ALTERA®

# SOPC WORLD 2002

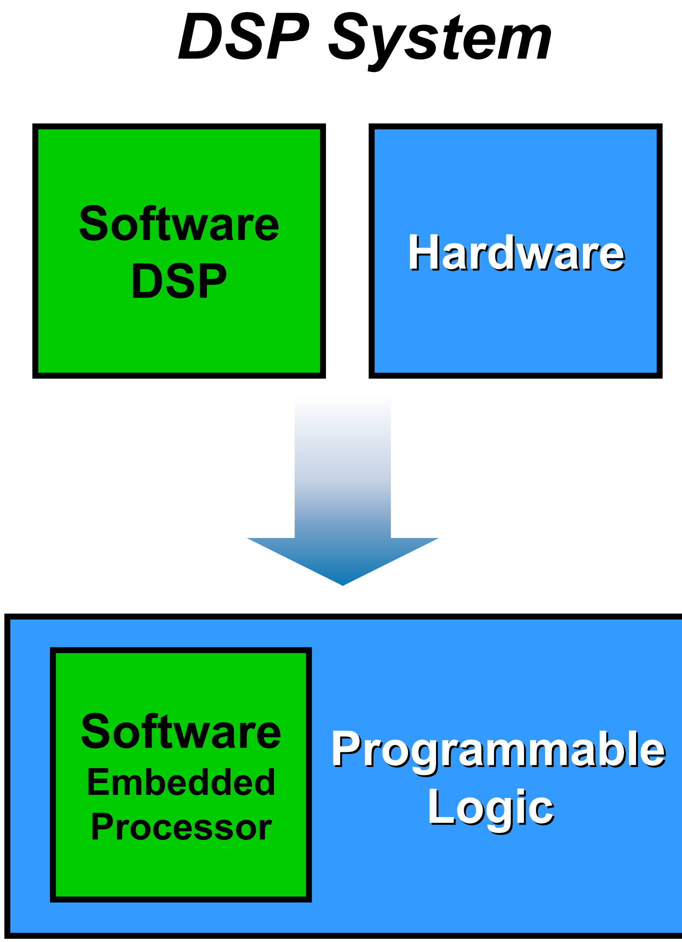
SOPC WORLD 2002

# Maximizing Productivity Using Simplified DSP Design Flow

ALTERA

# FPGAs for DSP Applications

- Benefits of FPGAs
  - 10x more DSP Throughput than DSP Processors
  - Cost-effective for Multi-Channel Applications
  - Flexible Hardware Implementation
  - System Integration Benefits
- Challenges
  - Designing with FPGAs is Difficult



# DSP Design Flow Challenges

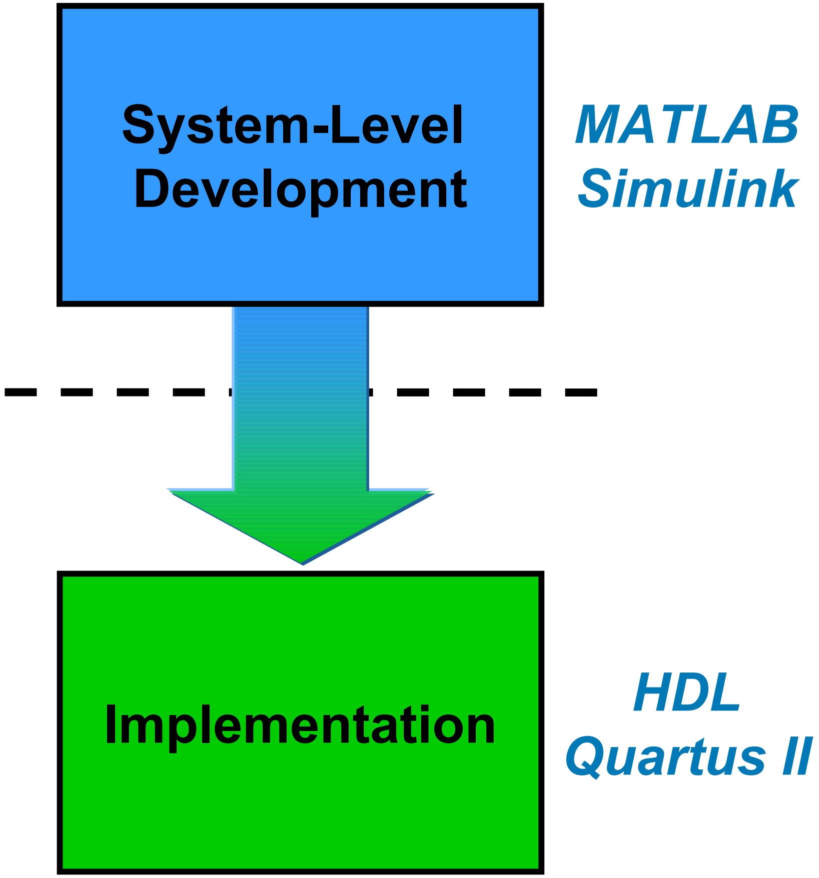
- System-Level Development & Verification
- Software/Hardware Co-Development
- Design Optimization

## System Development & Verification Challenges

- Multi-Platform
  - Development across Different Tools
- Modeling Accuracy
  - Floating-Point Simulation & Fixed-Point Implementation Incompatible
- Conversion
  - Manual Translation from System Level to Hardware

