



**Getting high with a little help
from my friends:
Configurable
processor interoperability with
ESL tools**

Grant Martin, Chief Scientist
Synopsys Interoperability Forum
5 November 2009: 1315-1445

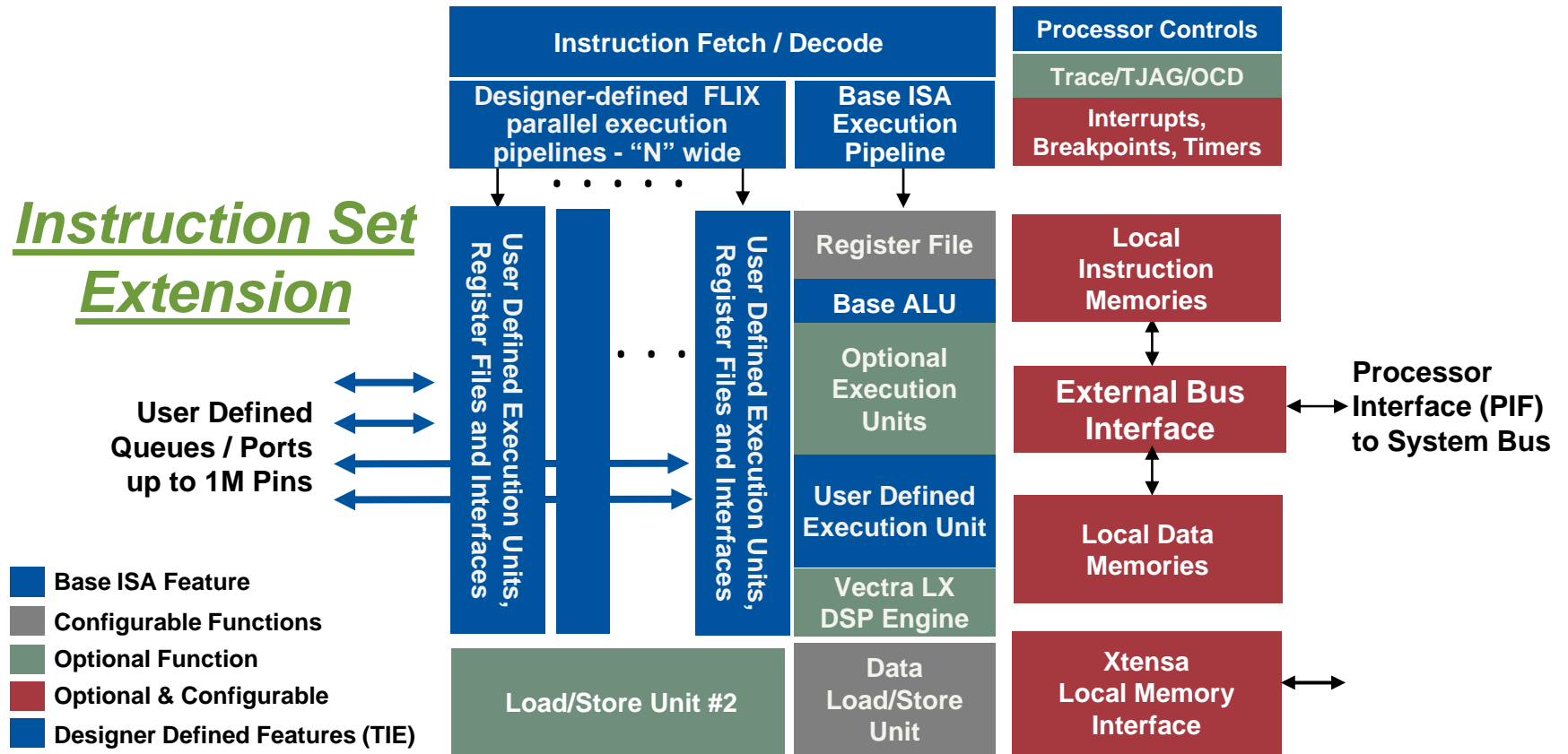
Outline

- ▬ Configurable, extensible processors
- ▬ Instruction set simulators
- ▬ ESL models and their use
- ▬ Challenges of 3rd party ESL integration
- ▬ Ecosystem
- ▬ ISS-Innovator integration

