Accelerating Your Success™

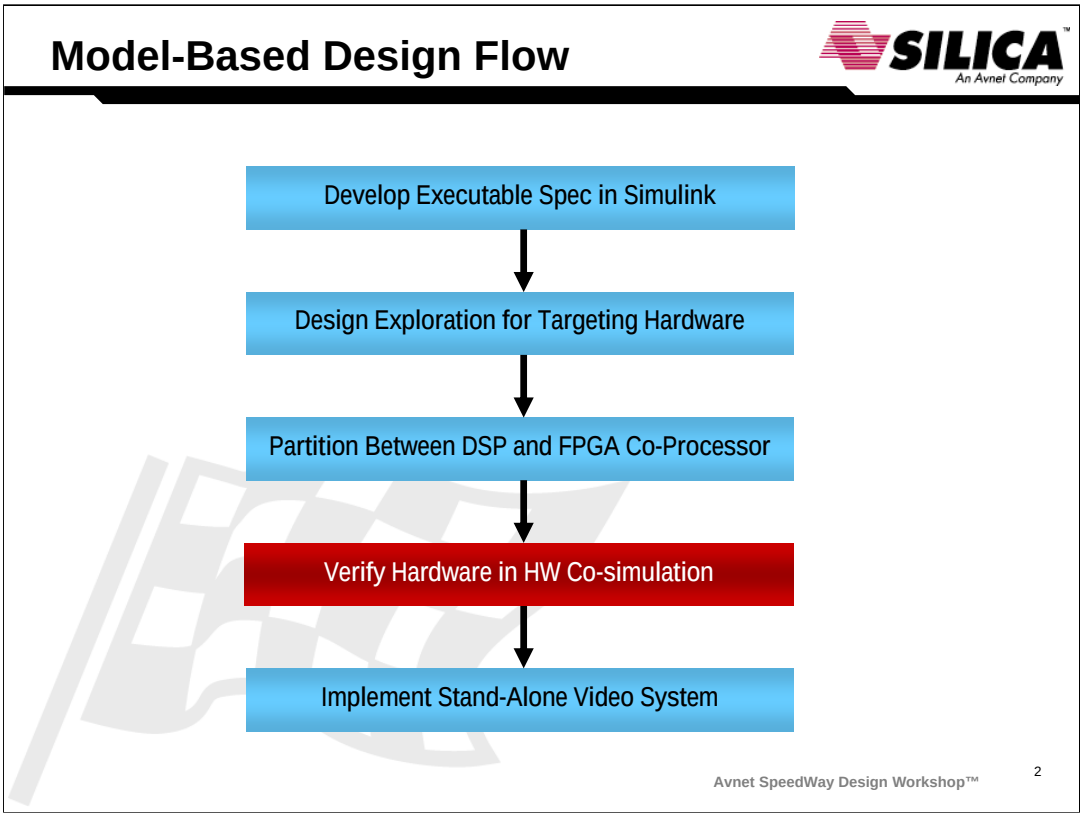
SILICA An Avnet Company

# Avnet Speedway Design Workshop™

## Creating FPGA-based Co-Processors for DSPs Using Model Based Design Techniques

## Lecture 4: FPGA Co-Processor Architectures and Verification

V10_1_2_0




We are now at the hardware co-simulation phase in the model-based design flow.