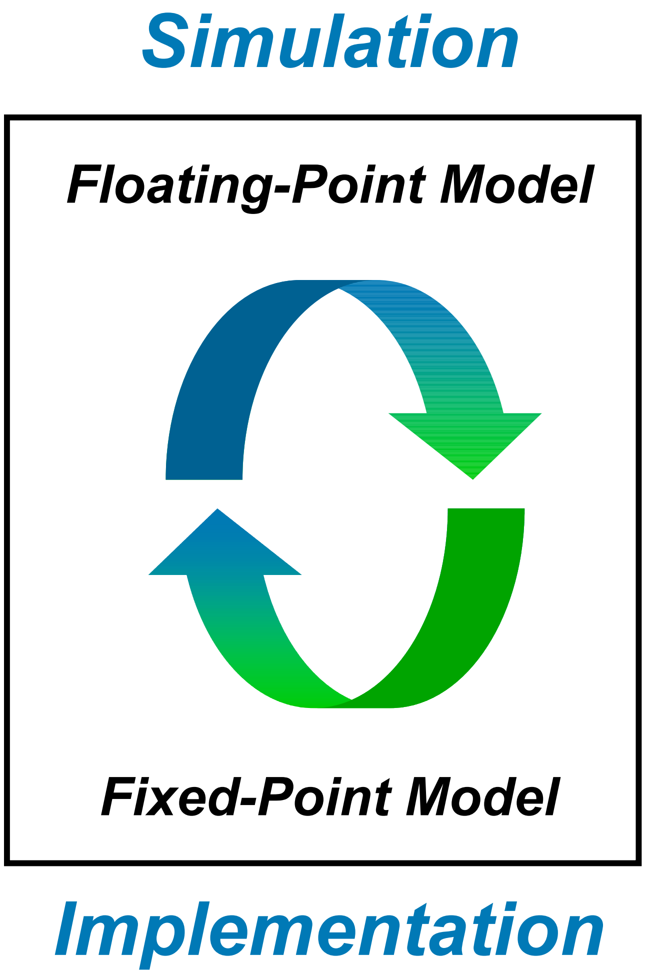
# Multi-Platform Challenges

- Lack of Integrated Design Environment
- Cannot Optimize During System Development Stage
  - Lacking Details on Underlying Architecture
- Risks During Implementation Stage
  - Ambiguous Interpretation of Specification



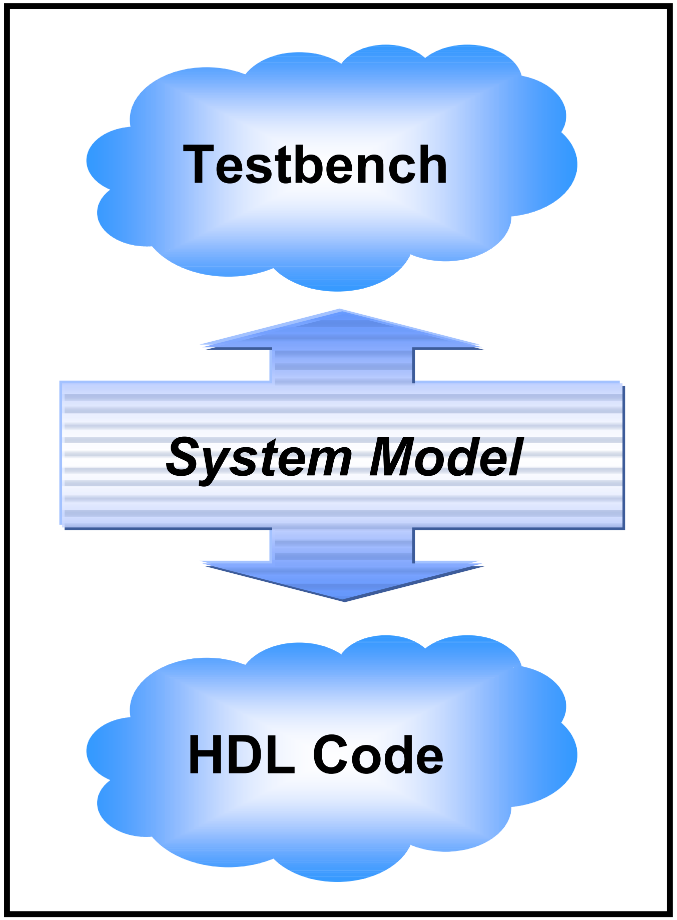
## Modeling Accuracy

- Floating-Point Models
  - Commonly used for Simulation
  - Most Efficient & Quickest Solution for Early Analysis
- Fixed-Point Models
  - Commonly used for Implementation
  - Suffers from Finite Word Length Effect
  - Need Truncation, Rounding & Saturation



# Conversion Challenges

- System Design to Hardware Implementation
  - Requires HDL Knowledge
  - Create RTL Model
  - Create Simulation Testbench
- Complex Conversion Rules
  - Bit Propagation
  - Multi-Rate Systems



## Software/Hardware Co-Development Challenges

- System Partitioning
  - Trade-off Flexibility Versus Performance
  - Need Multiple Design Iterations to Find Optimal Solution
- Integration of IP & Custom Logic
  - Different Bus Interfaces
- Software/Hardware Dependency
  - Frequent Updates to Header Files & Drivers