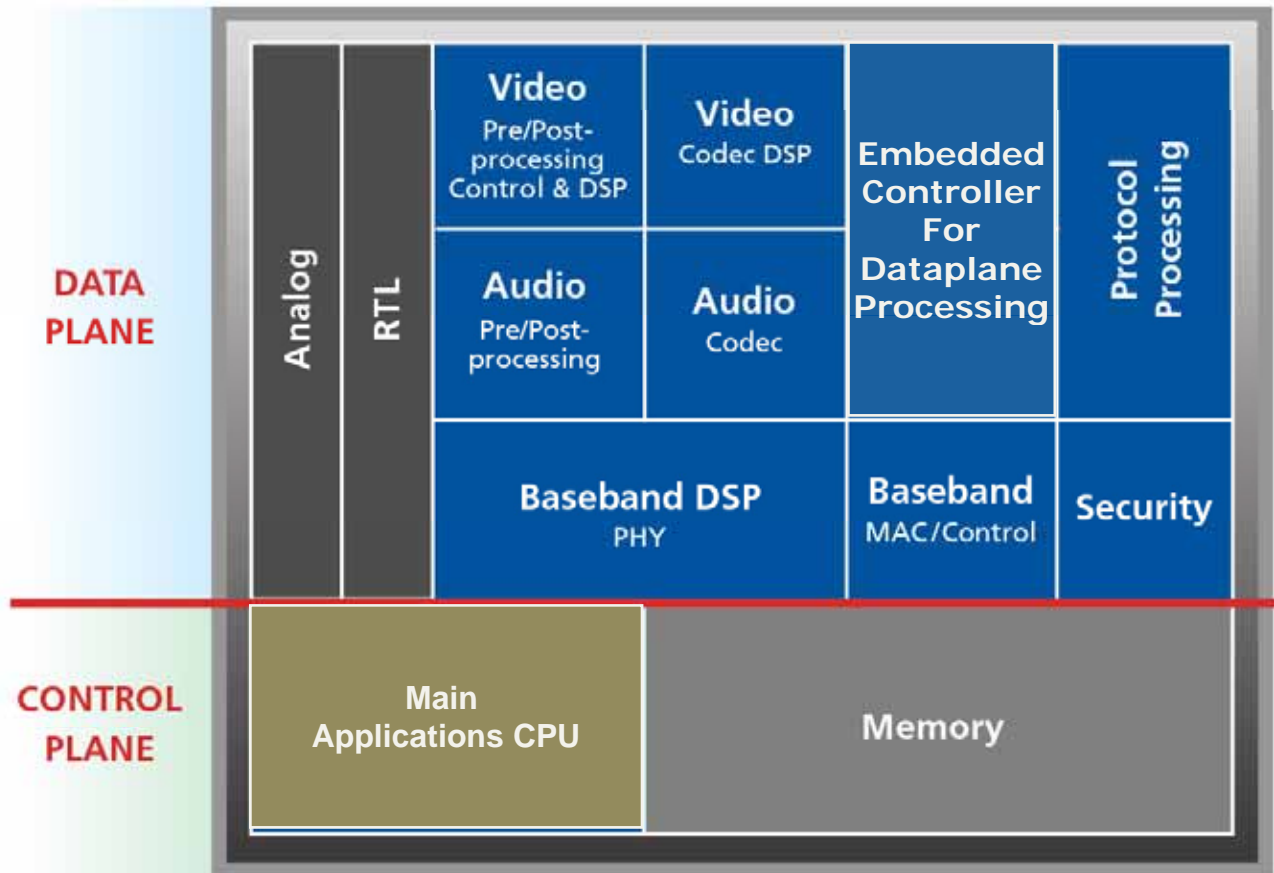
Configurable extensible processors

Configuration

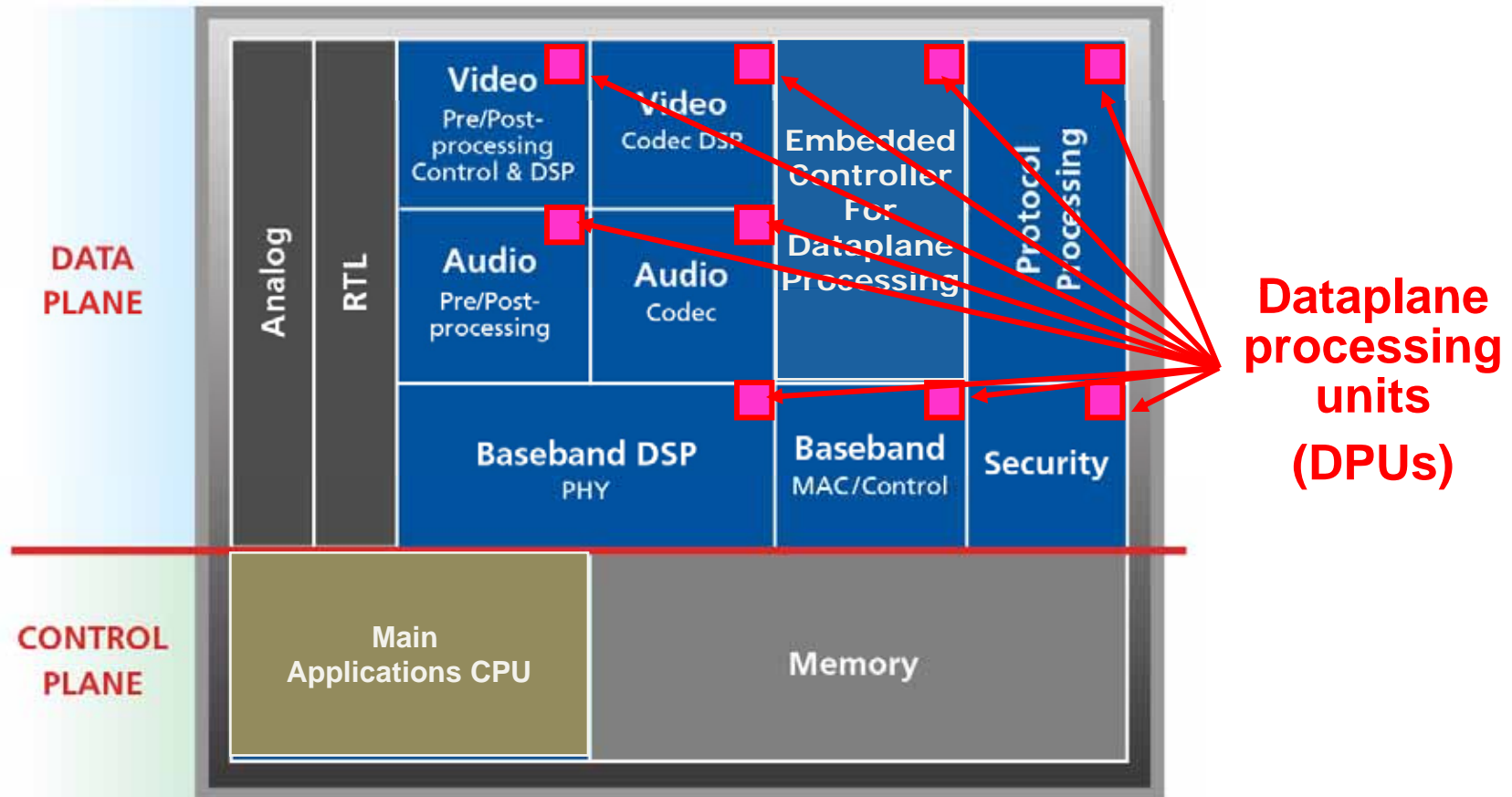
Instruction Set Extension



Dataplane Processing



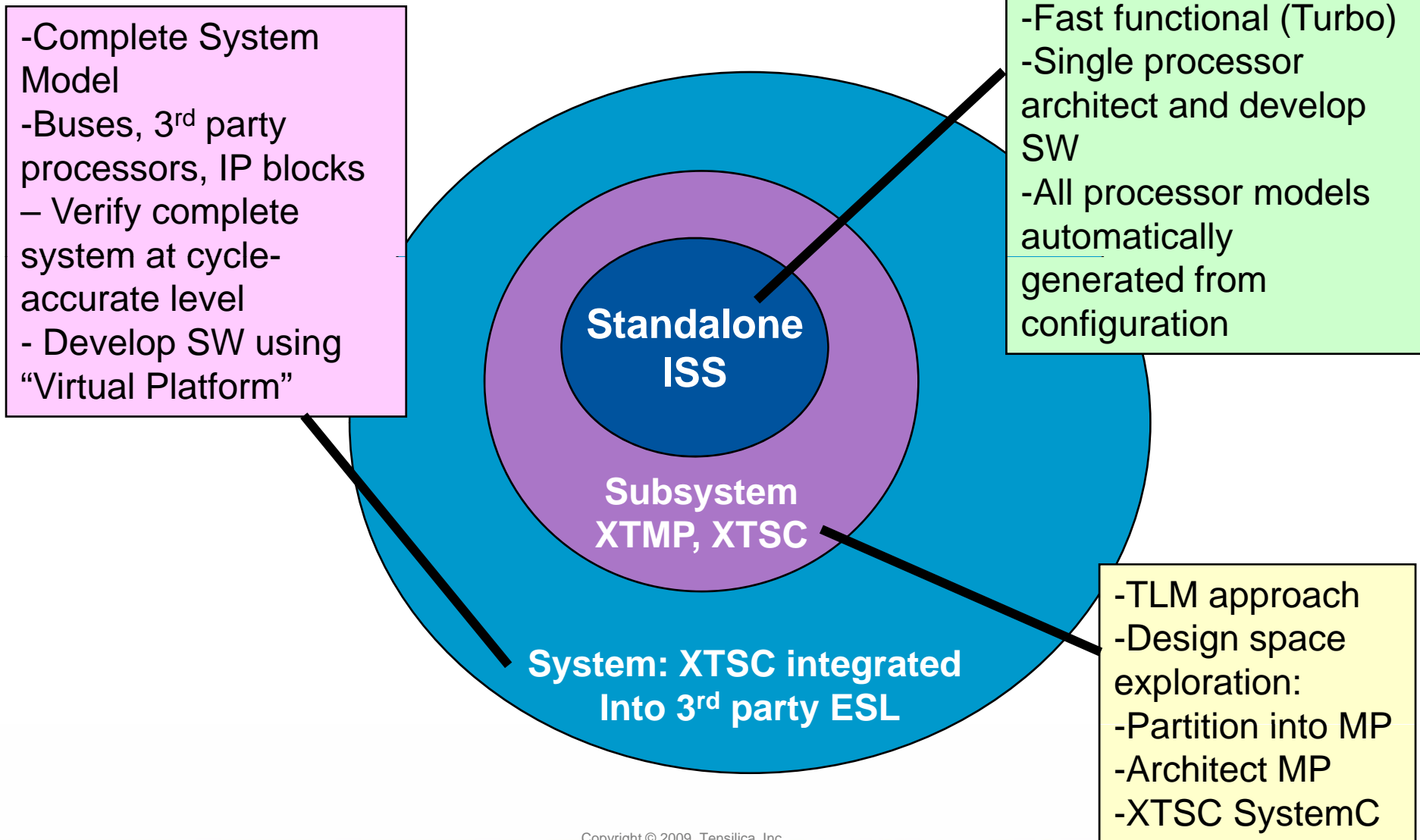
Dataplane Processing Units



Designing with processors is *System Design*

- Start with applications code, usually in C/C++
 - May be generated from algorithmic modelling tool – e.g. Matlab/Simulink, Synopsys System Studio, etc.
- Map it to a generic processor – e.g. a simple Tensilica DPU
- Profile it with the ISS in cycle-accurate mode
- Identify hot spots in the code
 - e.g. cycle-intensive loop nests in data crunching code
- Experiment with compiler optimisation levels
 - Down to `-O3 -SIMD`:
 - Inference of DPU vectorised and specialised instructions (multi-way MACs)
 - Automatic vectorisation
 - Automated feedback to improve vectorisation
