

OpenSPARC Slide-Cast

In Twelve Chapters

Presented by OpenSPARC designers, developers, and programmers
to guide users as they develop their own OpenSPARC designs and
to assist professors as they teach the next generation

This material is made available under Creative Commons Attribution-Share 3.0 United States License





Chapter Seven OPENSPARC FPGA IMPLEMENTATIONS

Thomas Thatcher Staff Engineer OpenSPARC Evangelist Sun Microsystems







Agenda







OpenSPARC T1 Hardware Package

- Documentation
 - doc/
- Full RTL
 - design/sys/iop
- Simulation scripts & full verification suite
 - Simulation Environment files verif/env - verif/diag Test lists and assembly code for tests

Synthesis scripts (Design Compiler, Synplicity, and XST)

- design/sys/synopsys design/sys/synopsys
 design/sys/synplicity
- design/sys/xst

Synopsys synthesis scripts Synplicity FPGA synthesis scripts Xilinx XST synthesis scripts

Xilinx EDK Project for full system on FPGA

design/sys/edk





RTL Hierarchy

Top level block for FPGA implementation

> design/sys/iop/iop_fpga.v

RTL Path

- > design/sys/iop/
 - I2b/rtl Level-2 Cache
 - ccx/rtl
 Cache Crossbar
 - ccx2mb/rtl
 Cache Crossbar to MicroBlaze adapter
 - fpu/rtl
 Floating-Point Unit
 - dram/rtl
 DDR2 DRAM controller

SPARC Core

www.opensparc.net

> design/sys/iop/sparc/

- ffu/rtl

– tlu/rtl

- rtl/ Top-level code
- ifu/rtl
 Instruction Fetch Unit Includes ITLB and I-Cache
- exu/rtl Execution Unit
- Isu/rtl
 Load-Store Unit Includes DTLB and D-Cache
 - Floating-Point Front-end Includes FP register file
 - Trap Logic Unit





Verification Environments

• Core1: Simulate a single SPARC core

- One SPARC core
- Level 2 cache
- Memory Controller
- Memory model

• Thread1: Simulate one single-thread SPARC core

- Same as Core 1, except that SPARC core is single thread

Chip8: Simulate the entire OpenSPARC T1

- Eight SPARC cores
- Level 2 Cache
- I/O Subsystem
- Memory controller
- Memory model

• For the netlist (gate level) simulation, use vector playback methodology (provided in DV guide)

www.opensparc.net





Synthesis Scripts

Scripts to run a synthesis tool

tools/bin/rsyn
 tools/bin/rsynp
 tools/bin/rxil
 Run Synopsys Design Compiler
 Run Synplicity FPGA Synthesis
 Run Xilinx XST FPGA synthesis

Input scripts for the synthesis tools

- design/sys/synopsys
 lnput scripts for Design Compiler
 design/sys/synplicity
 lnput scripts for Synplicity
 lnput scripts for XST
- Example Synthesis command:
 - rsynp sparc

Synthesize the SPARC core with Synplicity





Agenda



- **Download Contents**
 - > Simulation
 - Synthesis

Implementation of an OpenSPARC T1 system on FPGA





OpenSPARC Verification Environment

- Every test is written as an assembly language program
 - Must include "hboot.s" for reset code
- The SPARC assembler is run to generate an executable
- The ELF executable is converted to a memory image
 - > Virtual memory tables are added at this point
- The memory image is loaded into the memory model
- The simulation starts with a reset
- An architectural simulator (SAS) runs in lock-step with the RTL, checking the state at the end of each instruction





Typical Test Flow

- The Reset pin is asserted and the chip is initialized.
- The I/O block sends a Power-On Reset (POR) interrupt to a core (usually core 0, thread 0)
- The core wakes up, and begins fetching from address 0xfff0000020 (POR trap handler in the Boot ROM)
- Reset code will turn on caches, TLBs
- Control will be passed to the user code for the test