## Design Optimization Challenges

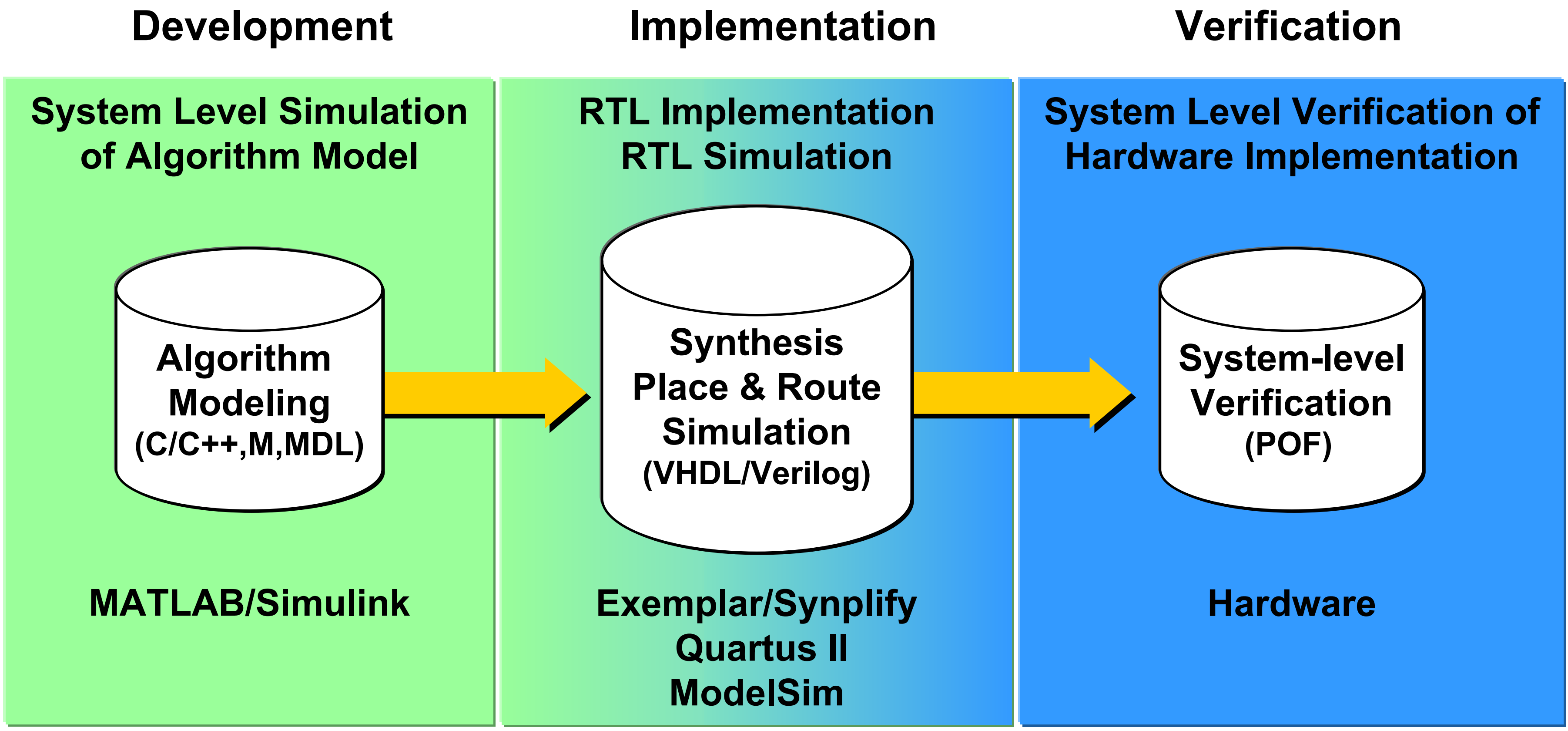
- C/C++ Coding
  - Inefficient Compiler
  - Needs to Be Tailored for DSP Specific Architectural Features
- Assembly Coding
  - Need Understanding of Specific Machine Instruction Set for Specific Processor for Optimization
  - Systems Getting Larger & More Complex
    - Not Feasible for Hand-Coding
  - May Not Be Sufficient for Certain Intensive Number-Crunching Requirements

## Addressing Challenges

- System-Level Development and Verification
  - DSP Builder Tool
    - System Integration
    - Bit-True & Cycle Accurate Models
    - Automatic Translation into Hardware
- Hardware/Software Integration
  - SOPC Builder Tool and Nios Processor
    - C-based design flow
- Design Optimization
  - Hardware Acceleration
  - Flexibility in System Partitioning

