



The Engine of SOC Design

**OSCI DAC 2007 Meeting - Panel:
Virtual platform and ESL IP exchange: Real world
challenges and what's expected from OSCI TLM 2.0**

Grant Martin

Chief Scientist, Tensilica

Monday 4 June 2007: 1200-1400



Tensilica and ESL Modelling

- Configurable, extensible processor IP
- For many years - system-level modeling with XTMP
 - C++ with a C-based API
 - Our TLM – no accepted standards
 - Cycle-accurate ISS
 - Late 2006 – fast functional mode (up to 100X)
- Late 2006 - SystemC modeling environment, XTSC
- Many interfaces
 - PIF – bus-type, memory-mapped
 - 7 local memory interfaces
 - Nearly unlimited TIE port, queue and lookup interfaces
- Interface XTSC to commercial 3rd party SystemC-based ESL tools
 - Generate adaptors: TLM_{us} to TLM_{them}
 - All TLMs different
 - Fast Functional ISS - “Virtual System Prototype” modeling



Tensilica ESL models

- Complete System Model
- Buses, 3rd party processors, IP blocks
 - Verify complete system at cycle-accurate level
- Develop SW using "Virtual Platform"

Standalone ISS

**Subsystem
XTMP, XTSC**

**System: XTSC integrated
Into 3rd party ESL**

ISS

- Cycle-accurate
- Fast functional
- Single processor architect and develop SW
- All processor models automatically generated from configuration

- TLM approach
- Design space exploration:
 - Partition into MP
 - Architect MP
 - XTSC SystemC



Integration into 3rd party SystemC-based ESL tool

The screenshot shows the CoWare Analysis interface integrated with a third-party SystemC-based ESL tool. The main window displays a complex SystemC block diagram with various components and connections. A red arrow points from a specific node in the diagram to a function trace window.

Function Trace for core HARDWARE.i_extsc_Max_le_32

id	name
25	write
26	_sprintf
27	_sprintf
28	vfprintf
29	_vfprintf_r
30	cvt
31	exponent
32	_wvrtomb_r
33	wvrtomb
34	wcrtomb_r

Console Output:

```
Reading Import Wire IMPWIRE: Value read: 0x7
Reading Import Wire IMPWIRE: Value read: 0x8
Reading Import Wire IMPWIRE: Value read: 0x9
Reading Import Wire IMPWIRE: Value read: 0xa
Got interrupt 4
Reading Import Wire IMPWIRE: Value read: 0xb
Reading Import Wire IMPWIRE: Value read: 0xc
Got interrupt 5
Reading Import Wire IMPWIRE: Value read: 0xd
Reading Import Wire IMPWIRE: Value read: 0xe
Reading Import Wire IMPWIRE: Value read: 0xf
interrupt_count = 2
Got interrupt 7
SystemC: simulation stopped by user.
```

File **View** **Help** **Simulation** **Plugins** **Help** **<No Perspective Active>** **Search:**

1:34 PM

extsc_2p_mem
extsc_lookup
extsc_master
extsc_master_rec
extsc_Max_le_32
extsc_mem
extsc_q
extsc_support
extsc_wire
extsc_wire_read
extsc_wire_sourc
extsc_wire_write
Memory
PIF2AHB_Bridge
ResetGenerator
System_RAM
System_ROM
Transactor_AHE
AHB
AHBArbitrator
AHBInitiator
AHBLiteTarget
AHBTARGET
Default
Default_Indexed_L
Default_UT
DefSlaveTarget
lookup
mem_request
pif_request_only
pif_response
PV
q_pop
q_push
REMAP
RESET
wire_read
wire_write

master_req
master_resp
extsc_Max_le_32
Function Trace
BInterrupt_exp
control
dram0_req
dram1_req
dram0_resp
dram1_resp
drom0_req
drom0_resp
EXPSTATE
IMPWIRE
inbound_pif_req
inbound_pif_resp
IPQ1
IPQ2
IPQ3
iram0_req
iram0_resp
iram1_req
iram1_resp
iram0_req
iram0_resp
lookup_ram
lut
lut_nordy
onebit
OPQ1
OPQ2
OPQ3
pif_req
pif_resp
reset
status
xmi_req
xmi_resp
extsc_support
extsc_xmi



Tensilica TLM Evolution.....TLM 2.0

- Concerns:
 - Too late?
 - Ad hoc integrations into 3rd party ESL tools already done
 - Little incentive to rework
 - Priorities: For us, cycle-accurate much more important than PV+T
 - PV+T prone to misunderstanding especially by SW developers
- Expectations:
 - Final TLM2 needs to help industry close current gaps in interoperability
 - Cycle-accurate - major importance
 - Well-defined **analysis** and **debug** interfaces
 - OSCI TLM group needs to educate users on abstractions
 - Possibly support research on PV+T estimators and system modeling
 - Scope of TLM needs to be better defined
 - Many people assume “interoperable” TLM implies interoperable **bus** model implementations