Test Completion

- At the end of the test, code will perform a software trap
- The trap is to one of two locations
 - > GOOD_TRAP address indicates success
 - > BAD_TRAP address indicates a problem
 - > NOTE: There are two or three addresses for GOOD_TRAP and two or three for BAD_TRAP
 - > User trap table, Supervisor trap table, Hypervisor trap table
- Example:

good_end: ta T_GOOD_TRAP nop





How to Run Diagnostic Tests

- Simulations run with sims
 - > Full Regresssion:
 - % sims -sim_type=vcs -group=core1_mini
 - > Common regressions
 - thread1_mini thread1_full
 - core1_mini core1_full
 - chip8_mini chip8_full
 - > Reporting Results:
 - % regreport \$PWD/2006_01_25_0 > report.log
 - > Single Test:
 - % sims -sim_type=vcs -sys=core1 -sas verif/diag/assembly/arch/exu/exu_add.s





Simulation Output

- A directory is created for each test
 - > Important Files:
 - diag.s Copy of the original assembly language file
 - diag.exe ELF executable of the test created by assembler
 - mem.image Memory image of the test, including virtual memory tables
 - symbol.tbl Symbol table for the elf executable
 - sim.log Simulation log file
 - sims.log Log from sims program: including simulation log
 - sas.log
 Log file created by the architectural simulator

Simulation Log file output

- Time 47800 Test reached GOOD_TRAP





Agenda



- Download Contents
 - Simulation
 - > Synthesis

Implementation of an OpenSPARC T1 system on FPGA





SPARC Core Options

- The SPARC core contains the following options
 - > Set by compiler defines
- Options:
 - > FPGA_SYN
 - Required for all other options
 - > FPGA_SYN_1THREAD
 - > FPGA_SYN_NO_SPU
 - > FPGA_SYN_8TLB
 - > FPGA_SYN_16TLB

Optimize code for FPGA

Create a single-thread core Do not include the SPU Reduce # of TLB entries to 8 (from 64) Reduce # of TLB entries to 16

> CONNECT_SHADOW_SCANConnect shadow scan in RTL





Running Synplicity FPGA Synthesis

- Setting Compile Options:
 - > Edit file:
 - design/sys/synplicity/env.prj
 - > Add Line:
 - set_option -hdl_define -set "FPGA_SYN FPGA_SYN_1THREAD"
- Synthesis Command:
 - %rsynp -all
 - %rsynp -device=XC5VLX110 sparc
- Synthesis Output
 - design/sys/iop/sparc/synplicity
 - XC5VLX110/
 - sparc.edf
 - sparc.srr

Synthesize all blocks

Synthesize sparc core, specify device

Directory where output files found Directory for target device EDIF output netlist Synthesis log file





Running XST Synthesis

Setting Compile Options

- > Edit File:
 - design/sys/iop/include/xst_defines.h
- > Add Lines:
 - `define FPGA_SYN
 - `define FPGA_SYN_1THREAD
- Synthesis Command:
 - %rxil -all
 - %rxil -device=XC5VLX110 sparc
- Synthesis Output
 - design/sys/iop/sparc/xst
 - XC5VLX110/
 - sparc.ngc
 sparc.v
 - sparc.srp

Synthesize all blocks

Synthesize sparc core, specify device

Directory where output files found

Directory for target device Xilinx/Verilog output netlists Synthesis Log file





Agenda

OpenSPARC T1 hardware package

Download Contents

Simulation

Synthesis

> Implementation of an OpenSPARC T1 system on FPGA





Why FPGAs

- Community Requested it
 - > Primarily academic community
 - > As a platform for course-work design and research
 - > Gentler introduction to commercial server class design
 - > Basic building block for more interesting variants
- Need a hardware prototype board, that is
 - > Re-configurable FPGA based
 - Scalable Available in large quantity throughout the world
 - > Standardized Supports standardized peripherals
 - > Economical





FPGA Implementation: Goals

- Proliferation of OpenSPARC Technology
- Proliferation of Xilinx FPGA Technology
 - > Make OpenSPARC FPGA-Friendly
 - > Create reference design with complete system functionality
 - > Boot Solaris/Linux on the reference design
 - > Open it up ..
 - > Seed ideas in the community

Enable multi-core research





FPGA Implementation: Benefits

- FPGAs provide a flexible design environment
 - > Fast turnaround for changes
 - > Enables experimentation in hardware
 - > Speeds up verification time