# Traditional DSP Design Flow in FPGA

- System Algorithm Design & FPGA Design Separated

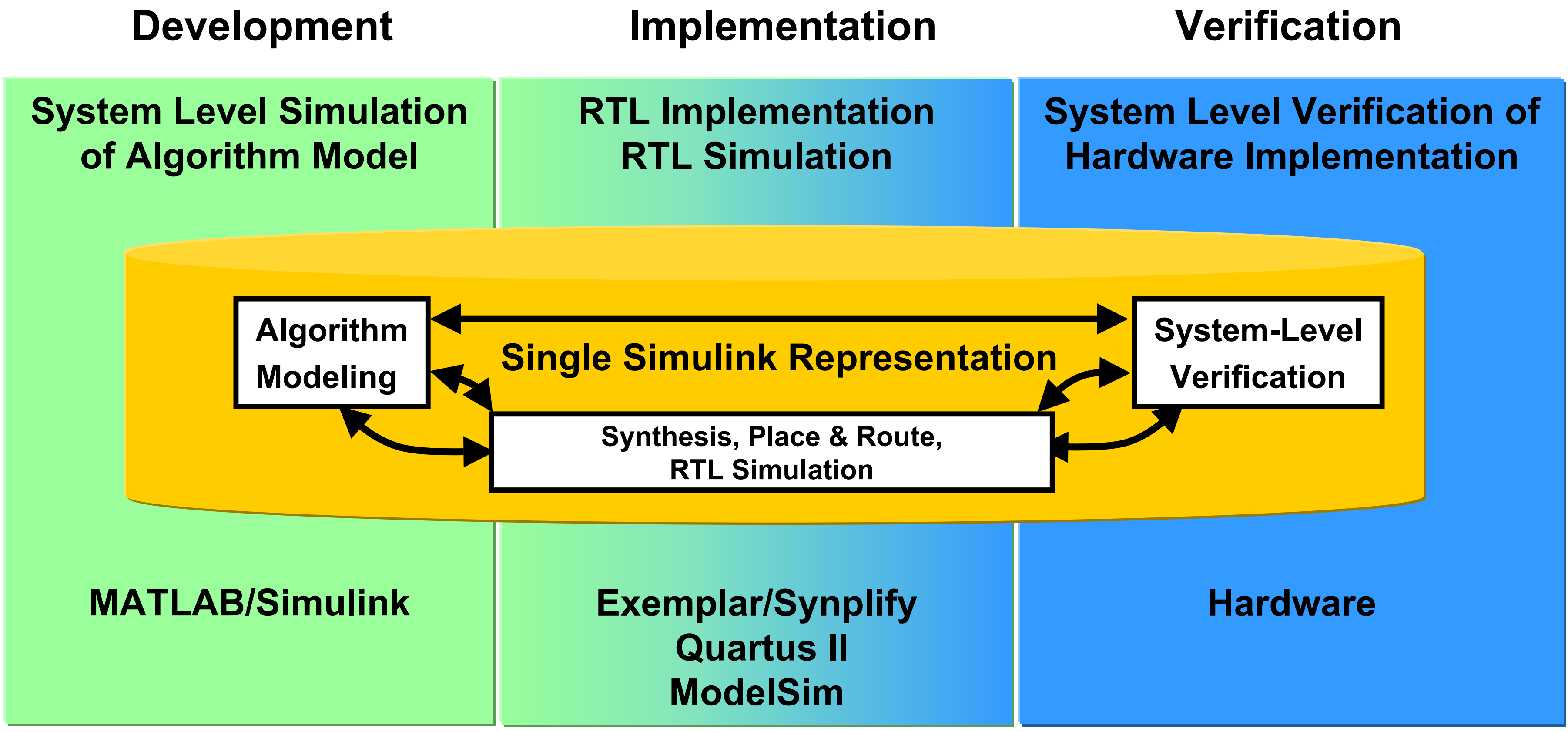

Development
Implementation
Verification
System Level Simulation of Algorithm Model
RTL Implementation RTL Simulation
System Level Verification of Hardware Implementation
Algorithm Modeling (C/C++,M,MDL)
Synthesis Place & Route Simulation (VHDL/Verilog)
System-level Verification (POF)
MATLAB/Simulink
Exemplar/Synplify Quartus II ModelSim
Hardware

SOPC WORLD 2002

ALTERA

# Integration Using DSP Builder

- System Algorithm Design and FPGA Design Integrated

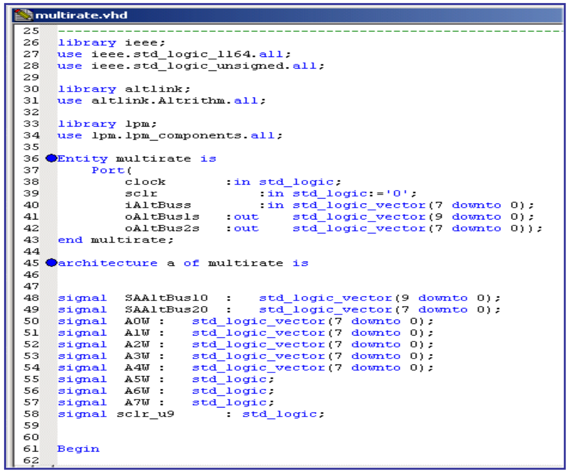

Development
Implementation
Verification
System Level Simulation of Algorithm Model
RTL Implementation RTL Simulation
System Level Verification of Hardware Implementation
Algorithm Modeling
Single Simulink Representation
System-Level Verification
Synthesis, Place & Route, RTL Simulation
MATLAB/Simulink
Exemplar/Synplify
Quartus II
ModelSim
Hardware

## Bit-True & Cycle-Accurate Models

- DSP Builder Provides Bit-True & Cycle-Accurate Simulink Blocks
- Ideal for System-Level Simulation
- Benefits
  - High-Level Abstraction
  - Don’t Model Hardware Detail Involving Unnecessary Data Path Calculations
  - Faster than RTL Simulation
  - Most Important Prior to Architecture Mapping
  - Accurate Hardware Results

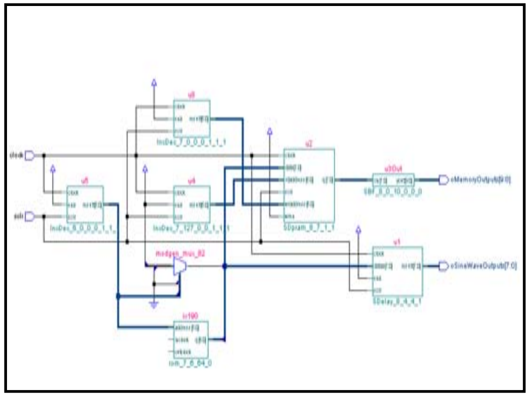
# Automatic Hardware Translation

## Creates HDL Code



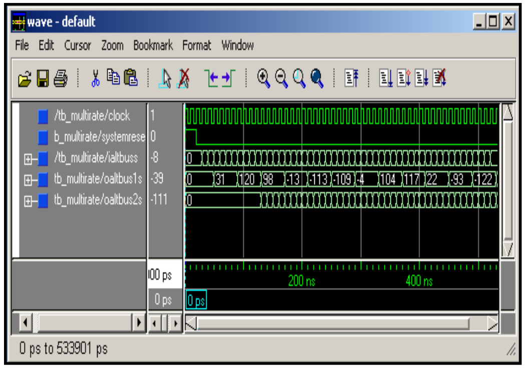
```
-----------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

library altlink;
use altlink.Altrithm.all;

library lpm;
use lpm.lpm_components.all;

Entity multirate is
    Port(
        clock        :in std_logic;
        sclr             :in std_logic:='0';
        iAltBuss         :in std_logic_vector(7 downto 0);
        oAltBus1s    :out    std_logic_vector(9 downto 0);
        oAltBus2s    :out    std_logic_vector(7 downto 0));
end multirate;

architecture a of multirate is


signal  SAAltBus10  :    std_logic_vector(9 downto 0);
signal  SAAltBus20  :    std_logic_vector(7 downto 0);
signal  A0W :   std_logic_vector(7 downto 0);
signal  A1W :   std_logic_vector(7 downto 0);
signal  A2W :   std_logic_vector(7 downto 0);
signal  A3W :   std_logic_vector(7 downto 0);
signal  A4W :   std_logic_vector(7 downto 0);
signal  A5W :   std_logic;
signal  A6W :   std_logic;
signal  A7W :   std_logic;
signal sclr_u9      : std_logic;


Begin
```