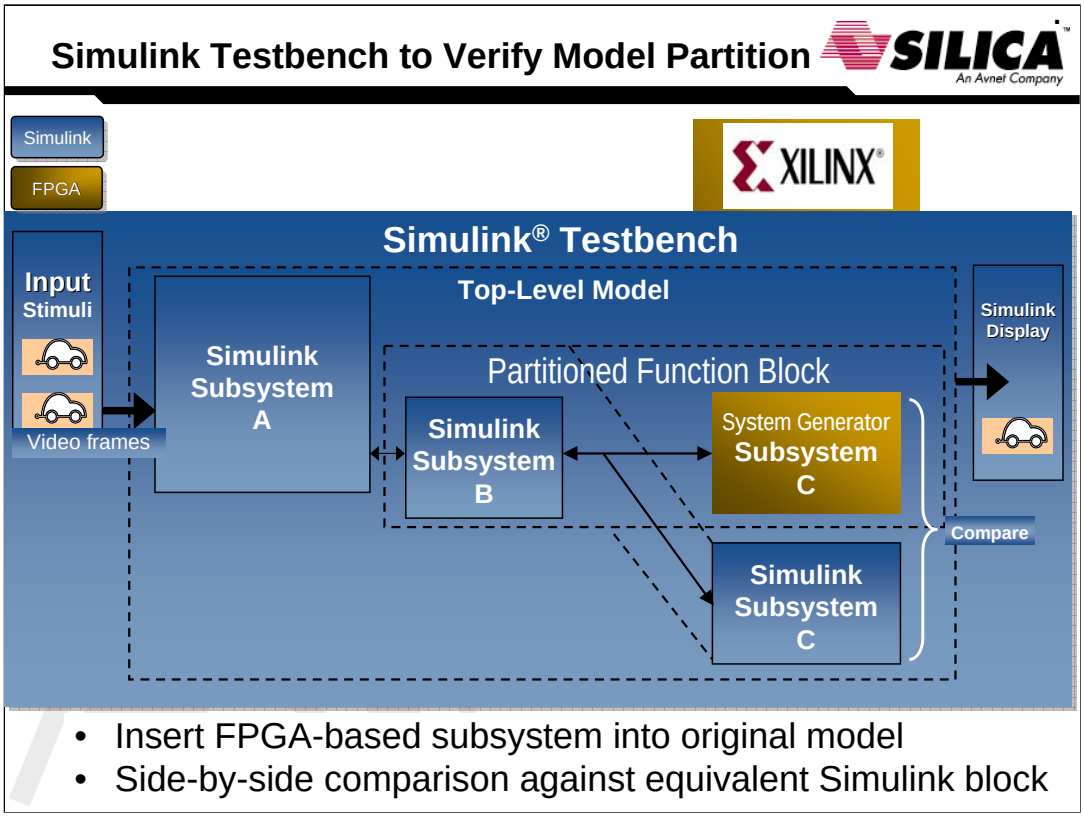
## The Problem We Wish to Solve

**Verification of hardware-based video processing systems using traditional HDL simulators is tedious and slow.**

**We propose a Simulink-based verification methodology using FPGA hardware co-simulation which operates on entire video frames and offers dramatic acceleration of simulation times.**

## Agenda

- FPGA model verification using Hardware co-simulation with System Generator and Simulink




In lab 3, we partitioned the video stabilization algorithm by implementing SAD in the FPGA. We built the SAD as a System Generator model and verified it's functionality in isolation, using test images.

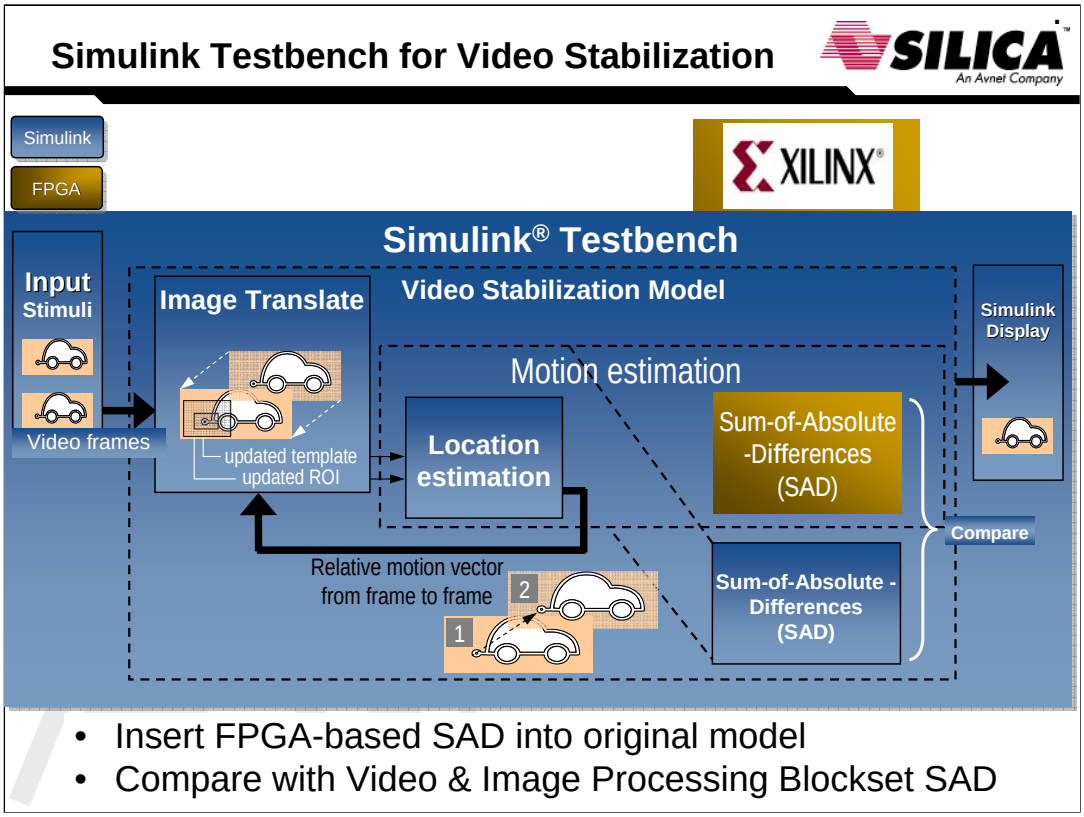
Now that we have decided how to partition the Simulink model between DSP and FPGA ...