- Cost Savings
 - > Don't have to pay fabrication costs for each new chip





Creating an FPGA-friendly Design

- The following changes were made to the OpenSPARC T1 code
- Re-code sections for more efficient FPGA synthesis
 - > Use Block RAMs effectively
 - > Efficiently synthesize logic
- Put in options to reduce size
 - > Four threads --> one thread
 - > Reduce TLB entries from 64 to 8
 - > Remove modular arithmetic unit from design





FPGA Implementation

Initial version released May 2006

(on OpenSparc.net website)

- > full 8-core, 32-thread
- > First-cut implementation; not yet optimized for Area/Timing
- > Synplicity scripts for Xilinx/Altera FPGAs
- Reduced version released Mar 2007 Release 1.4
 - > Single-core, single-thread
 - > Reduced size TLB
 - > Optimizations for Area





New Features of Release 1.6

- Support for Virtex-5 ML505 board
 Unarrodod to XC5/// X110T
 - > Upgraded to XC5VLX110T
- Implementation of 4-thread core on FPGA
- Complete OpenSolaris Image
- Quick-start files, enable you to boot OpenSolaris on day one.





OpenSPARC T1 on FPGAs

- Single thread version
 - ~40K Virtex-2/4 LUTs, 30K Virtex-5 LUTs
 - > Optimized for area
 - No modular arithmetic (MA), reduced TLBs
 - > Easily meets 20ns cycle time (50MHz)
 - > Fits into a Xilinx XC4VFX60
 > Full TLB and MA included: 50K Virtex-4 LUTs



Plot of 1-thread design on XC4VFX60





Single Thread T1 on FPGAs

- Functionally stable
 - > Passing mini and full regressions
- Completely routed
 - > No timing violations
 - > Easily meets 20ns (50MHz) cycle time
- Expandable to more threads
 - > Reasonable overhead for most blocks (~30% for 4 threads)
 - > Some bottlenecks exist (Multi-port register files)





OpenSPARC T1 on FPGAs

- Four thread version
 - Functionality identical to Niagara1 core – on FPGAs
 - > No Modular Arithmetic unit
 - > 16-entry TLB
 - > 69K Virtex-2/4 LUTs, 51K Virtex-5 LUTs
 - > 40%+ reduction in area compared to original design
 - > Runs at 10 MHz
 - > Block RAMs used: v4: 127, v5: 115



Plot of 4-thread design on XC5VLX110T





OpenSPARC T1 FPGA variants



T1 – 8 cores, 32 threads

T1 - 1 core, 4 threads

1 core, 1 thread

www.opensparc.net





System-on-FPGA

- Goal: Create a working system on an FPGA Board
 - > Requires: core, memory interface, peripherals
 - > Core requires L2 cache for coherence, and connectivity to memory controller
 - > This won't fit on the FPGA
- Needed a small replacement for L2
 - > And we had an aggressive schedule
- Solution:

> Use a Xilinx MicroBlaze Core to process memory transactions





System Block Diagram







System Operation

- OpenSPARC T1 core communicates exclusively via cache-crossbar interface (CCX)
 - > PCX (processor-to-cache), CPX (cache-to-processor)
 - > Glue logic block forwards packets between OpenSPARC core and Microblaze
- Microblaze firmware polls T1 core and system peripherals
 - > Services memory and I/O requests
 - > Performs address mapping
 - > Returns results to the core
 - Maintains L1 cache coherence





T1 EDK Project

- System captured in Xilinx EDK project
 - > T1 core and Microblaze glue logic defined as Xilinx peripheral cores ("pcores")
 - > T1 netlist generated via Synplicity or Xilinx XST
 - > Implemented on a Xilinx XC5VLX110T