## HDL Synthesis

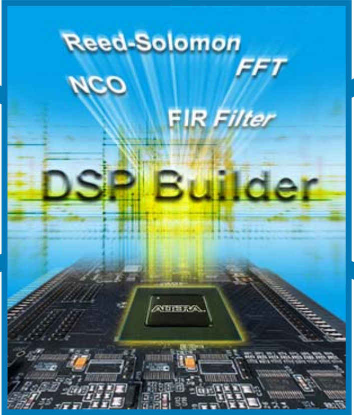


## Place & Route

## Creates Simulation Testbench



```
/tb_multirate/clock           1
b_multirate/systemrese        0
/tb_multirate/ialtbuss        -8
tb_multirate/oaltbus1s        -39
tb_multirate/oaltbus2s        -111

oaltbus1s: 0 31 120 98 -13 -113 -109 -4 104 117 22 -93 -122
00 ps
0 ps
0 ps to 533901 ps
```



## Download Design to DSP Development Kits

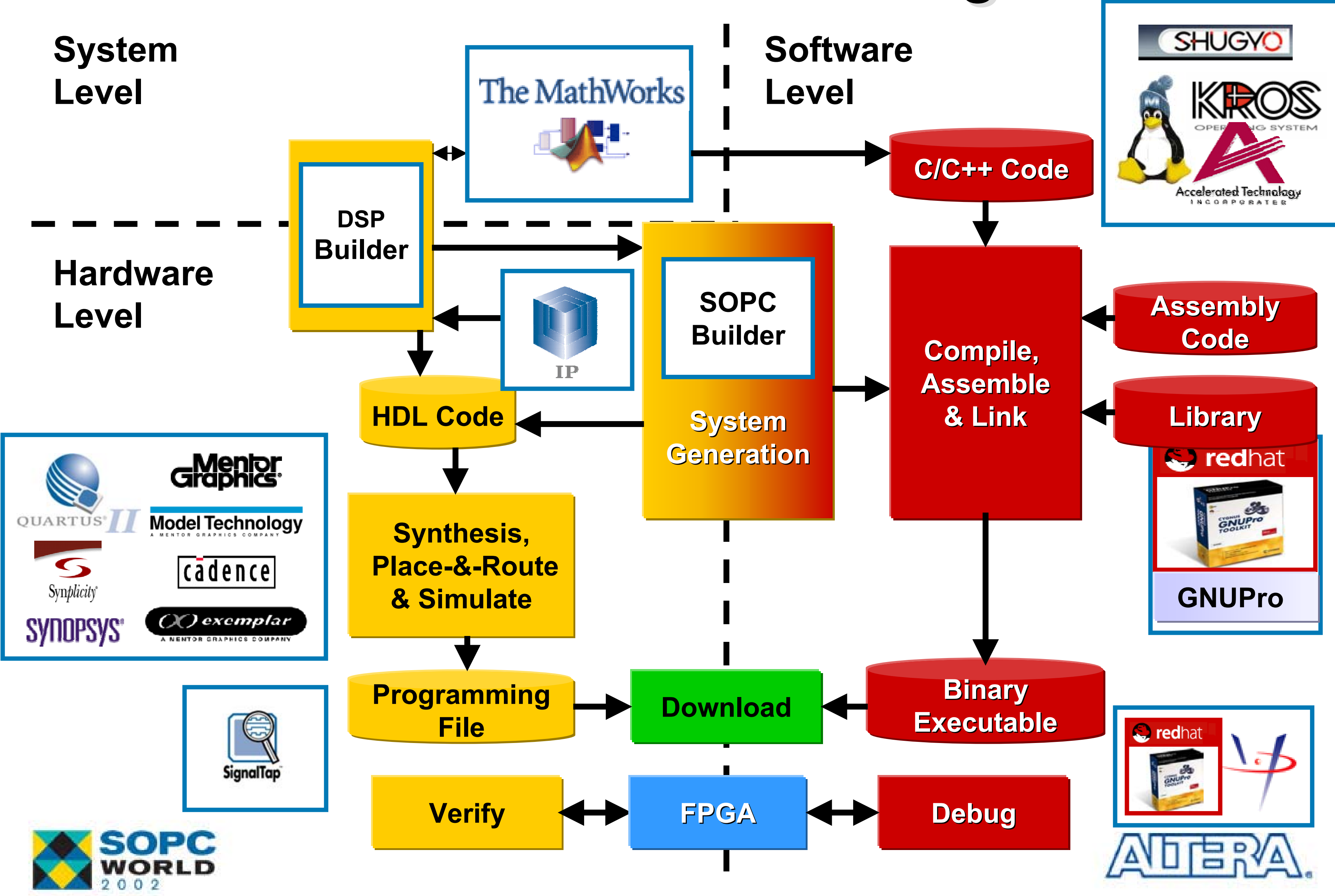
## Creates Plug In to Processor

# Accelerated Path to Co-Design

- SOPC Builder Tool
  - Combines Existing Soft & Hard IP Blocks & Associated Software
  - Generates Interfaces between Hardware & Software
  - Solves Problem of Linking IP Cores from Several Vendors
  - Available in Quartus II Software
- Supports Existing Altera Intellectual Property (IP) & ARM®-Based Excalibur & Nios® Embedded Processors
- Allows Flexibility for Changes to Software/Hardware Partitioning

Nios™
SOPC Builder
Concept to System in Minutes
QUARTUS® II

# Hardware/Software DSP Design Flow

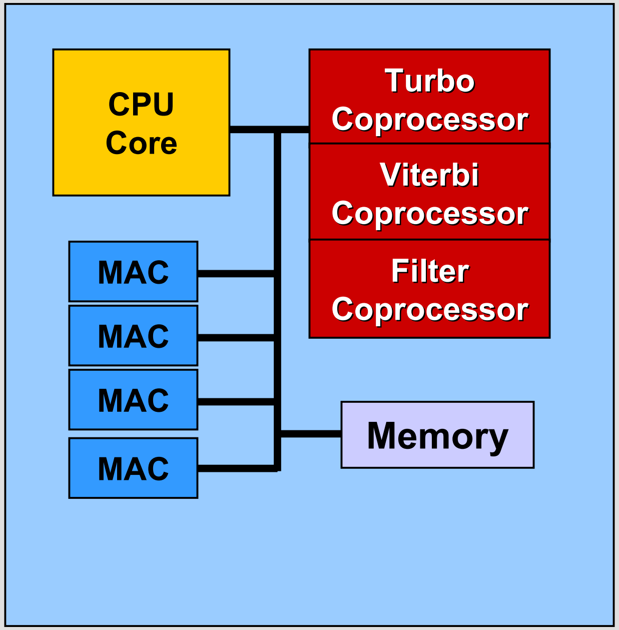


## Hardware Acceleration

- Implement Computationally Intensive & Repetitive Tasks in Hardware
  - Filters, Encoders/Decoders
- Examples in DSP Processors
  - TI TMS320C6416
    - VCP – Viterbi Coprocessor
      - 350 Voice Channels at 12.2 Kbps
  - Motorola MSC8102
    - EFCOP – Enhanced Filter Coprocessor
      - 4 Processors at 300 MHz
- Dedicated Hardware Accelerators Are Inflexible

