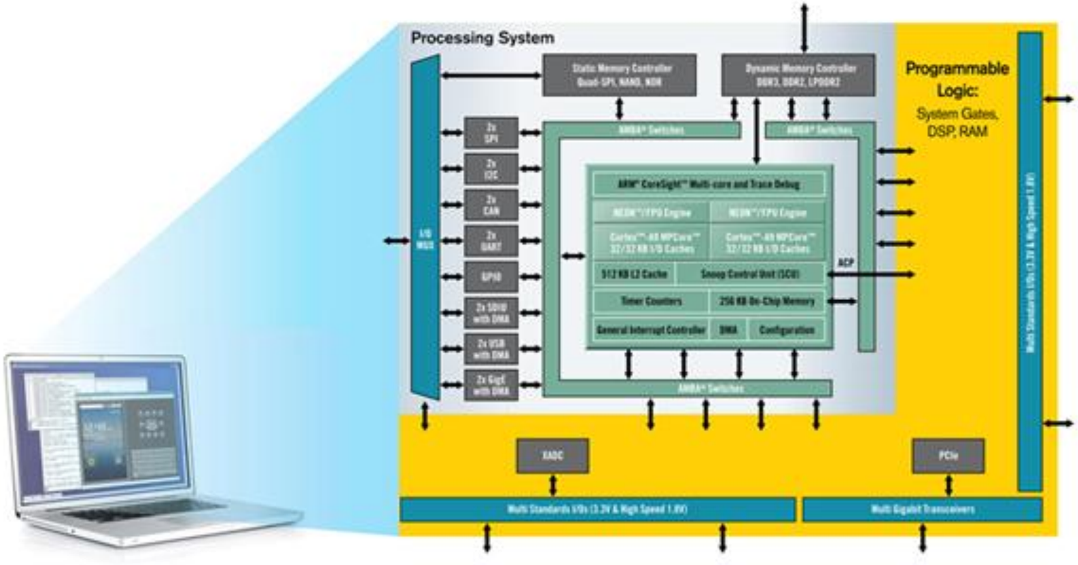
# Die virtuelle Plattform:

## Der Einsatz von Zynq fuer die Verifikation und das Debugging von konfigurierbaren Systemen

Dr. Endric Schubert
Missing Link Electronics
Marlene-Dietrich-Straße 5
89231 Neu-Ulm

www.missinglinkelectronics.com
Tel: +49 (731) 141-149-0


Processing System
Static Memory Controller Quad-SPI, NAND, NOR
Dynamic Memory Controller DDR3, DDR2, LPDDR2
Programmable Logic:
System Gates, DSP, RAM
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
I/O MUX
AMBA® Switches
AMBA® Switches
ARM® CoreSight™ Multi-core and Trace Debug
NEON™/FPU Engine
NEON™/FPU Engine
Cortex™-A9 MPCore™ 32/32 KB I/D Caches
Cortex™-A9 MPCore™ 32/32 KB I/D Caches
512 KB L2 Cache
Snoop Control Unit (SCU)
ACP
Timer Counters
256 KB On-Chip Memory
General Interrupt Controller
DMA
Configuration
AMBA® Switches
XADC
PCIe
Multi Standards I/Os (3.3V & High Speed 1.8V)
Multi Standards I/Os (3.3V & High Speed 1.8V)
Multi Gigabit Transceivers