- Identify opportunities for instruction extensions and other configuration parameters
 - SIMD, instruction fusions, VLIW (FLIX)
 - Cache sizing, use of local Instruction and Data memories
 - TIE ports and queues direct to the datapath
- Modify configuration and extend the processor ISA until optimisation targets achieved
 - All done at the System Level
- If a single processor won't do – design multiple heterogeneous ASIPs

Tensilica ESL models and their Uses



Why third party ESL tool integrations?

- It's not just a Tensilica world
 - Users often will build systems with multiple processors from multiple vendors
- Even if all processors were Tensilica:
 - Users may be using standard buses such as AHB, APB, AXI
 - Tensilica does not offer a set of standard bus models
 - XTSC offers a set of generic components (e.g. routers, arbiters, memories) but these use XTSC protocols, not standard bus protocols
 - We do not offer a rich library of third party IP models as are often found with commercial ESL tools
 - Users may have models conforming to other standards:
 - E.g. OSCI TLM1, TLM2, OCP-IP
 - Users may want graphical analysis capabilities not offered in our system modelling environments
- Users may want to control a complex multi-vendor multi-processor debugging environment using a single debug 'cockpit'

Not all flowers: challenges of integration with 3rd party ESL tools

— Standards

- We developed XTSC after TLM 1.0 and before TLM 2.0
- We develop our own approaches for transactions, fast models, direct memory access, etc.
- Retrofitting costs – a cost we are in general unable to bear

— OSCI TLM 2.0

- Limitations: Bus locking, DMI dynamic invalidation, lack of agreed cycle-accurate use model