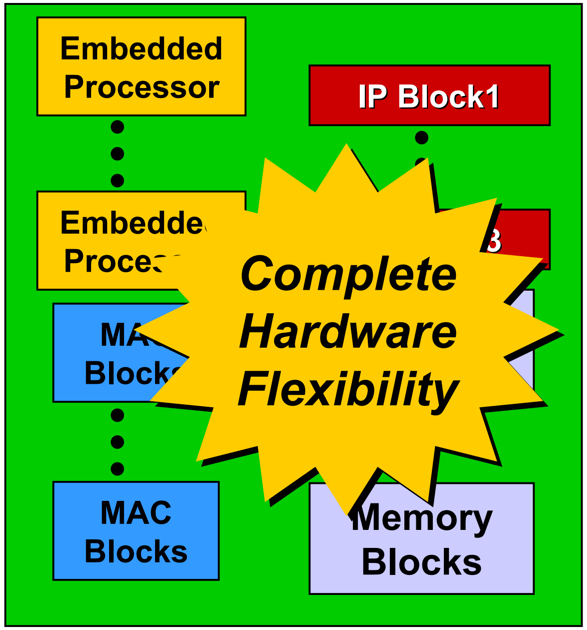
# Optimization Using FPGAs

## DSP Processors



- Fixed CPU Architecture
- Fixed Memory Structure
- Fixed Bus Structure
- Predefined Hardware Accelerator Blocks
- *Few* MAC Blocks

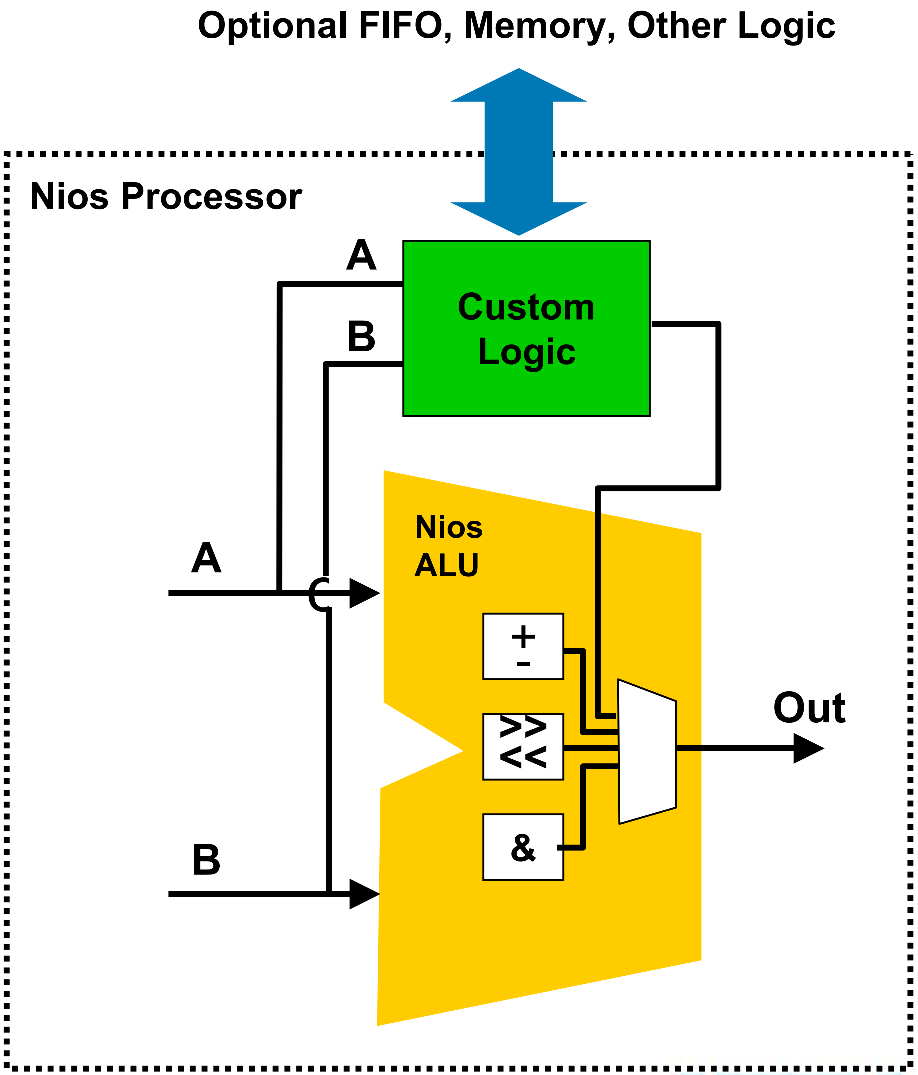
## FPGAs



- Customizable CPU Structure
- Customizable Memory Structure
- Customizable Bus Structure
- User-Defined Hardware Accelerator Blocks
- *Large* Number of MAC Blocks

