single precision | | |
|---|---|---|
| Precision | DSP usage compared to single precision | Logic usage compared to single precision |
| **f16m10** | 0.6 | 0.3 |
| **f26m17** | 0.9 | 0.6 |
| **f32m23** | 1 | 1 |
| **f35m26** | 1.2 | 1.4 |
| **f46m35** | 2.2 | 2.2 |
| **f55m44** | 3.7 | 3.4 |
| **f64m52** | 5.0 | 4.6 |

(*scaled to single = 1)


© 2015 Altera Corporation—Public

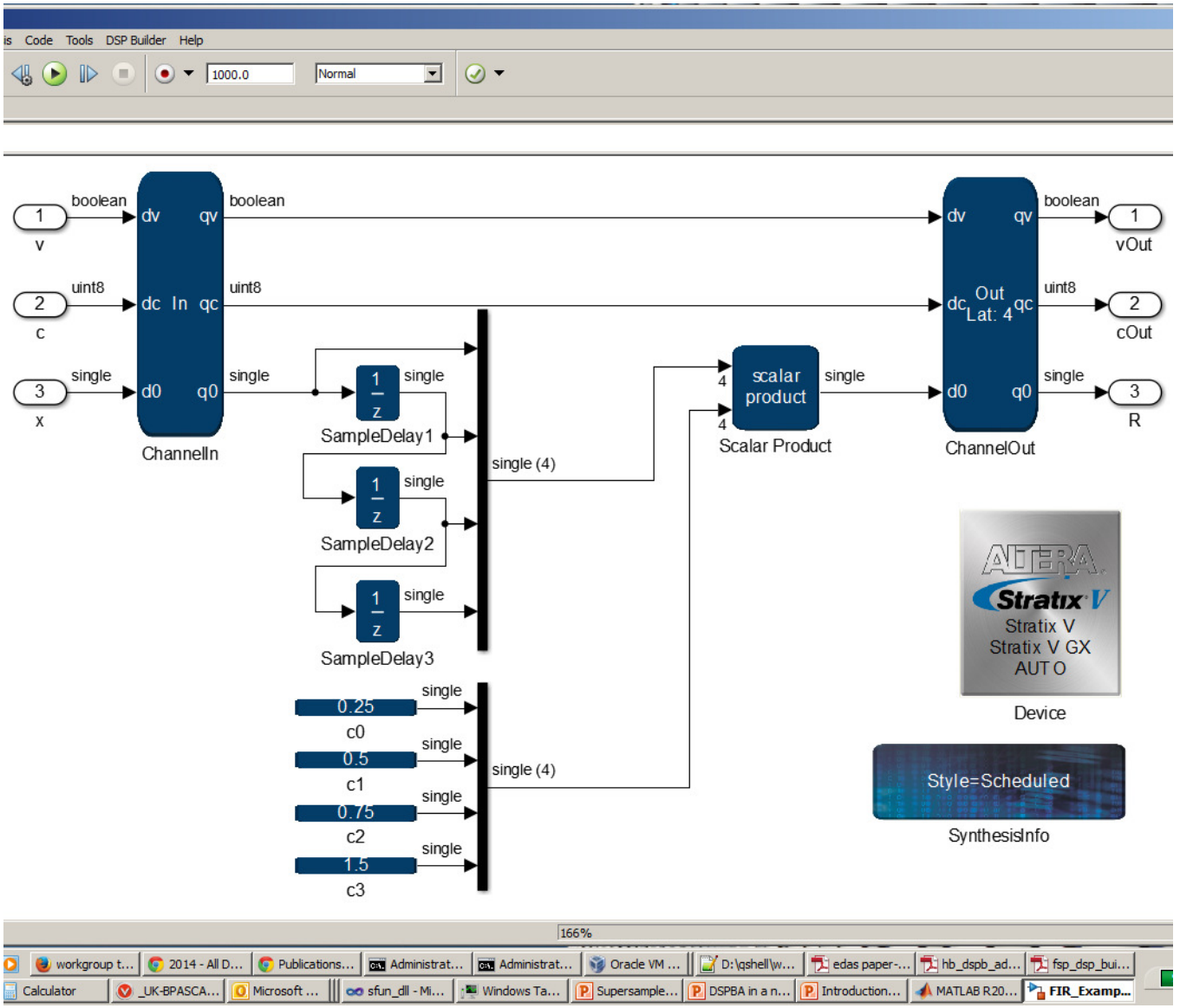
# High-level Design Specification





© 2015 Altera Corporation—Public

# FIR Filter Example





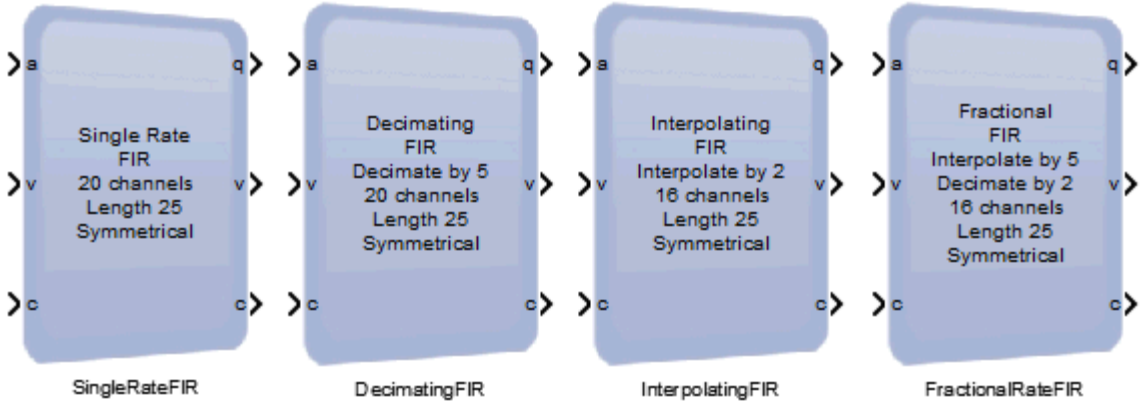
© 2015 Altera Corporation—Public

# Results

| Configuration | ALMs | DSPs |
|---|---|---|
| Stratix V, 128-tap | 60881 | 128 |
| Arria 10, 128-tap, HFP | 1676 | 131 |


© 2015 Altera Corporation—Public

# IP blocks: FIR Filters

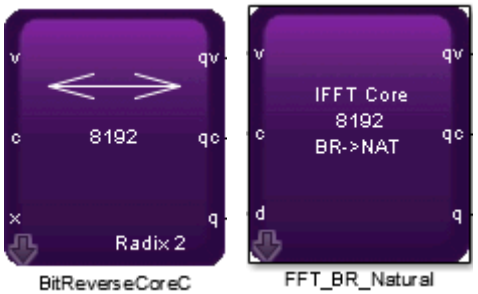



- Supports multichannel, TDM, sample rate > clock rate