Xilinx Platform Studio – /import/dlg	-data22/durgam/opensparo/T1/ma	ay31/magara/design/sys/ipga/	sysiem.xmp – [8	System Assembly View 1] 😑 🕀 😣			
📁 File Edit View Project Hardware Software Device Configurati	on Simulation Window Help			* X X			
X	Filters						
Project Applications IP Catalog F O L	L F 📲 Businterface 🕽 Ports 🤇	Addresses 🎦 Filters (Applied)					
Software Projects	M S Name	Bus Connection IP Type	IP Version				
Contract rejecto	Provincializa ()	microblaze	5.00 c				
Default microplaze 0 bootloop	$\square \square $	opb_v20	1.10.0				
- Default microblaze 0 xmdstub	€- ≫ ilmb	lmb_v10	1.00.a				
Project: TestApp_Memory	← ← ◆ dimb	Imb_v10	1.00.a				
E-Processor: microblaze_0	- E- ccx2mb_0_to_microblaz.	fsl_v20	2.10.a				
- Executable: /import/dtg-data22/durgam/op	🗗 🗢 microblaze_0_to_ccx2m.	fsl_v20	2.10.a				
E-Compiler Options	└┘── E-◇ debug_module	opb_mdm	2.00.a				
E-Sources O	🔶 📴 🗢 dimb_cntir	lmb_bram_if_cntlr	12.00.a				
-Headers Contraction Contraction	○ E- > ilmb_cntir	lmb_bram_if_cntlr	2.00.a				
🗄 🛱 Project: TestApp_Peripheral	₽ → Imb_bram	bram_block	1.00.a				
E-Processor: microblaze_0		opb_uartiite	1.00.b				
- Executable: /import/dtg-data22/durgam/op	DDR_SDRAM_32Mx64	mch_opb_ddr	1.00.0				
E-Compiler Options	P ORGate_1	util_reduced_logi	c 1.00.a				
E-Sources	₽-♀sysclk_inv	util_vector_logic	1.00.a				
E⊢Headers	₽- → clk90_inv	util_vector_logic	1.00.a				
Project: ccx2mb_test_sw	₽ ddr_cik90_inv	uti_vector_logic	1.00.a				
EF Processor: microblaze_U	dcm_0	dom_module	1.00.a				
Executable: Importidg-data22/durgamop	er ocm_1	em2mb	1.00.a				
E-Compiler Options	CCK2mb_0	ion fore	1.00.4				
E-Sources	Er 100_10ya_0	lob_lb8ø	1.00.0				
Project: ccx rentar sw							
Processor microblaze 0							
Executable: /import/dtg-data22/durgam/op							
E-Compiler Options							
E-Sources							
E-Headers V							
Block Diagram	System Assembly View1						
INFO:Xst:2260 - The FF/Latch <q z[8]=""> in Unit <rf32x80 ctrl="" dff="" dl="" wr=""> is equivalent to the following 7 FFs/Latches : <q td="" 🗖<=""></q></rf32x80></q>							
	22 Maria Managaran Constanting Santaran - Anto		**************************************				
INFO:Xst:2260 - The FF/Latch <q_z[2]< td=""><td>JJ> in Unit <rf32x80_d< td=""><td>itt_wr_ctrl_dl> is e</td><td>quivalent</td><td>to the following 11 FFs/Latches : <</td></rf32x80_d<></td></q_z[2]<>	JJ> in Unit <rf32x80_d< td=""><td>itt_wr_ctrl_dl> is e</td><td>quivalent</td><td>to the following 11 FFs/Latches : <</td></rf32x80_d<>	itt_wr_ctrl_dl> is e	quivalent	to the following 11 FFs/Latches : <			
INFO:Xst:2260 - The FF/Latch <q_2[8< td=""><td>> in Unit <rt32x8u_df< td=""><td>f_din_dl> is equiva</td><td>lent to th</td><td>e tollowing 2 FFs/Latches : <q_2[]<="" td=""></q_2[></td></rt32x8u_df<></td></q_2[8<>	> in Unit <rt32x8u_df< td=""><td>f_din_dl> is equiva</td><td>lent to th</td><td>e tollowing 2 FFs/Latches : <q_2[]<="" td=""></q_2[></td></rt32x8u_df<>	f_din_dl> is equiva	lent to th	e tollowing 2 FFs/Latches : <q_2[]<="" td=""></q_2[>			
INFU:Xst:2260 - The FF/Latch <q_2[3.< td=""><td>LJ> in Unit <ri32x80 d<br="">LS in Unit <rf32x80 df<="" td=""><td>irr ain al> is equiv</td><td>alent to t uivalant t</td><td>ne following 2 FFS/Latches : <q_2[3< td=""></q_2[3<></td></rf32x80></ri32x80></td></q_2[3.<>	LJ> in Unit <ri32x80 d<br="">LS in Unit <rf32x80 df<="" td=""><td>irr ain al> is equiv</td><td>alent to t uivalant t</td><td>ne following 2 FFS/Latches : <q_2[3< td=""></q_2[3<></td></rf32x80></ri32x80>	irr ain al> is equiv	alent to t uivalant t	ne following 2 FFS/Latches : <q_2[3< td=""></q_2[3<>			