# Hardware Acceleration in FPGA

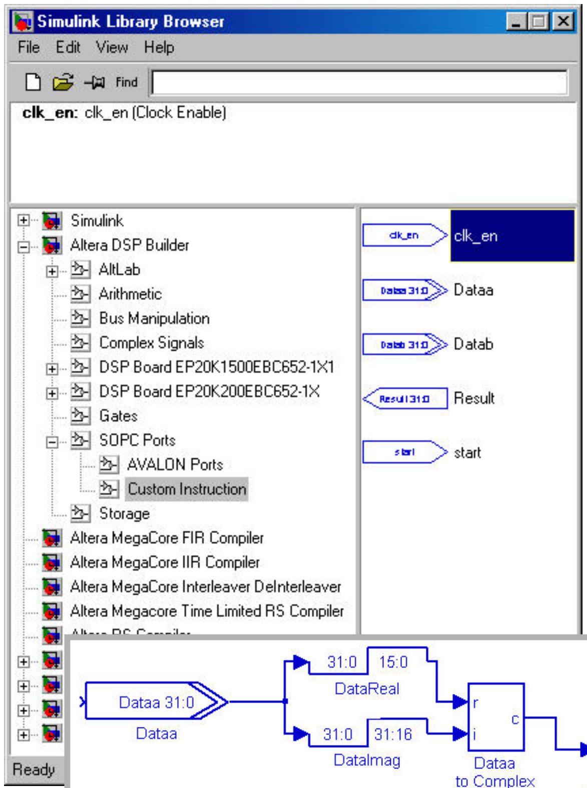
- Two Implementation Options
  - Custom Peripheral
  - Custom Instruction
- Custom Peripheral
  - Interface to Nios through Avalon Bus
- Custom Instruction
  - Adds Customized Logic to Nios ALU
  - Generates C & Assembly Macros

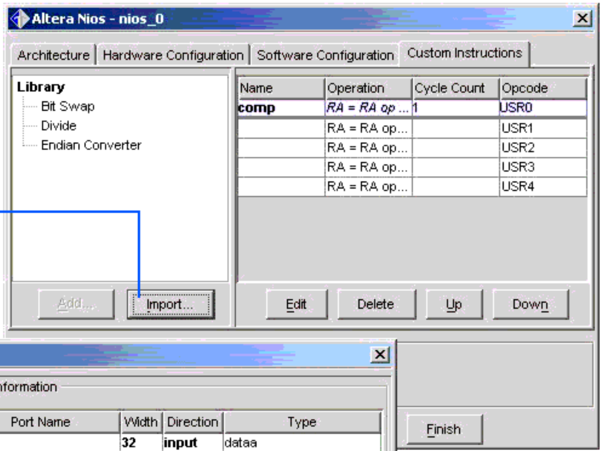


SOPC WORLD 2002

ALTERA

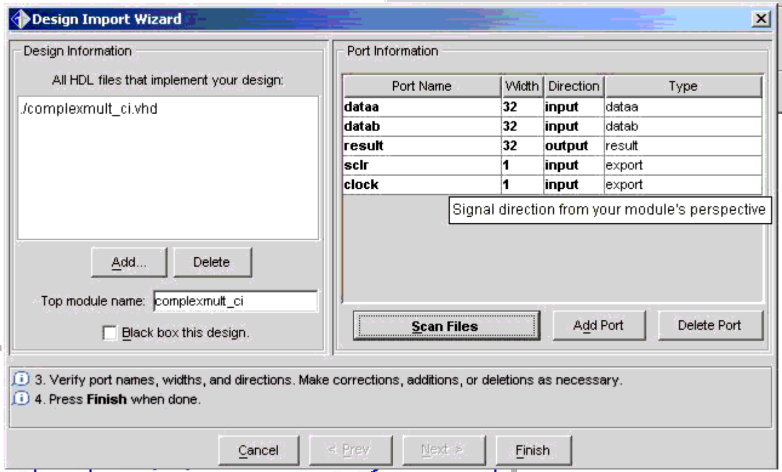
# Custom Instructions


Simulink Library Browser
File Edit View Help
Find
clk_en: clk_en (Clock Enable)
Simulink
Altera DSP Builder
AltLab
Arithmetic
Bus Manipulation
Complex Signals
DSP Board EP20K1500EBC652-1X1
DSP Board EP20K200EBC652-1X
Gates
SOPC Ports
AVALON Ports
Custom Instruction
Storage
Altera MegaCore FIR Compiler
Altera MegaCore IIR Compiler
Altera MegaCore Interleaver DeInterleaver
Altera Megacore Time Limited RS Compiler
clk_en
Dataa
Datab
Result
start
Ready



| Name | Operation | Cycle Count | Opcode |
|---|---|---|---|
| comp | RA = RA op ... | 1 | USR0 |
| | RA = RA op... | | USR1 |
| | RA = RA op... | | USR2 |
| | RA = RA op... | | USR3 |
| | RA = RA op... | | USR4 |

**Import Custom Instruction in SOPC Builder**



All HDL files that implement your design:

./complexmult_ci.vhd

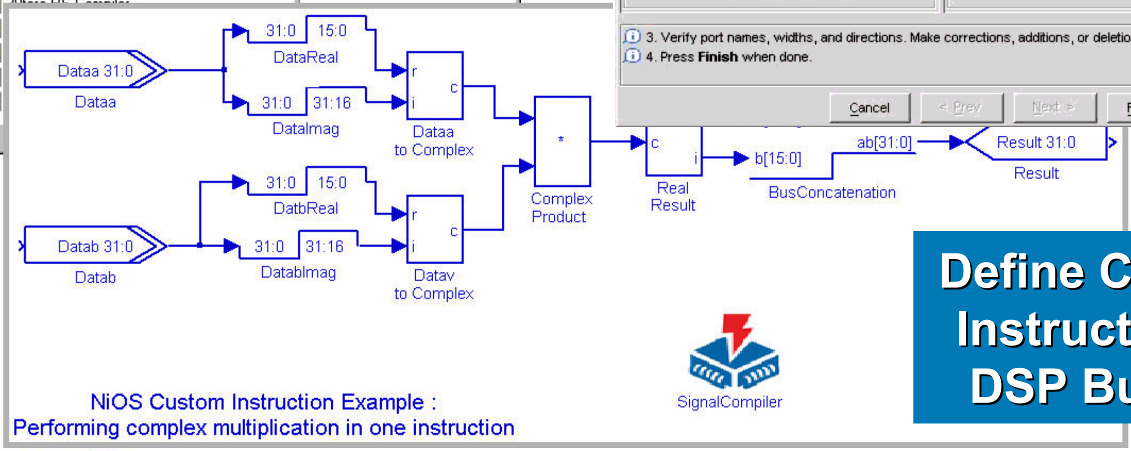
Top module name: complexmult_ci

Black box this design.

| Port Name | Width | Direction | Type |
|---|---|---|---|
| dataa | 32 | input | dataa |
| datab | 32 | input | datab |
| result | 32 | output | result |
| sclr | 1 | input | export |
| clock | 1 | input | export |

Signal direction from your module's perspective

3. Verify port names, widths, and directions. Make corrections, additions, or deletions as necessary.
4. Press **Finish** when done.