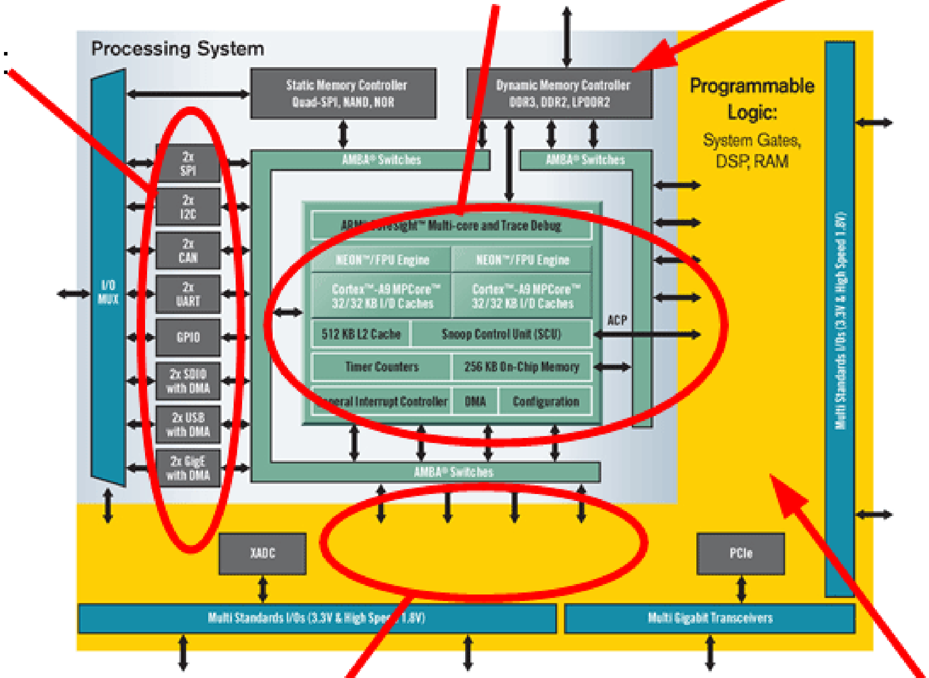
Courtesy Xilinx

# Challenges of Debugging Your Own ASSP

2x ARM Cortex-A9

**Processing System (PS)**
**== Microcontroller**

Hard IP peripherals:
- SPI
- UART
- GPIO
- USB
- GigE
- …

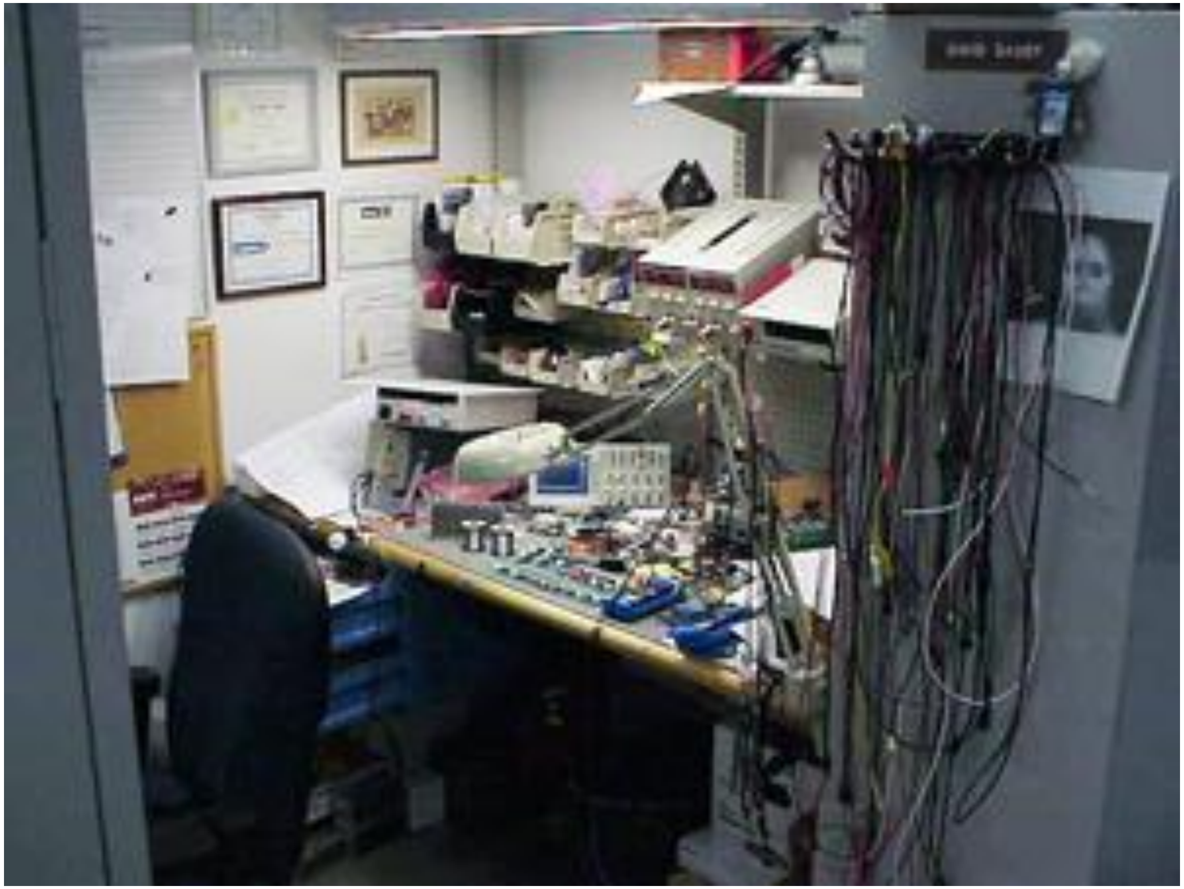


Interfaces between PS and PL:
- 2 AXI Master/Slave Ports
- 4 AXI HP Fifo Ports
- ACP Slave Port
- 16 configurable interrupts