— Various use models

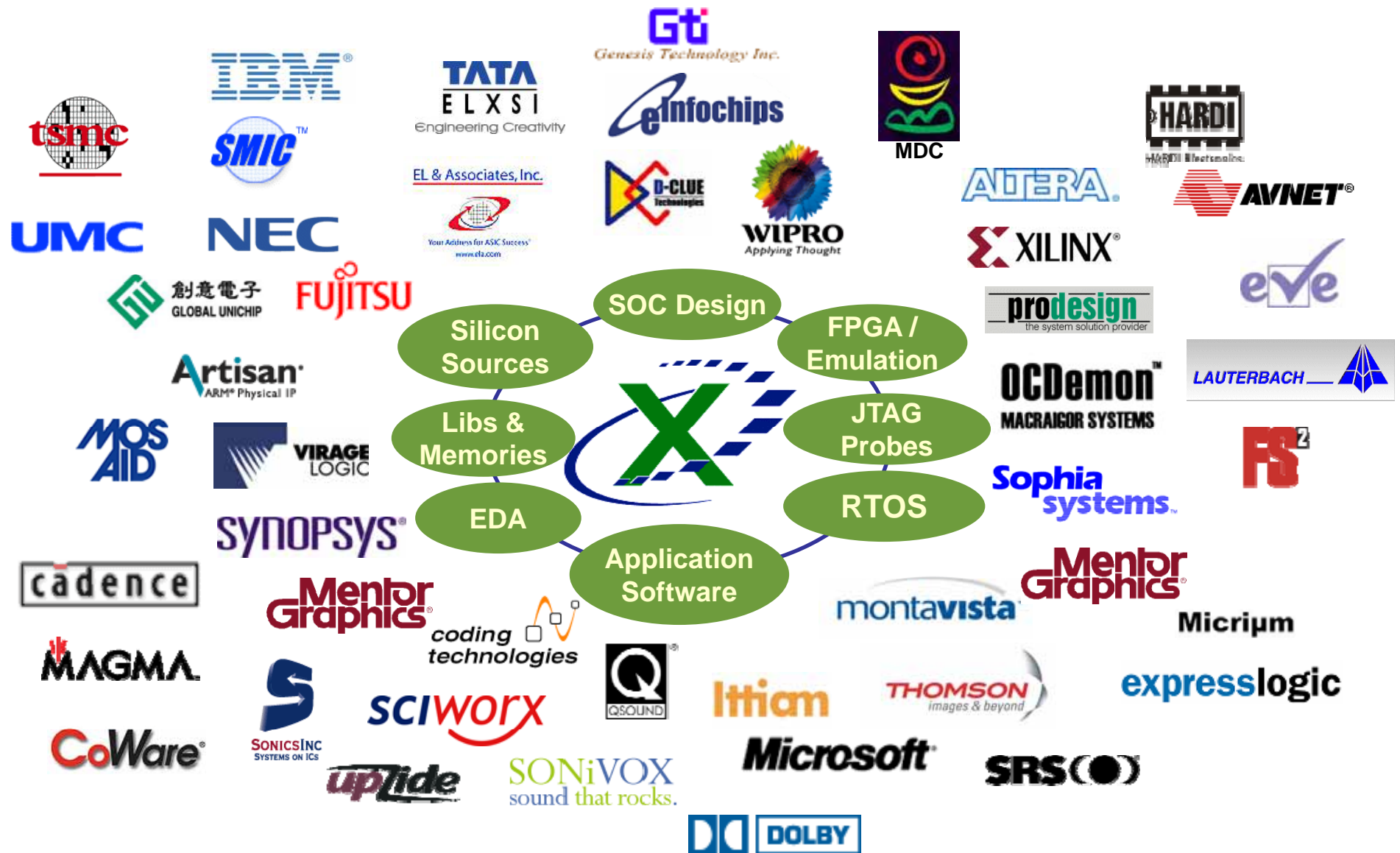
- Mapping Untimed, Loosely Timed, Approximately Timed and ????? Into our user's world:
 - Fast Functional (TurboXim) and Cycle-accurate
 - Accuracy of anything “approximate” or mixing cycle-accurate and “approximate” is indeterminate
- Virtual platform or Virtual prototype use model becoming better established: good match to TurboXim

Not all flowers: challenges of integration with 3rd party ESL tools, continued

- Challenges of configurable, extensible IP
 - The issue is the interfaces:
 - Local memory interfaces (IRAM, IROM, DRAM, DROM, XLMI)
 - Standard memory-mapped interface via the PIF (often bridged to AHB or AXI)
 - TIE ports, queues and lookup interfaces
 - Instruction extensions are in general “internal” to the ISS libraries
- Three different approaches (at least) to integration:
 - Hand written wrapper
 - Difficult with many possible configurations and interfaces
 - Automatically generated wrappers
 - XTMP has some introspection
 - XTSC has considerable extra introspection
 - Best solution to support ALL configurations
 - Toolkit approach via specific adaptors for the various interfaces
 - Also a viable approach to support ALL configurations
 - Requires a little more work from the users