NiOS Custom Instruction Example :
Performing complex multiplication in one instruction

**Define Custom Instruction in DSP Builder**

SOPC WORLD 2002

ALTERA

## Performance Using Custom Instructions

| Floating-Point Operation (32-Bit Data) | CPU Clock Cycles | | Speed Increase |
|---|---|---|---|
| | Software Library | Custom Instruction | |
| Multiplication axb | 2874 | 19 | 151x |
| Multiply & Negate –(axb) | 3147 | 19 | 165x |
| Absolute \|a\| | 1769 | 18 | 98x |
| Negate –(a) | 284 | 19 | 15x |

**Note: These Performance Calculations are Compiler-Dependant. Taken Using the Cygnus Compiler Included in Version 2.1 of Nios Embedded Processor**

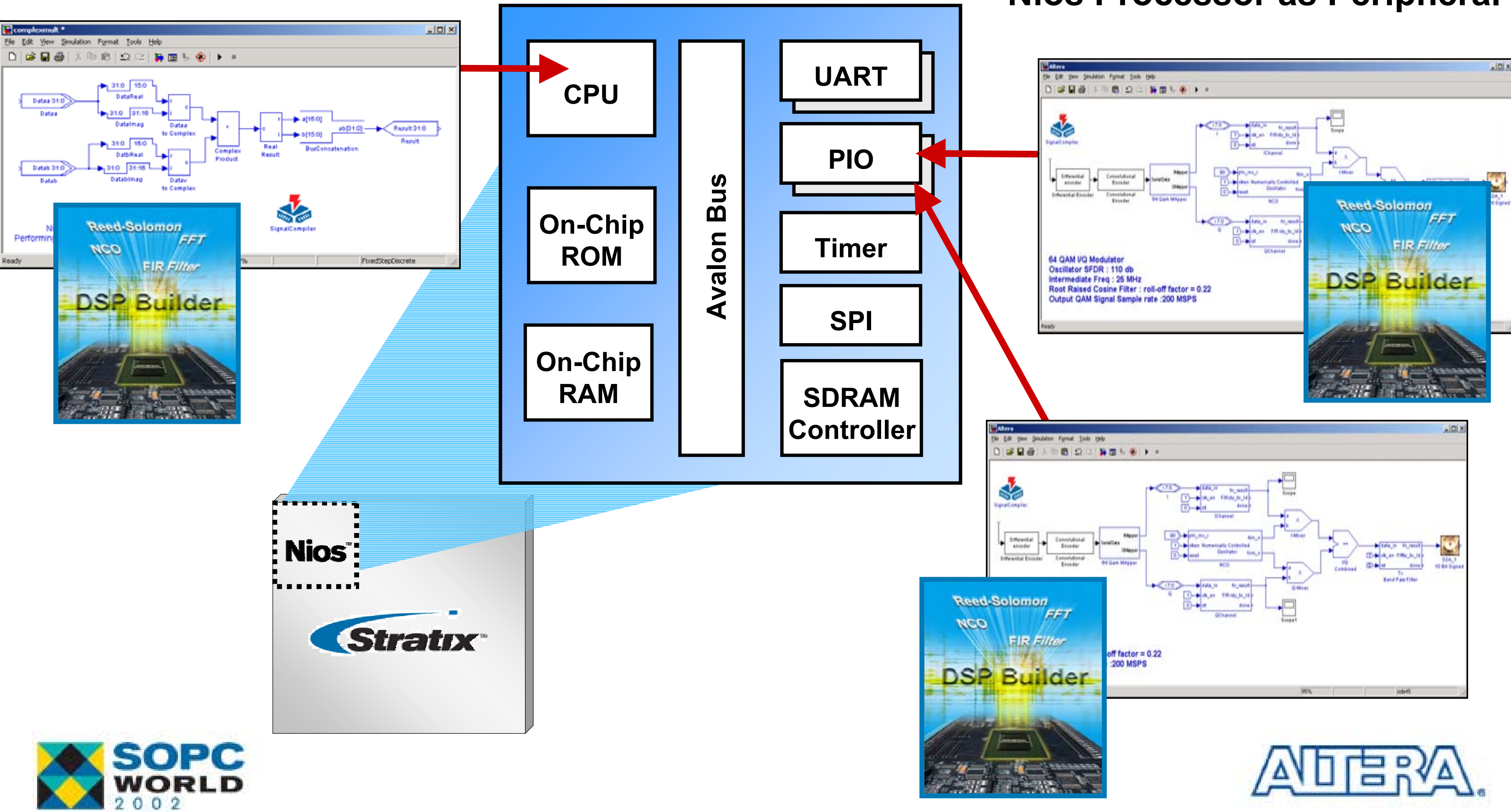
# Acceleration in DSP Builder

*Custom Instruction*

**Algorithms Developed in DSP Builder become Integral Part of ALU**

*Custom Peripheral*

**Algorithms Developed in DSP Builder Can Be Connected to Nios Processor as Peripheral**



## Summary

- Integrated Design Platform for Efficient DSP Design Flow
  - DSP Builder Tool
  - Accurate Modeling
  - Seamless Flow from System to Hardware
- Versatile Tool for Software/Hardware Integration
  - SOPC Builder and Nios Embedded Processor
  - Easy System Partitioning
- Hardware Acceleration for Design Optimization
  - Hardware Flexibility
- DSP Design with FPGAs Becoming Easier