**Programmable Logic (PL)**
**== FPGA**

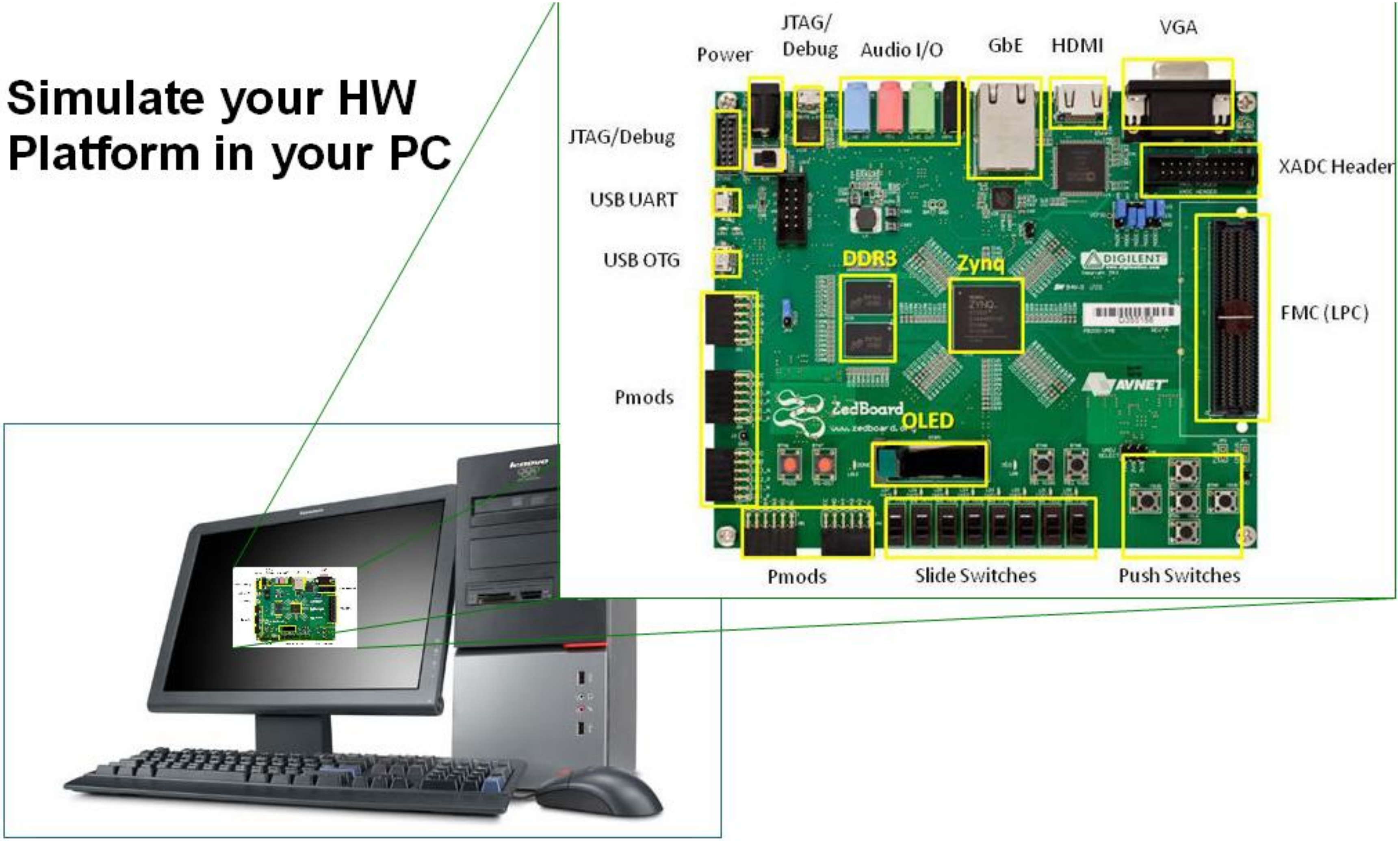
# Another Challenge When Building Your Own ASSP: Making Hardware and Software Work Together.