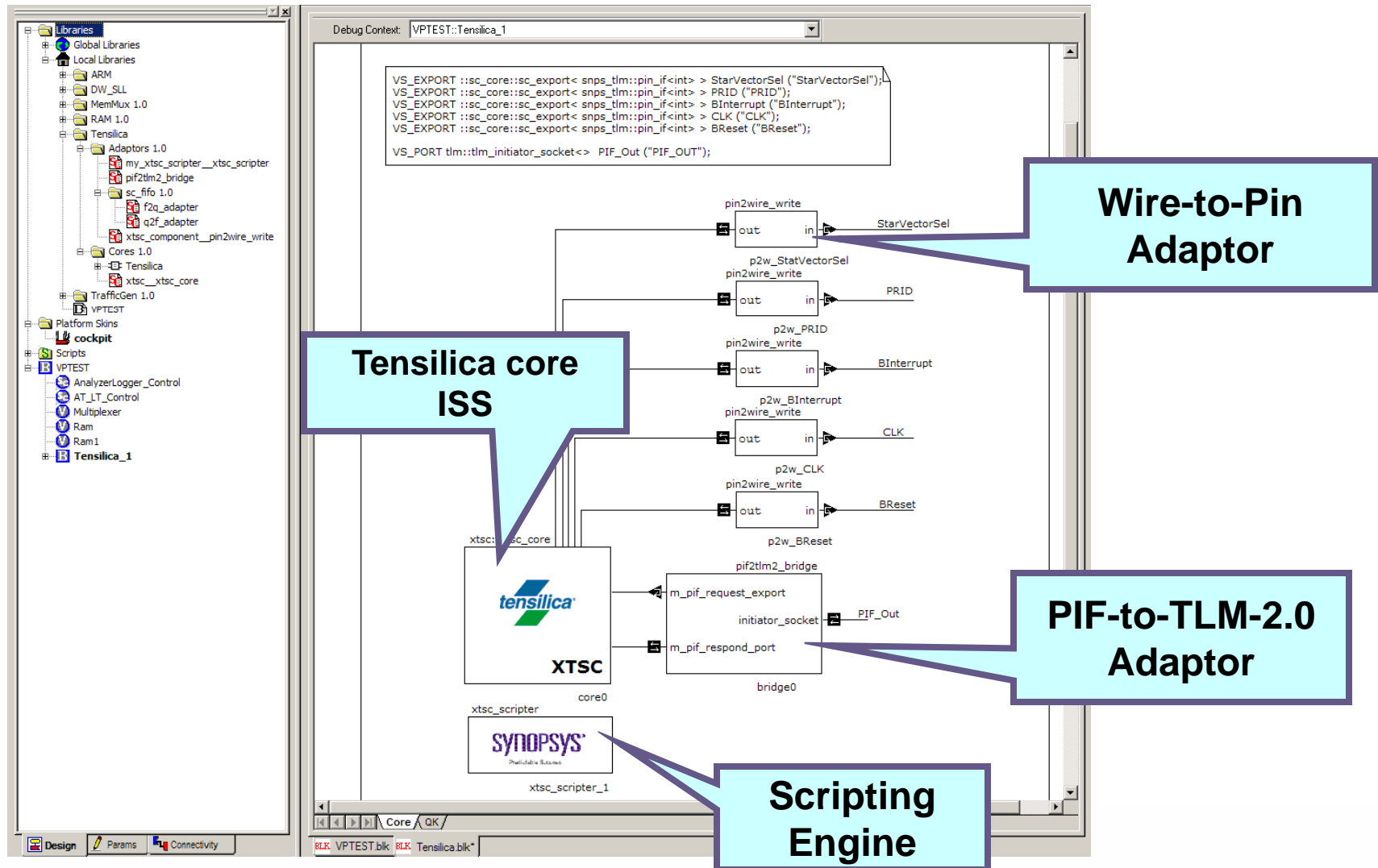
Tensilica Ecosystem (2007)



Tensilica ESL Ecosystem: Customer-driven



Tensilica DPU ISS in Synopsys Innovator



Tensilica DPU in Synopsys Innovator



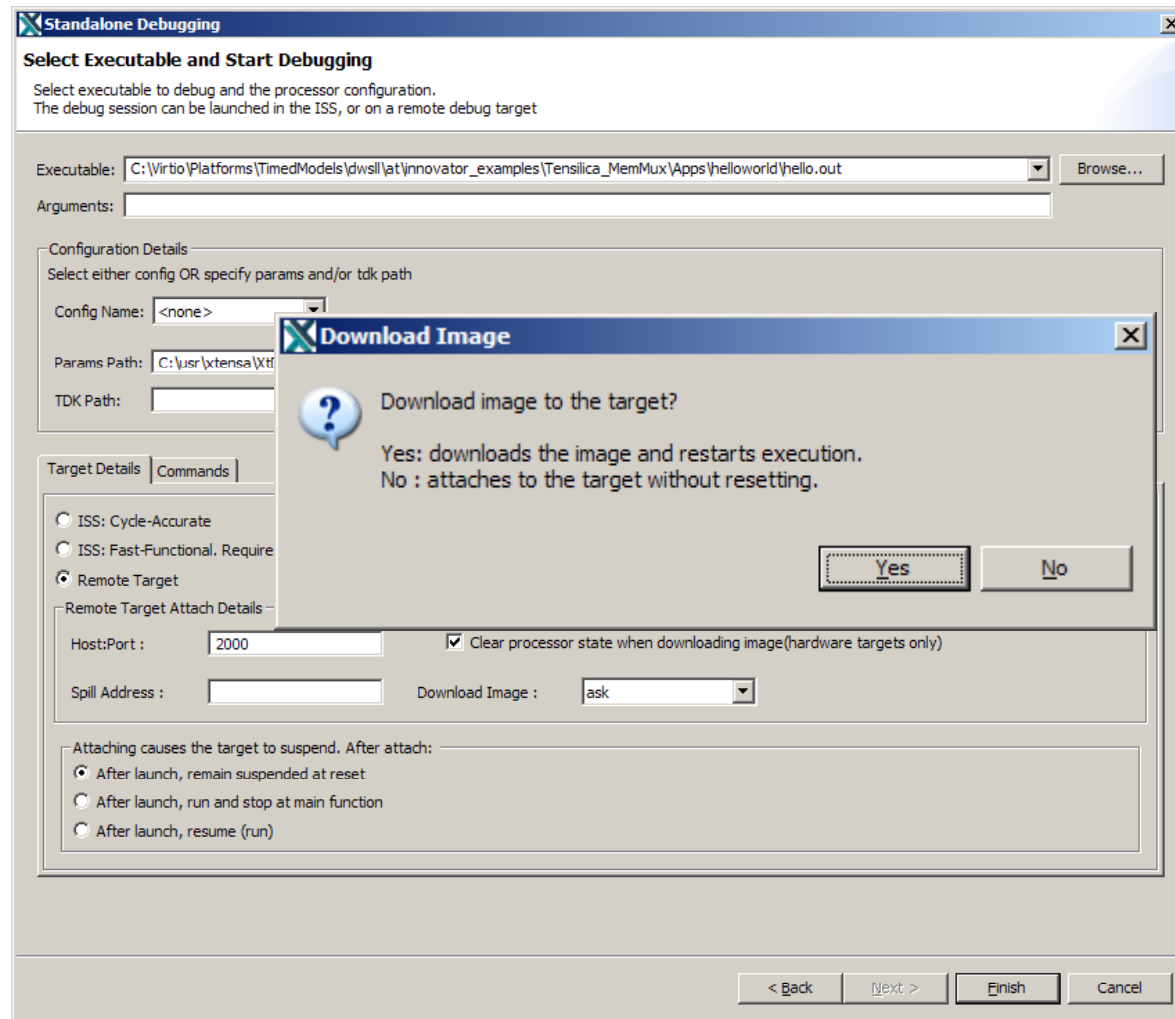
The screenshot displays the Synopsys Innovator IDE interface. The main window shows a block diagram of the Tensilica Diamond Standard Series 545CK processor. The diagram includes components such as I-RAM, D-RAM 0, D-RAM 1, Fetch/Decode, RFO I/F, Interrupt Unit, Timers, On-Chip Debug, Trace Port, Vector Reg File, Reg File, ALU, MACs, Branch, Vntrbi Accel, Mul 16, Ld/St #1, Ld/St #2, System I/F, PIF, Bridge, and MainMemory. External interfaces include CLK, BReset, BInterrupt, PRID, StarVectorSel, and PIF_Out.