< mouse click >

... we use a Simulink testbench to verify that our partitioned model gives the same results in simulation as the original, by inserting the FPGA block for side-by-side comparison.

Note Simulink's ability to propagate 2-D arrays as frames in the model, which eases the modeling and simulation of video and image processing algorithms.

This is a verification methodology to exercise the FPGA functionality only; the DSP is not involved yet because we are still at the verification stage. When the FPGA co-processor has been validated in simulation, we will move towards a stand-alone system in lab 5 where all functionality in the left side of the above diagram will be implemented in the DSP.




Verification process described in previous slide, applied to the video stabilization model.

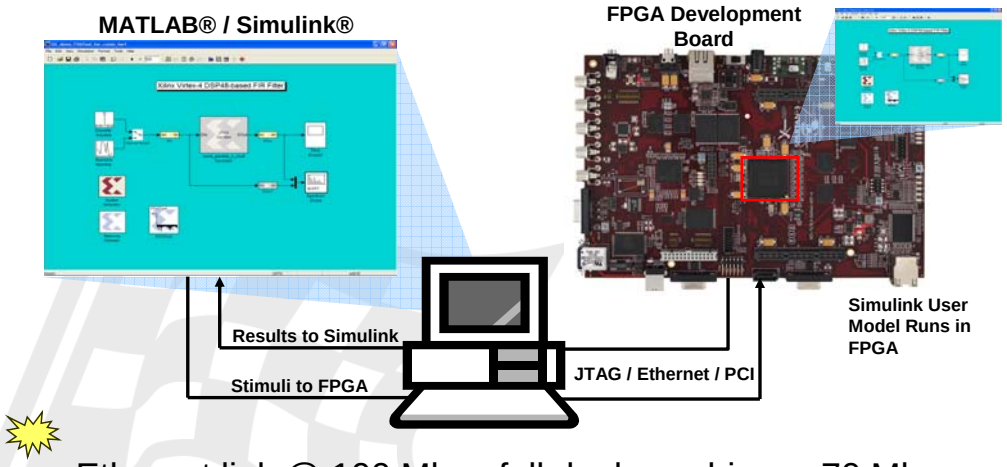
The technique we use to include the FPGA co-processor into the Simulink testbench is called hardware co-simulation ...

## Hardware Co-Simulation in Xilinx DSP Tools

- An enabling technology for
  - Simulation **Acceleration**
  - On-chip **Verification**
  - Run-time **Debugging**
  - Software/Hardware **Interaction**
  - **Benchmarking** (Fmax and Power)
- Rapid prototype of FPGA signal processing designs on actual board

## Xilinx Hardware Co-Simulation

- Bit true and cycle true validation in FPGA
- Accelerates simulation by orders of magnitude




- Ethernet link @ 100 Mbps full duplex achieves 70 Mbps data payload

Avnet SpeedWay Design Workshop™ 8

**Hardware co-simulation CONCEPT SLIDE** :

Xilinx System Generator for DSP features a powerful simulation mode called hardware co-simulation, shown above.

The benefits :

•Immediate proof of concept of DSP model directly in hardware

•simulation acceleration, especially on large models such as 16K FFT

•allows simulation of IP cores delivered in netlist format.

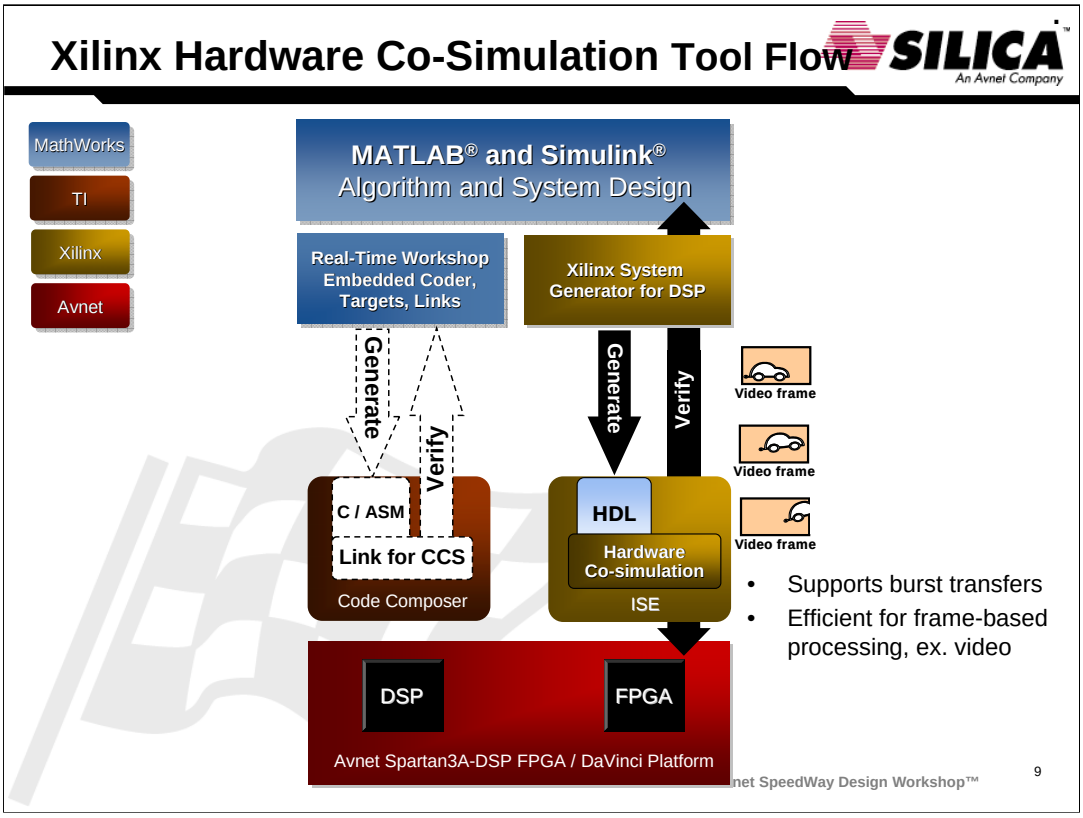
Simulink runs on the PC, sending input data over the Ethernet interface to the model running in the FPGA. The input data consists of a Simulink signal source. Processed results output from the model are sent back for display in Simulink.