# ASSP System-on-Chip Design – An Embedded Designers Life

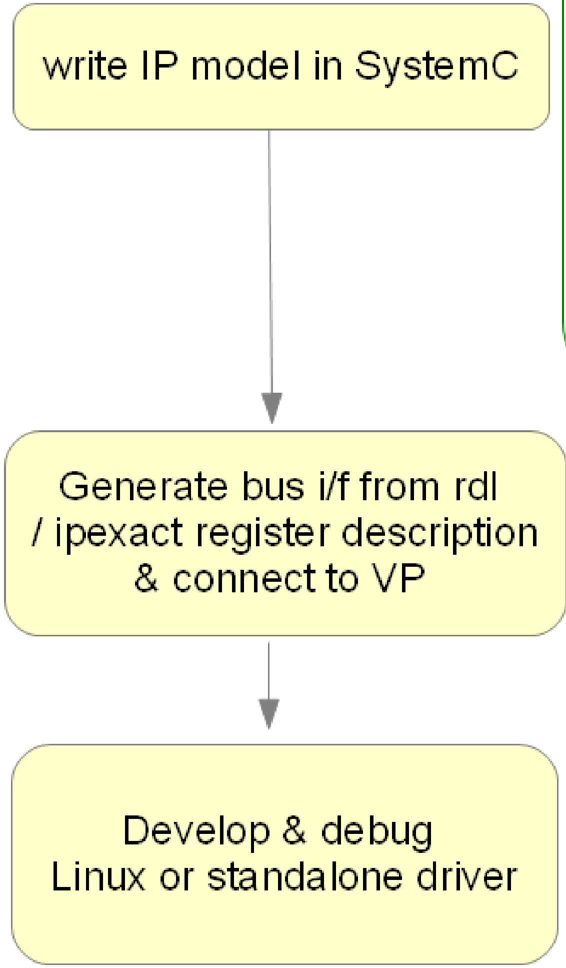


# What is a Virtual Platform?

**Simulate your HW Platform in your PC**

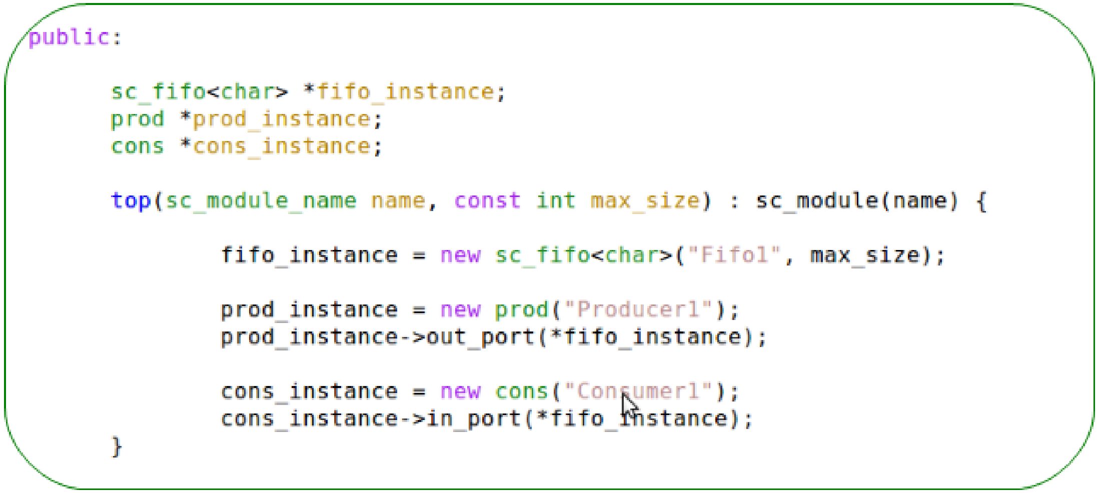


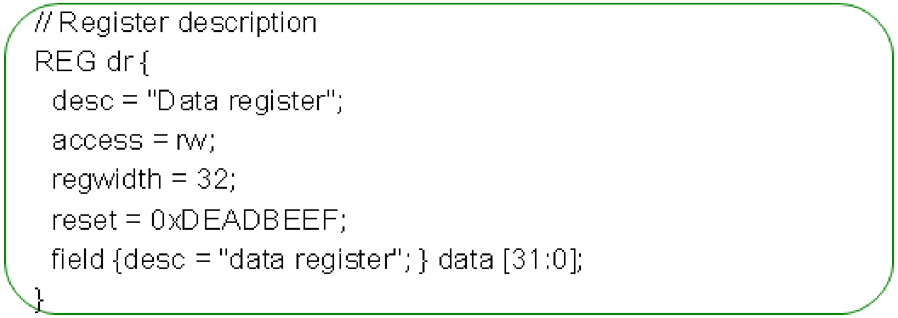
# Virtual Platform Methodology


write IP model in SystemC

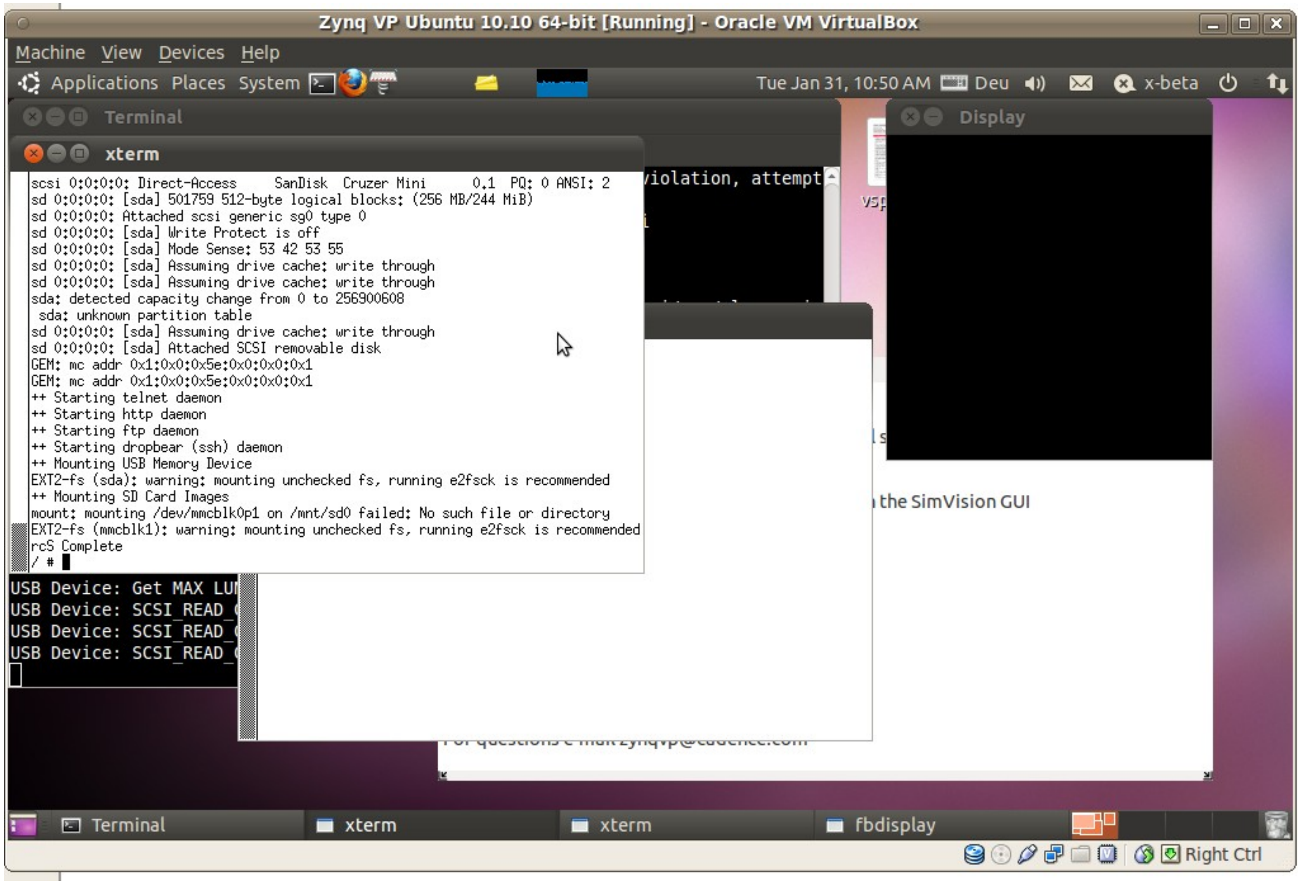
Generate bus i/f from rdl / ipexact register description & connect to VP

Develop & debug Linux or standalone driver


```
public:

    sc_fifo<char> *fifo_instance;
    prod *prod_instance;
    cons *cons_instance;

    top(sc_module_name name, const int max_size) : sc_module(name) {

        fifo_instance = new sc_fifo<char>("Fifo1", max_size);

        prod_instance = new prod("Producer1");
        prod_instance->out_port(*fifo_instance);

        cons_instance = new cons("Consumer1");
        cons_instance->in_port(*fifo_instance);
    }
```


```
// Register description
REG dr {
  desc = "Data register";
  access = rw;
  regwidth = 32;
  reset = 0xDEADBEEF;
  field {desc = "data register"; } data [31:0];
}
```