INFO.Ast.2200 - The FF/Latch <q 7[8]<="" td=""><td>> in Unit <rf32v80_df< td=""><td>fwr_ctrl_dl> is eq</td><td>uivalent t</td><td>o the following 11 FFs/Latches . <q_< td=""></q_<></td></rf32v80_df<></td></q>	> in Unit <rf32v80_df< td=""><td>fwr_ctrl_dl> is eq</td><td>uivalent t</td><td>o the following 11 FFs/Latches . <q_< td=""></q_<></td></rf32v80_df<>	fwr_ctrl_dl> is eq	uivalent t	o the following 11 FFs/Latches . <q_< td=""></q_<>			
INFO:Xst:2260 - The FF/Latch < 71^{-1}	$ 1\rangle$ in Unit <rf32x80 d<="" td=""><td>If din d1> is equiv</td><td>alent to t</td><td>he following 2 FFs/Latches : <g 1<="" 7="" td=""></g></td></rf32x80>	If din d1> is equiv	alent to t	he following 2 FFs/Latches : <g 1<="" 7="" td=""></g>			
	.,	10 odd1,					
INFO:Xst:2260 - The FF/Latch <wrdate< td=""><td>a_d1_Z[112]> in Unit <</td><td>mra> is equivalent</td><td>to the fol</td><td>lowing FF/Latch : <wrdata_d1_z[10]></wrdata_d1_z[10]></td></wrdate<>	a_d1_Z[112]> in Unit <	mra> is equivalent	to the fol	lowing FF/Latch : <wrdata_d1_z[10]></wrdata_d1_z[10]>			
INFO:Xst:2260 - The FF/Latch <wrdate< td=""><td>a d1 7[113]> in [[nit <</td><td>mra> is equivalent</td><td>to the fol</td><td>lowing FF/Latch : <wrdata 7[11]="" d1=""></wrdata></td></wrdate<>	a d1 7[113]> in [[nit <	mra> is equivalent	to the fol	lowing FF/Latch : <wrdata 7[11]="" d1=""></wrdata>			
INFO:Xst:2260 - The FF/Latch <wrdata< td=""><td>a d1 Z[75]> in Unit <m< td=""><td>ra> is equivalent t</td><td>o the foll</td><td>owing 7 FFs/Latches : <wrdata d1="" td="" z1<=""></wrdata></td></m<></td></wrdata<>	a d1 Z[75]> in Unit <m< td=""><td>ra> is equivalent t</td><td>o the foll</td><td>owing 7 FFs/Latches : <wrdata d1="" td="" z1<=""></wrdata></td></m<>	ra> is equivalent t	o the foll	owing 7 FFs/Latches : <wrdata d1="" td="" z1<=""></wrdata>			
INFO:Xst:2260 - The FF/Latch <wrdate< td=""><td>_d1 Z[123]> in Unit <</td><td>mra> is equivalent</td><td>to the fol</td><td>lowing 7 FFs/Latches : <wrdata 2<="" dl="" td=""></wrdata></td></wrdate<>	_d1 Z[123]> in Unit <	mra> is equivalent	to the fol	lowing 7 FFs/Latches : <wrdata 2<="" dl="" td=""></wrdata>			
INFO:Xst:2260 - The FF/Latch <rdptr< td=""><td>_d1_1_z[0]> in Unit <m< td=""><td>µra≻ is equivalent t</td><td>o the foll</td><td>owing FF/Latch : <rdptr_d1_z[0]> -</rdptr_d1_z[0]></td></m<></td></rdptr<>	_d1_1_z[0]> in Unit <m< td=""><td>µra≻ is equivalent t</td><td>o the foll</td><td>owing FF/Latch : <rdptr_d1_z[0]> -</rdptr_d1_z[0]></td></m<>	µra≻ is equivalent t	o the foll	owing FF/Latch : <rdptr_d1_z[0]> -</rdptr_d1_z[0]>			
INFO:Xst:2260 - The FF/Latch <byte_< td=""><td>ven_dl_Z[7]> in Unit <</td><td>mra> is equivalent</td><td>to the fol</td><td>lowing 5 FFs/Latches : <byte_wen_d1< td=""></byte_wen_d1<></td></byte_<>	ven_dl_Z[7]> in Unit <	mra> is equivalent	to the fol	lowing 5 FFs/Latches : <byte_wen_d1< td=""></byte_wen_d1<>			
INFU:Xst:2260 - The FF/Latch <byte_< td=""><td>ven_d1_Z[13]> in Unit</td><td><mra> is equivalent</mra></td><td>to the fo</td><td>llowing 5 FFs/Latches : <byte_wen_d< td=""></byte_wen_d<></td></byte_<>	ven_d1_Z[13]> in Unit	<mra> is equivalent</mra>	to the fo	llowing 5 FFs/Latches : <byte_wen_d< td=""></byte_wen_d<>			
INFU:AST:2260 - The FF/Latch <byte 1<="" td=""><td>ven af 2[19]> in Unit d1 1 7[3]> in Unit /~</td><td><pre>\mra> is equivalent </pre></td><td>co the foll</td><td>awing PF/Latch : <rdptp 7(2)<="" d1="" td=""></rdptp></td></byte>	ven af 2[19]> in Unit d1 1 7[3]> in Unit /~	<pre>\mra> is equivalent </pre>	co the foll	awing PF/Latch : <rdptp 7(2)<="" d1="" td=""></rdptp>			
Inro.Ast.2200 - The rr/Latch <raptr< td=""><td>ייד דיצו⊃ן> דוו טוודג ≺ת</td><td>uaz is equivalent t</td><td>o une 1011</td><td>owing reynation . Nrupte of 2[5]> M</td></raptr<>	ייד דיצו⊃ן> דוו טוודג ≺ת	uaz is equivalent t	o une 1011	owing reynation . Nrupte of 2[5]> M			
Output Warning Error							
Paoty							
neauy							