Shown above at the top right is the hardware co-simulation block, which stands for all or part of the System Generator model, after generating hardware to target board. (All Avnet Xilinx development boards have JTAG support in Xilinx System Generator for DSP. Avnet V5 LX50, V5 95SXT and Spartan-3A DSP / DaVinci are supported for hardware co-simulation over Ethernet @ 100 Mbps full-duplex)

Anticipated questions :

Is hardware co-simulation running at the full board system clock rate ?

What is the difference between single-step and free-running modes for hardware co-simulation ?



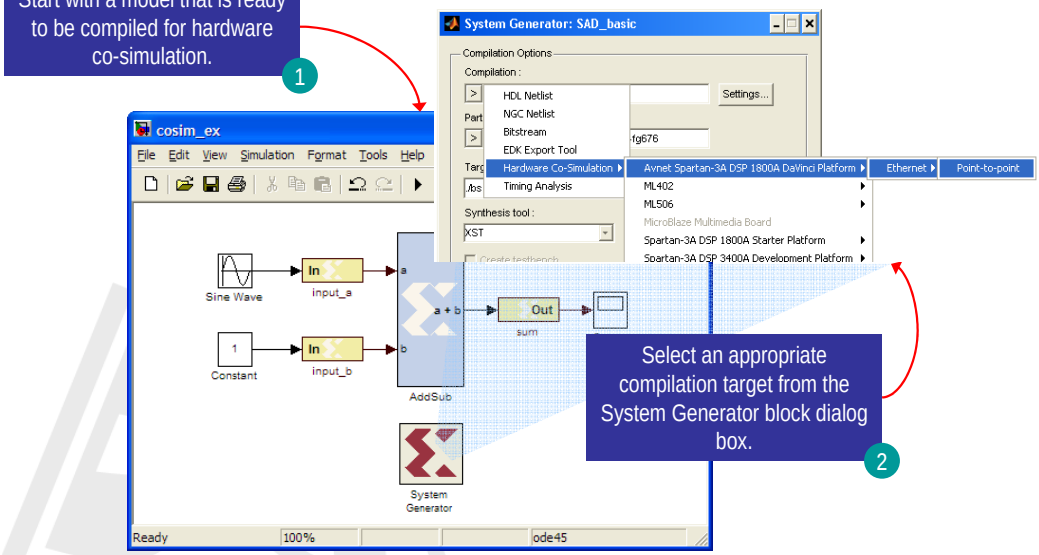

System Generator provides hardware co-simulation, making it possible to incorporate a design running in an FPGA directly into a Simulink simulation. Compiling a model for hardware co-simulation automatically creates a bitstream and associates it to a System Generator co-simulation block. When the design is simulated in Simulink, stimuli are sent to the compiled hardware model in the FPGA and results sent back to Simulink. This allows the compiled portion to be tested in actual hardware and can speed up simulation dramatically.