# Virtual Platforms Can Run Software Fast (Sometimes Faster Than Real)


Zynq VP Ubuntu 10.10 64-bit [Running] - Oracle VM VirtualBox
Machine View Devices Help
Applications Places System
Tue Jan 31, 10:50 AM Deu x-beta
Terminal
xterm
Display
violation, attempt
the SimVision GUI
Terminal
xterm
xterm
fbdisplay
Right Ctrl

```
scsi 0:0:0:0: Direct-Access     SanDisk  Cruzer Mini      0.1  PQ: 0 ANSI: 2
sd 0:0:0:0: [sda] 501759 512-byte logical blocks: (256 MB/244 MiB)
sd 0:0:0:0: Attached scsi generic sg0 type 0
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Mode Sense: 53 42 53 55
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Assuming drive cache: write through
sda: detected capacity change from 0 to 256900608
 sda: unknown partition table
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Attached SCSI removable disk
GEM: mc addr 0x1:0x0:0x5e:0x0:0x0:0x1
GEM: mc addr 0x1:0x0:0x5e:0x0:0x0:0x1
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting dropbear (ssh) daemon
++ Mounting USB Memory Device
EXT2-fs (sda): warning: mounting unchecked fs, running e2fsck is recommended
++ Mounting SD Card Images
mount: mounting /dev/mmcblk0p1 on /mnt/sd0 failed: No such file or directory
EXT2-fs (mmcblk1): warning: mounting unchecked fs, running e2fsck is recommended
rcS Complete
/ #
```

```
USB Device: Get MAX LU
USB Device: SCSI_READ_
USB Device: SCSI_READ_
USB Device: SCSI_READ_
```

# What is SystemC?

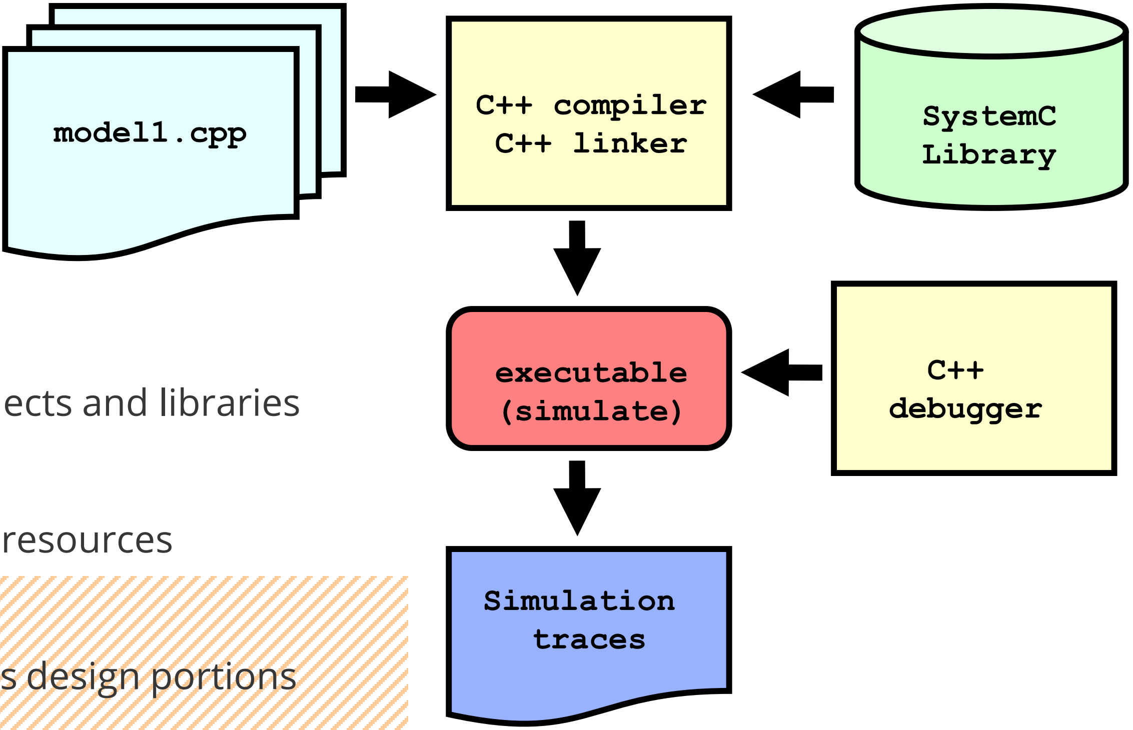
- Extension to ISO C++
  - Means to express concurrency
  - Communication mechanisms
  - Reactivity
  - Concept of Time
  - Event driven simulation kernel
  - A modeling methodology

- Open Source Library managed by Open SystemC Initiative (OSCI)
  - www.systemc.org
  - Current members: ARM Ltd. Cadence Design Systems, Inc. CoWare, Inc. Forte Design Systems Intel Corporation Mentor Graphics Corporation NXP Semiconductors STMicroelectronics  Synopsys, Inc. Actis Design, LLC Atrenta, Inc. Bluespec, Inc. Broadcom Corporation Calypto Design Systems, Inc. Canon Inc. Carbon Design Systems  Celoxica Ltd. ChipVision Design Systems AG Denali Software Inc. Doulos Ltd. ESLX, Inc. Fraunhofer Institute for Integrated Circuits Freescale Semiconductor Inc. GreenSocs Ltd.  Industrial Technology Research Institute (ITRI)  JEDA Technologies Inc.  Infineon Technologies AG  NEC Corporation Semiconductor Technology Academic  Research Center (STARC) SpringSoft, Inc.  Synfora Inc.  Tenison EDA VaST Systems   Technology Corporation