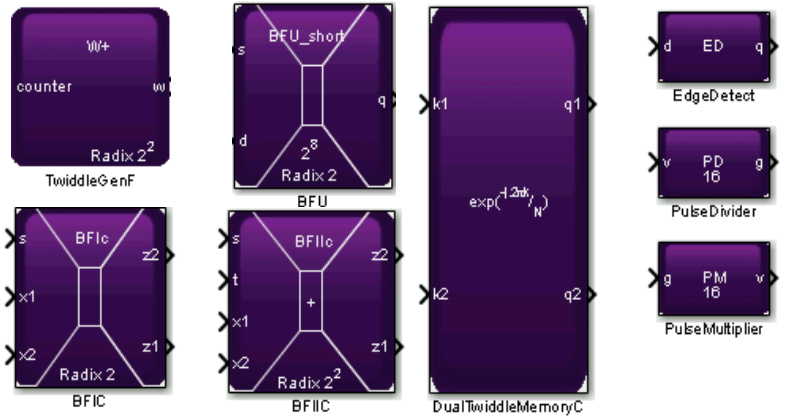

© 2015 Altera Corporation—**Public**

# IP blocks: FFTs

- Full FFTs
  - Fixed and Floating point
  - White-box – user parameterizable & editable IP
  - Parallel,
  - Variable Sized
  - Bit-Width Pruning Schemes




- Basic building blocks
  - Build your own custom FFTs




- Parameterizable demos
  - Including folded FFT demo
    - (low data rate, low resource)





© 2015 Altera Corporation—Public

# Conclusion





© 2015 Altera Corporation—**Public**

# Take-aways

- Floating point support is increasingly more important in FPGA applications
- Innovation in FP support for FPGAs is critical
  - Devices
  - Tools
- Because of growing diversity in the needs customers have for FP support, this is a great time for FP innovation


© 2015 Altera Corporation—Public

# Thank You

© 2015 Altera Corporation—**Public**

All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/legal.