*< mouse click >*

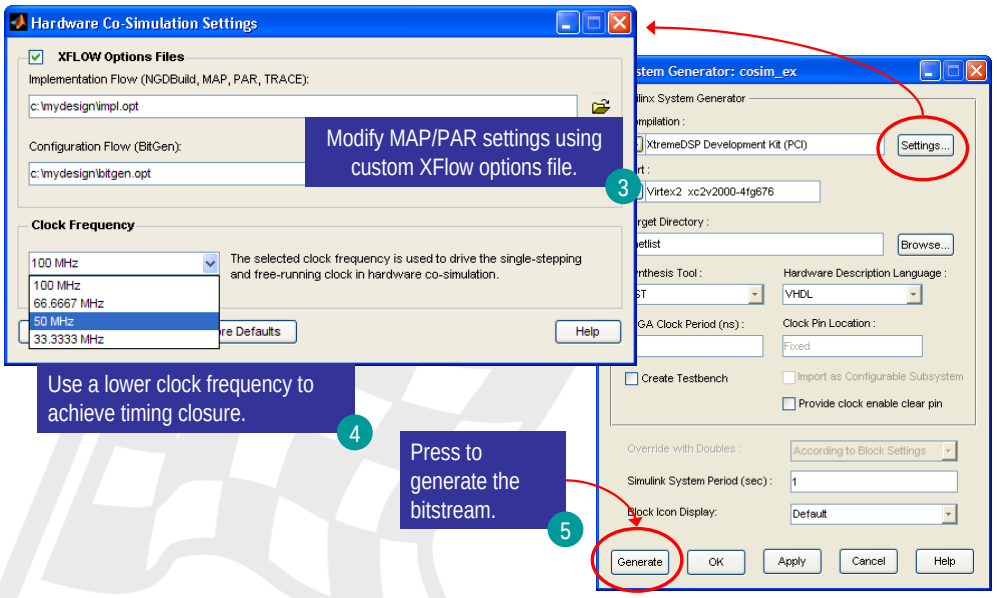
The above diagram represents a testbench that generates video frames in Simulink, transfers them to the FPGA for realtime processing, and sends back the results for display in Simulink. The ability to pass entire video frames to a portion of the model residing in the FPGA, and to retrieve the results from the FPGA for display in Simulink, makes hardware c-simulation very efficient for video system testbenches.

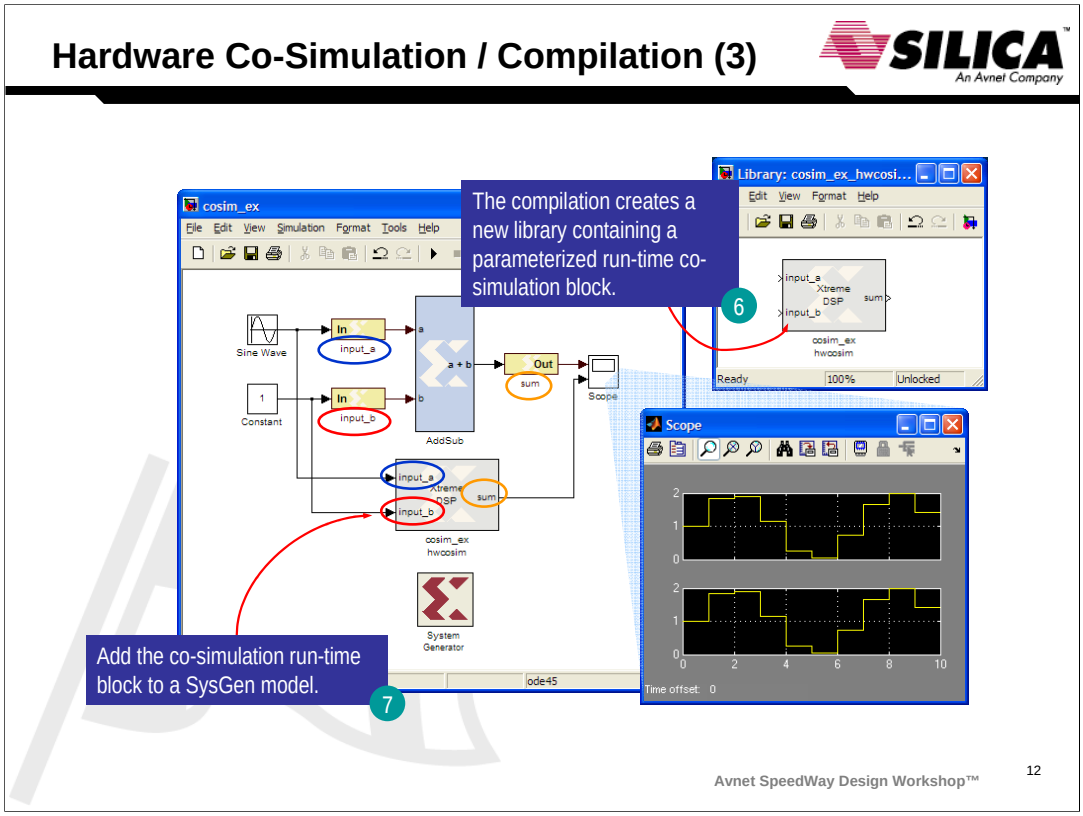
This is a test methodology between Simulink and the FPGA co-processor only; the DSP is not involved.

## Hardware Co-Simulation / Compilation



# Hardware Co-Simulation / Compilation (2)






The co-simulation block executes in the FPGA, while receiving input from Simulink and sending results back to Simulink.