# History and road-map of SystemC

- V1.0 - Hardware design flow
  - RTL and behavioral modeling of hardware blocks
- V1.1 - Timed functional modeling, e.g. for busses
- V2.0 - System design flow
- V2.1 - Transaction-Level Modeling (TLM)
  - OSCI TLM Standard V1.0
- Standardized LRM as IEEE 1666 12/2005
- V2.2 - Released 3/2007
  - OSCI TLM Standard V2.0 in works
- Future: more RTOS and software support

# SystemC Design Flow

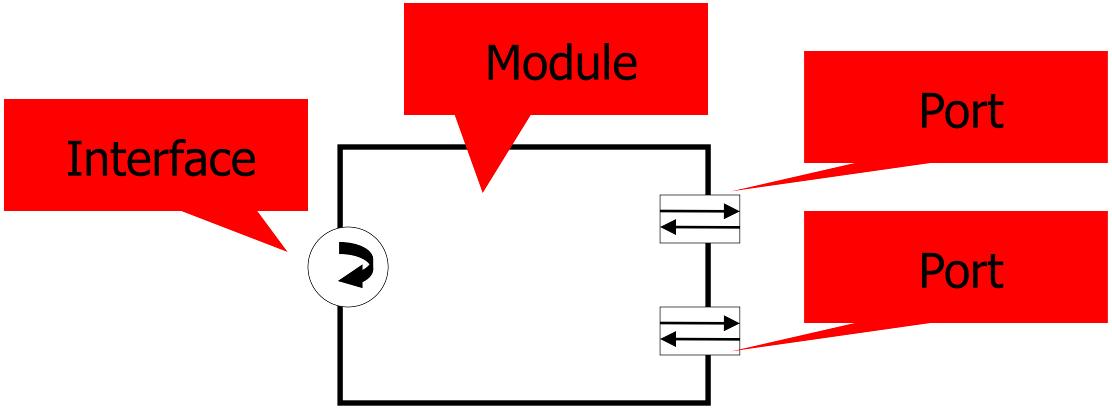


- 1. Compilation
  - C++ compiler transforms C++ text into object code
- 2. Linking
  - C++ linker builds executable out of objects and libraries
- 3. Execution
  - Executable is started allocates system resources
- 4. Elaboration
  - SystemC kernel connects and initializes design portions
- 5. Simulation
  - SystemC kernel works on event queue until no more events

# SystemC Modules and Hierarchy

- Modules as containers for structure
  - Module = C++ class

```
class Adder : public sc_module {
   // port declarations
   sc_in<int>  in1;
   sc_in<int>  in2;
   sc_out<int> out1;

   // signals, processes, etc
   …
   // constructor
   …
 };
```



  - Instance = C++ object

```
Adder module_add1("a1");
```

# SystemC Concurrency via Processes

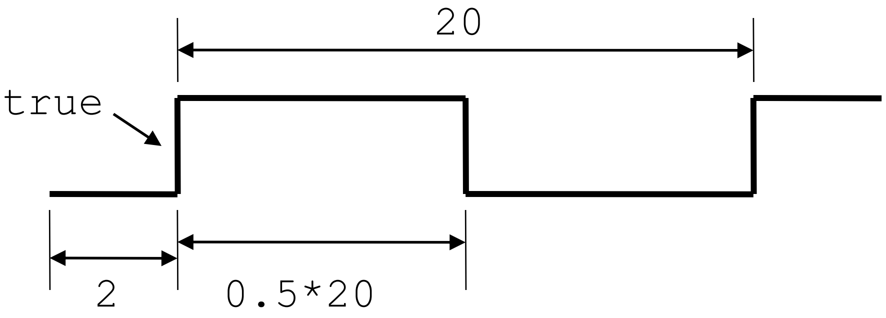
- Process is basic unit of functionality
  - The means for concurrent execution
  - Contained in a module as a class member function
    - `void compute(void) {…}`
  - Needs to be registered with simulation kernel
  - Synchronization via events
- Two kinds of processes `SC_METHOD`, `SC_THREAD`
  - Example

```
– SC_MODULE(ddm) {
…   int inc = 0; // not static
  void compute();
  SC_CTOR(ddm) {
    SC_THREAD(compute);
    sensitive << inp;
  }
};
                         ddm.hpp
```

```
void ddm::compute() {

  while(true) {
    outp.write(inp.read()+inc);
    inc++;
    wait();
  }
}
                                 ddm.cpp
```