T1 EDK Project (Cont'd)

- Entire system placed & routed
- Downloaded to FPGA on ML505 board
- Use Debugger to load software into memory
- Run!
- View program output via serial cable connected to a PC



ML505 Board (not upgraded)





Included in EDK Project

- SystemAce file for quick start-up
- EDK system setup files
- Synplicity-generated netlist:
 - > 4 threads, 16 TLB entries, no SPU.
- Firmware to process cache crossbar packets
 - > Setup to run stand-alone tests on the board
 - > Setup to boot Hypervisor
- Full-system simulation setup using Modelsim





FPGA Board







Latest OpenSPARC Development Kit

- • A kit for OpenSPARC development now available
- • Manufactured by Digilent Inc.
- Commercial interests can directly order from http://www.digilentinc.com
- Board based on the ML505, but with an XC5VLX110T FPGA
- • Kit Includes:
- Board, with power supply and 256 MB DRAM
- > Platform USB download cable
- Host to host SATA crossover cable
- Compact flash card with OpenSPARC T1 1.6 ace files





OpenSPARC Development Kit Contents







Quick Start-Up

- Files:
 - > design/sys/edk/ace/
 - OpenSPARCT1_1_6_os_boot.ace
 - OpenSolaris Boot on a 4-thread core (on ML505-110T)
 - OpenSPARCT1_1_6_Hello_World.ace
 - Run a standalone program under hypervisor
- Procedure:
 - > Format a compact flash card with Xilinx filesystem
 - > Copy a file to the compact flash card
 - Insert CF card into board socket (set DIP switches)
 - > Connect Serial port on board to a computer
 - > Using a null-modem serial cable
 - > Use Hyperterminal or some other terminal to connect





Quick Start-Up (Cont'd)

Boot Process

- > Turn on the board
- > At OBP "OK" prompt, type "boot"
 - > boot -m milestone=none (Fast single-user boot)
 - > boot -mverbose (Enable networking)
- > At login prompt (30-60 minutes later) login as root
- > Interesting commands
 - > psrinfo

Will show 4 processors

> uname





Running Stand-alone Tests

- We use the ELF executable and the memory image created by the simulation
- Memory Map table created
 - > Maps different program segments into 256 MB DRAM
 - > Compiled into firmware executable.
- Download and run the firmware
 - > Firmware will send wake-up to core
 - > Will process packets
- Will report success or failure (GOOD_TRAP/BAD_TRAP)