## Hardware Co-Simulation / JTAG

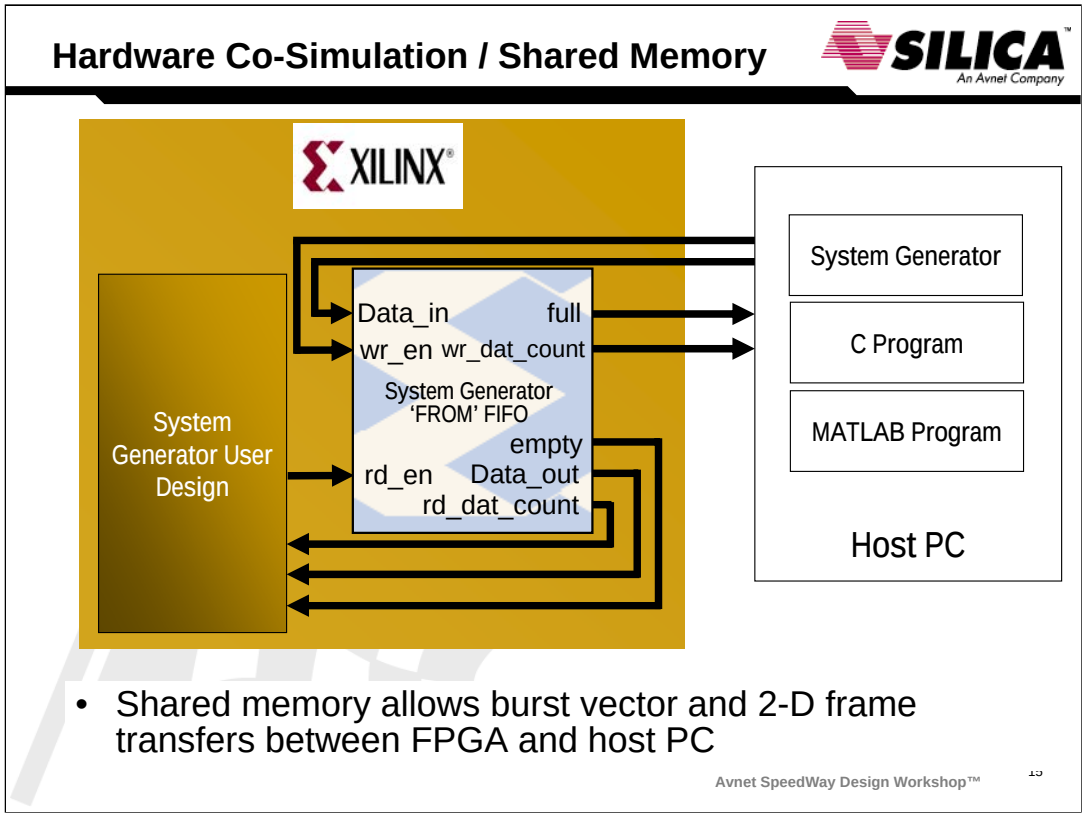
- Burst-transfer support
  - 1 Mbps down to the board
  - 0.5 Mbps back from the board
- Selectable cable speeds
- I/O buffering performance boost
  - Improvement is relative to the number of ports
- Platform USB support
- Additional boards can be added to the JTAG flow using a wizard in less than 20 minutes

## Hardware Co-Simulation / Ethernet

- Two flavors:
  - Network-based
    - Remote access
    - 10/100 Base-T network
    - Ethernet-based configuration
  - Point-to-Point
    - Requires a direct connection between host PC and FPGA
    - 10/100/1000 Base-T
    - Ethernet *or* JTAG-based (i.e., Platform USB or PC4) configuration
- Avnet Spartan-3A DSP DaVinci supports Point-to-Point 10/100 Base-T



Future support is planned for co-simulation over GigE.




Xilinx hardware co-simulation supports burst transfers between Simulink and the FPGA. Burst transfers improve simulation performance because fewer transactions are required over the course of a simulation. The general idea is to transmit data vectors (or frames) to and from the hardware platform in a single transaction (as opposed to multiple transactions when using single word read / writes). This can improve simulation performance dramatically, especially over GigE.

During a Simulink simulation cycle, shared memory allows the following sequence of events to transfer bidirectional data between FPGA and host PC:

Bursting data from host PC TO FPGA:

-Buffer a series of scalar input data values into a Simulink vector;

-Transfer the vector data to a shared memory FIFO residing on the FPGA using a burst transfer

-The FIFO allows crossing clock domains between bursty incoming data over Ethernet to the FPGA system clock

-Use the FPGA, in free-running mode, to sequentially process the entire input buffer

Bursting data from FPGA to host PC (not shown for clarity):