A callout box on the left contains the text: "Information for debugger attachment".

The console window at the bottom shows the following output:

```
Connected to platform at port# 5343
sc_top_inst.VPTEST.Tensilica_1.core0: SOCKET:20000
NOTE: sc_top_inst.VPTEST.Tensilica_1.core0 - 0.0/000: Debug info: port=20000 wait=true ()
Platform Launch took 17 seconds
XTMP_LIST_CORE_INFORMATION START 2.0
SYNCHRONIZED_RUN=0
core0: filip-m65;20000;wait;C:\Virtio\Platforms\TimedModels\dws11\at\innovator_examples\Tensilica_MemMux;C:\usr\xtensa\XtDevToolsI
XTMP_LIST_CORE_INFORMATION END
```

Tensilica DPU in Synopsys Innovator: Debugger attach



Tensilica DPU in Synopsys Innovator: Debugging

The screenshot displays the Synopsys Innovator IDE interface. On the left, a project tree shows the hierarchy of components, including 'VPTEST' and its sub-components like 'AnalyzerLogger_Control', 'AT_LT_Control', 'Multiplexer', 'Ram', and 'Ram1'. The main workspace shows the 'Debug Context: VPTEST' with a detailed block diagram of the 'Tensilica Diamond Standard Series 545CK' processor. The diagram includes components such as 'Fetch/Decode', 'Vector Reg File', 'Reg File', 'System I/F', 'I-RAM', 'D-RAM 0', 'D-RAM 1', 'Interrupt Unit', 'Timers', 'On-Chip Debug', 'Trace Port', 'Ld/St #1', 'Ld/St #2', 'ALU', 'Branch', 'Mul 16', 'Viterbi Accel', '8 16x16 MACs', 'PIF', 'Bridge', and 'MainMemory'. External signals like 'CLK', 'BReset', 'BInterrupt', 'PRID', and 'StarVectorSel' are also shown.

At the bottom, the console window shows the following output:

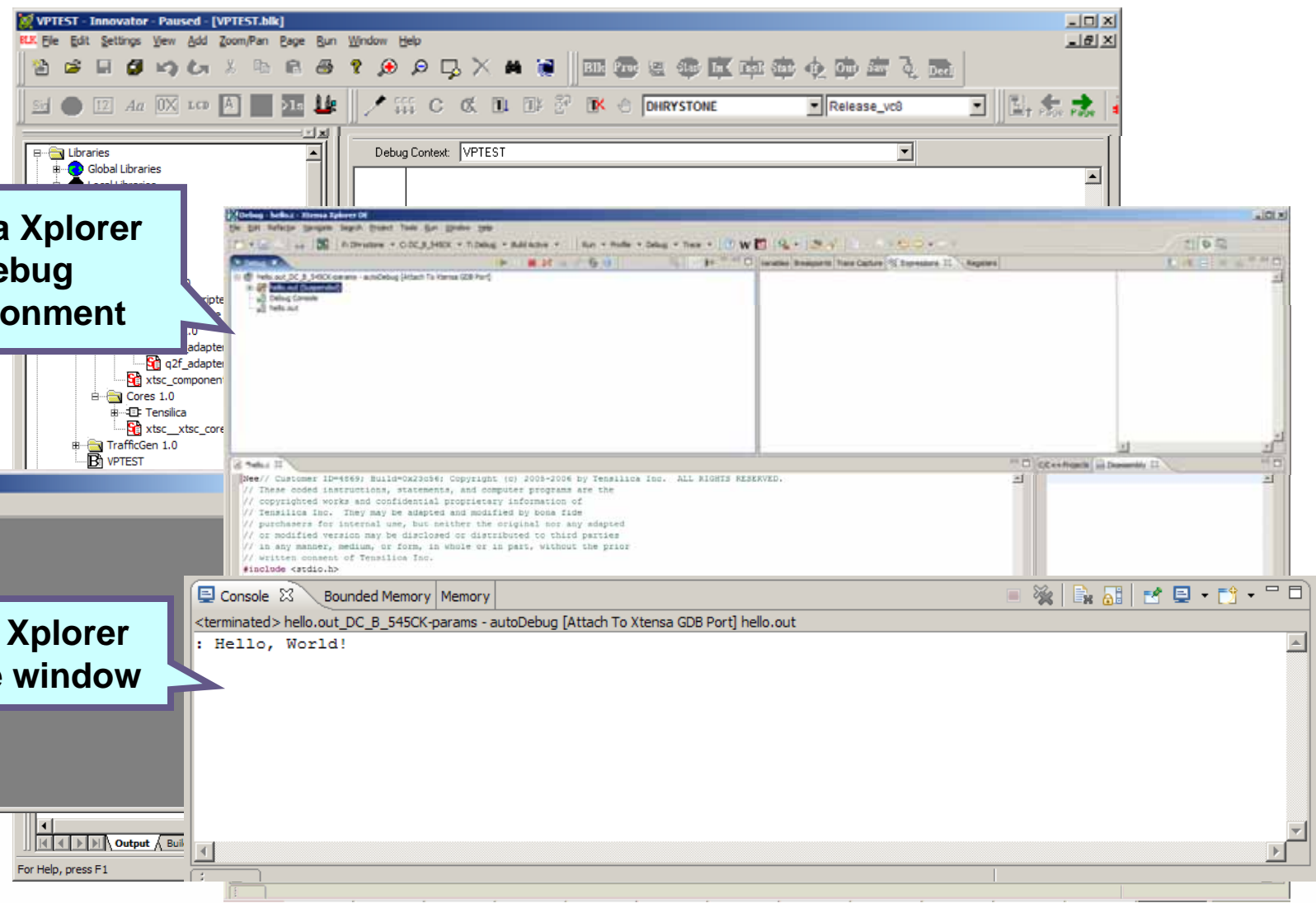
```

Connected to platform at port# 5343
sc_top_inst.VPTEST.Tensilica_1.core0: SOCKET:20000
NOTE sc_top_inst.VPTEST.Tensilica_1.core0 - 0.0/000: Debug info: port=20000 wait=true ()
Platform Launch took 17 seconds
XTMP_LIST_CORE_INFORMATION START 2.0
SYNCHRONIZED_RUN=0
core0: filip-m65;20000;wait:C:\Virtio\Platforms\TimedModels\dws11\at\innovator_examples\Tensilica_MemMux;C:\usr\xtensa\XtDevToolsI
XTMP_LIST_CORE_INFORMATION END
  
```

Tensilica DPU in Synopsys Innovator: Debugging

**Xtensa Xplorer
Debug
Environment**

**Xtensa Xplorer
console window**



Future evolution of such integrations

- OSCI TLM 2 should be standardised via IEEE 1666:
 - Would be nice to deal with outstanding issues:
 - Bus locking via modified standard payload
 - Support for static DMI option (without dynamic invalidation being necessary)
 - Defined Cycle-accurate mode in TLM2 (should be possible using 4/6 phase concepts)
- Tensilica looking at supporting PIF to TLM2 in 2010, and Dynamic DMI invalidation
- OSCI CCI (Configuration, Control and Inspection) Working Group may define some useful standards to adopt
 - Biggest issue with OSCI is its “closed-shop” mentality
 - Needs to actively solicit opinion across the industry including from non-members
 - It needs to understand the retrofitting issues and the cost of this to come up with credible proposals

Conclusion

- ▀ Third party ESL tools are a vital part of designing with configurable and extensible processors
 - An important part of our Ecosystem
- ▀ User demand has given us and third parties the impetus to collaborate on integrations
- ▀ These are in active use by users today
- ▀ Support for more advanced capabilities will grow as the whole area matures and standards develop further