How to Run Stand-alone tests

- Run the simulation of the test using sims
- Generate the memory table for the test
 - genmemimage.pl -single -f memory-image-file -name test_name
- Copy the memory table to the EDK project
 % cp mbfw_diag_memimage.c ccx-firmware-diag/src
- Re-build the firmware
- Download
- Run





Running Hardware Regressions

- Run the sims regression
- Generate the memory tables for each test
 - genmemimage.pl -d regression-dir
 - Creates a directory named diags
- Edit the diag list
 - design/sys/edk/scripts/
 - thread1_mini.list, thread1_full.list, core1_mini.list, or core1_full.list

Run the regression script

% xtclsh edk-project-dir/scripts/rundiags.tcl -edk edk-project-dir
 -list edk-project-dir/scripts/diag_mini.list -d diag_dir -model core1 -suite {thread1_mini thread1_full core1_mini core1_full}





Memory Allocation (256 MB DDR2)

 256 MB DDR2 DRAM is at MicroBlaze Address 0x50000000

MicroBlaze Address

DRAM Utilization

0x5000_0000	MicroBlaze Firmware			
0x5010_0000	OpenSPARC Memory Space: 174 MB			
	0x00_0000_0000 – 0x00_0fdf_ffff			
0x5aef_ffff	Ram Disk Image (80 MB)			
0x5ff0_0000	OpenSPARC Boot Prom: 0xff_f000_000			

Function





Booting Solaris on an FPGA Board

- MicroBlaze firmware is compiled and loaded into DRAM
 - > A fixed memory translation table is used to map OpenSPARC addresses to MicroBlaze addresses
- Boot PROM image and RAM disk images loaded as data into DRAM
- The firmware program is started





Software Stack

- Use Standard software installation
- Use a virtual disk in RAM to hold the Solaris binaries
- Some memory copy sections performed by MicroBlaze
- MicroBlaze firmware now performs floating-point operations, so emulation is not needed







Boot Sequence

- The processor starts at the Power-On Reset (POR) trap handler
- Reset code is executed: Caches & TLBs enabled
- Control passed to Hypervisor
 > Hypervisor copies itself from PROM to RAM area
 > Passes control to Open Boot PROM (OBP)
- OBP then loads the operating system





Steps to Boot the Operating System

- Download the bit file to the FPGA
- Start the debugger
 - > Download the MicroBlaze firmware

% dow mb-firmware-hv/executable.elf

- > Download the PROM image
 - % dow -data prom.bin 0x5ff00000
- > Download the RAM disk image
 - % dow -data ramdisk_image.bin 0x5af00000
- Start the firmware
 - % run
- > At OBP prompt, type the boot command
 - Ok boot -m milestone=none





Solaris Boot

🍣 new - HyperTerminal							
File Edit View Call Transfer Help							
D 🚅 🍵 🔏 🗈 🖰	r in the second						
				[🔨			
T1_INF0: cyclic_add: handler 0x125bc68 cyt_interval 0x7735940 Booting to milestone "none". Requesting System Maintenance Mode (See /lib/svc/share/README for more information.) Console login service(s) cannot run							
Root password for system maintenance (control-d to bypass): single-user privilege assigned to /dev/console. Entering System Maintenance Mode							
Jan 4 17:47:31 su: 'su root' succeeded for root on /dev/console Sun Microsystems Inc. SunOS 5.10 Generic January 2005							
bin	home	platform	system				
cbclinks	import	proc	tĺb				
dev	java	read_time	tmp				
devices	kernel	run	usr				
dhry		run_commands.sh	var				
doe	lost+found	SDIN	workspace				
dungoon	MICFO mpt	scue	WS				
late	net	shared	wwss				
exnort	ont	src					
#_		5.0					
				v			
Connected 4:41:57 Au	to detect 9600 8-N-1 SCR	OLL CAPS NUM Capture	Print echo	.:			

www.opensparc.net





Thanks for watching the OpenSPARC Slide-Cast!

Let us hear from you! The OpenSPARC Team would appreciate your feedback on this in the http://www.OpenSPARC.net forum.

This material is made available under Creative Commons Attribution-Share 3.0 United States License