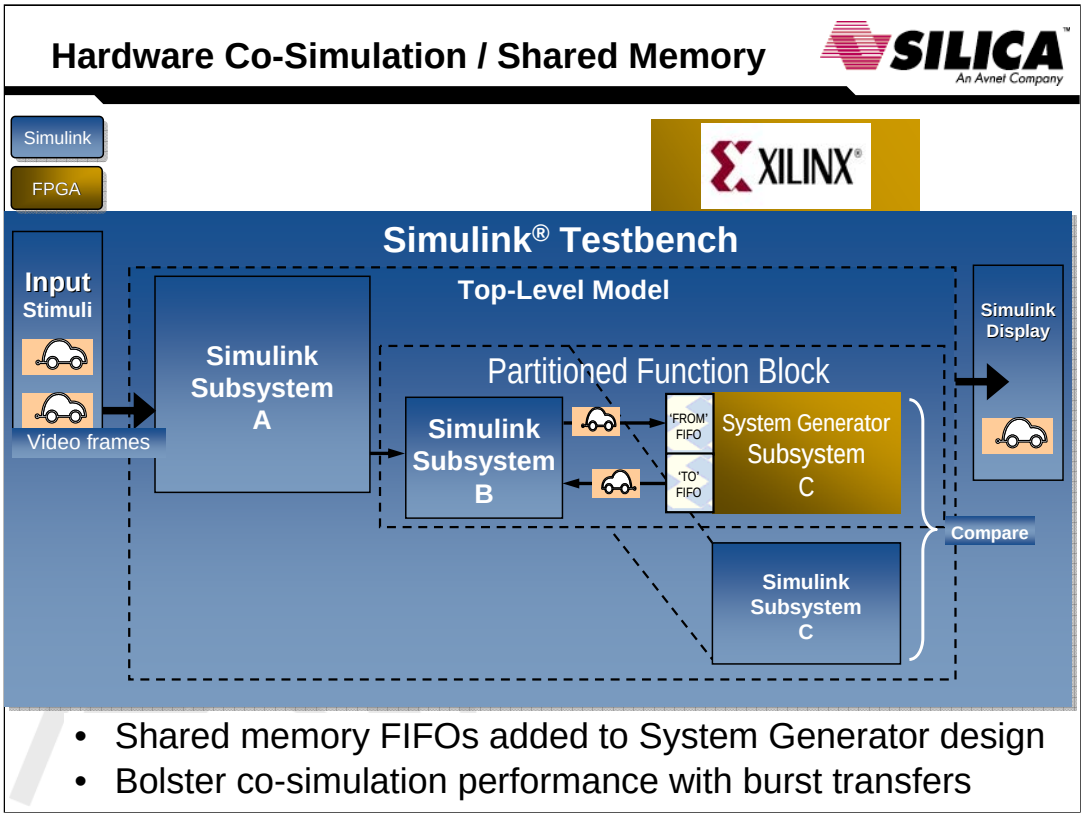
-Use the FPGA to write the processed output data into an output buffer

-Transfer the contents of the output buffer back into Simulink and reconstruct the data as a Simulink vector

-Unbuffer the vector into a series of output scalar values

In similar fashion, the host PC can exchange 2-D arrays or frames with the FPGA. This is especially useful for hardware co-simulation of video systems.

The shared memory functionality depicted in the diagram is the System Generator 'FROM FIFO' block, which receives data from the host as input stimuli to the FPGA design under test. A companion 'TO FIFO' block is available (not shown).




To use vector-based transfers in hardware co-simulation, you must augment your model with input and output buffers to stream data through the target FPGA.

< mouse click >

Shared FIFOs provide abstractions necessary to implement these buffers.

In lab 4, you will add 'To' and 'From' Shared FIFO blocks to the FPGA SAD design to transfer template and Region-of-Interest 2-D arrays from Simulink to the FPGA, and SAD results in the form of minimum SAD coordinates + 3x3 SAD neighborhood from FPGA to Simulink. It's important to note that hardware co-simulation can operate in one of 2 modes: single-stepped or 'free-running' mode. We choose to operate in free-runnng mode, meaning the FPGA is clocked by a physical oscillator on the board at it's full rate, or some convenient sub-multiple. The shared FIFOs expose 'empty/full' ports for flow control between the FIFOs and Simulink that provide a way to stall the SAD computations when the input buffer is empty. This technique allows the FPGA to execute at high speed as long as input data is available in the 'From FIFOs'.

## Summary

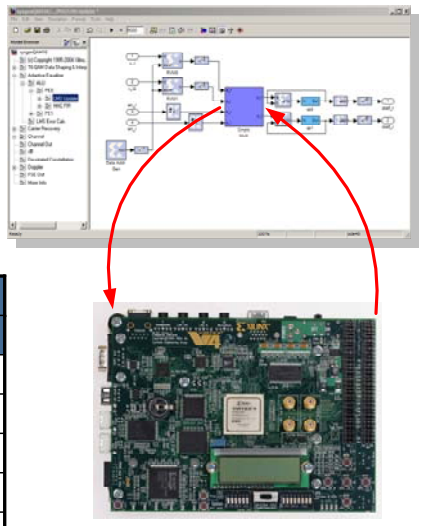
- FPGA model verification using Hardware co-simulation with System Generator and Simulink

… proceed to lab 4 *Verifying FPGA co-processor functions using hardware co-simulation*

## Reference Slides

## Accelerating Verification through Hardware

- **Hardware co-verification removes simulation bottleneck**
  - **Up to 1000x simulation performance improvement**
  - **Automates FPGA and board setup process**



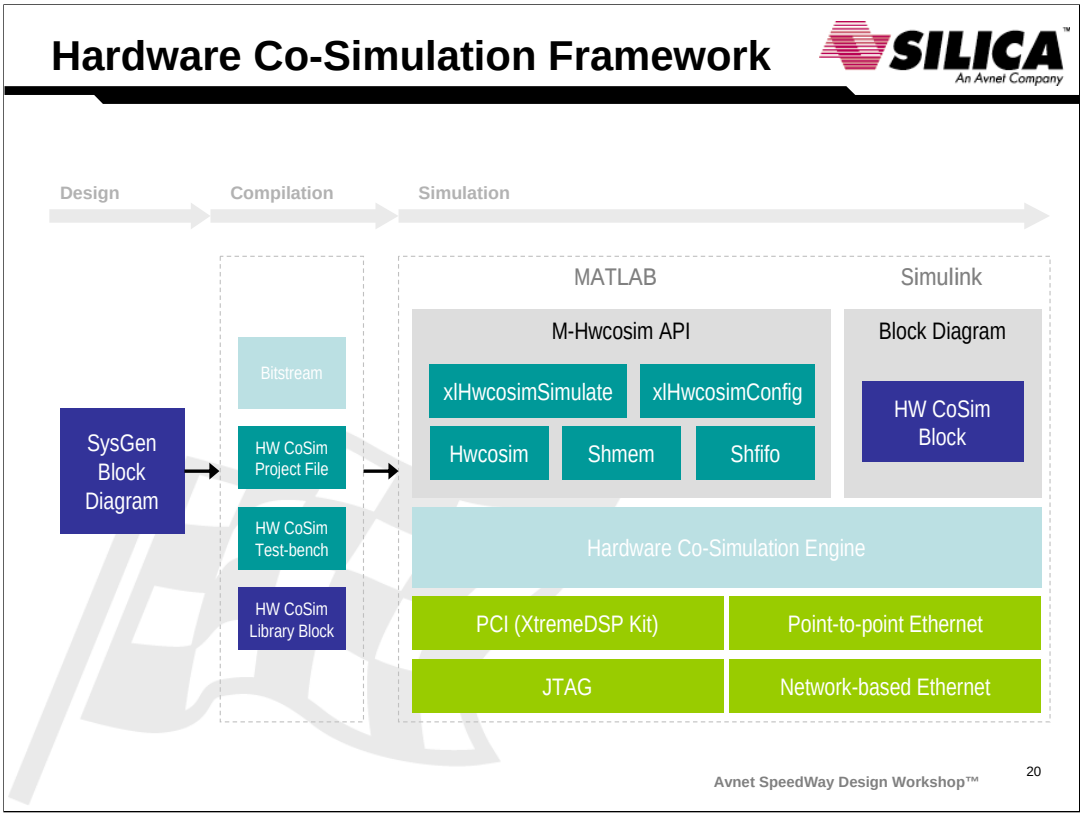
| Design | Simulation Time (Seconds) | | |
|---|---|---|---|
| | Software | HW Co-Sim | Increase |
| Beamformer | 113 | 2.5 | 45X |
| OFDM BER Test | 742 | .75 | 989X |
| DUC CFR | 731 | 23 | 32X |
| Color Space Converter | 277 | 4 | 69x |
| Video Scalar | 10422 | 92 | 113X |

Avnet SpeedWay Design Workshop™ 19

System Generator supports hardware in the loop flows to over 20 different boards
HIL is supported for both the Simulink and MATLAB modeling environments
JTAG, PCI and Ethernet based HW co-sim is supported
Additional boards can be added to the flow using a wizard in less than 20 minutes
Provides simulation increases up to 1000x

**Discovery Questions**

Will the FPGA be used for algorithm acceleration or as an FPGA prototype?
Will the FPGA be replaced in the final product?
What benefit will replacing the FPGA bring?




To conclude this discussion of Xilinx hardware co-simulation, the above diagram shows a comprehensive framework in which hardware co-simulation can be scripted in detail under control of MATLAB, or entirely under Simulink. Scripting using the MATLAB API is an advanced technique new in System Generator for DSP 10.1, beyond the scope of this course. More information on the MATLAB API is available in on-line help in Xilinx System Generator for DSP 10.1. ***Programmatic Access | M-Code Access to Hardware Co-Simulation***

We will focus exclusively on hardware co-simulation in Simulink for the remainder of the course.