# SystemC Model of Time

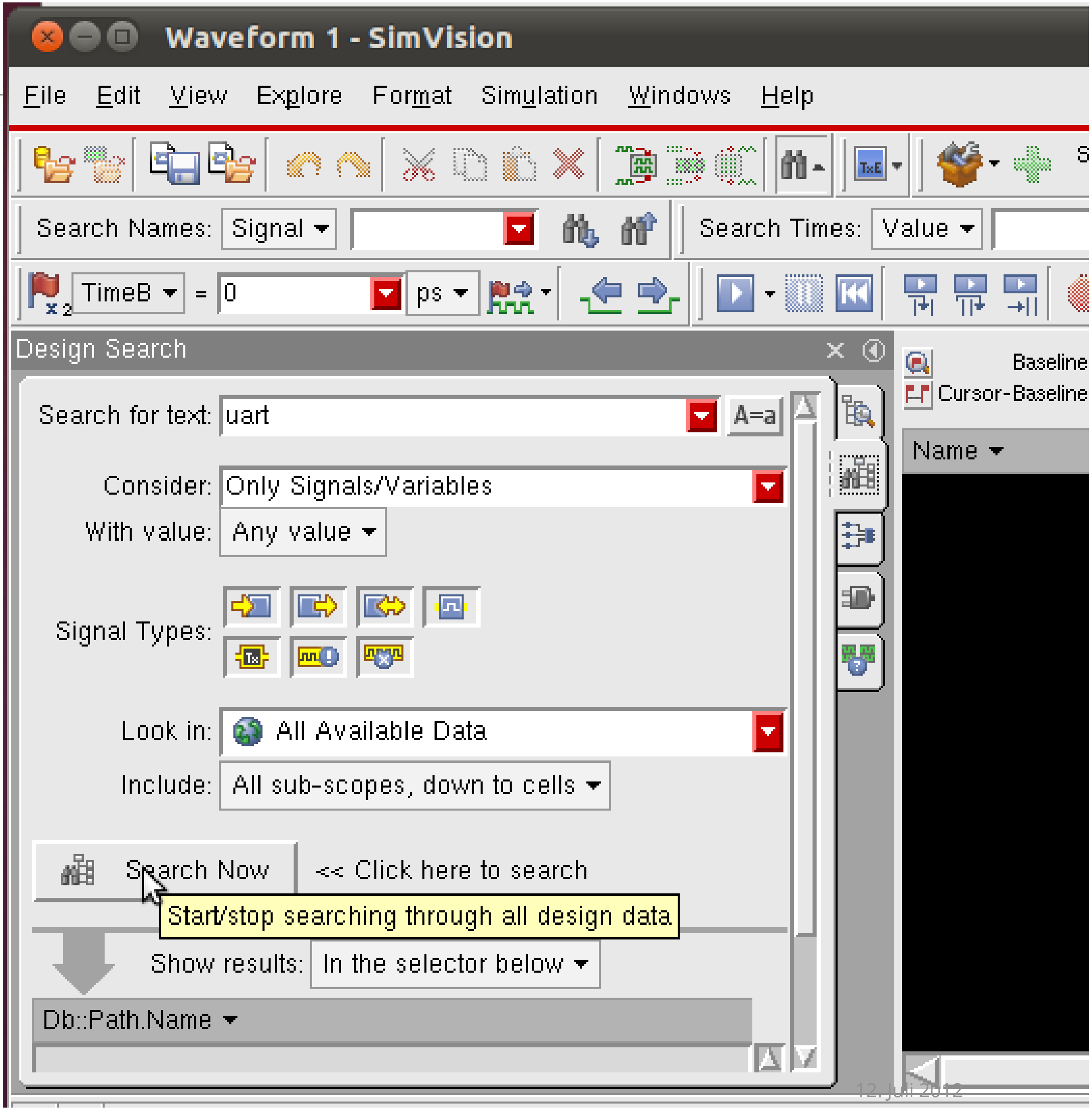
- Integer-valued time model inside simulation kernel
  - Unsigned 64-bit, more bits if needed
  - Minimum time resolution, rounded up
- Time units, enumerated type `sc_time_unit`
  - `class sc_time {…};`
  - `SC_FS` - femtoseconds
  - `SC_PS` - picoseconds (default)
  - `SC_NS` - nanoseconds
  - `SC_US` - microseconds
  - `SC_MS` - milliseconds
  - `SC_SEC` - seconds
- Clock module
  - `sc_clock clk1("clk1", 20, 0.5, 2, true);`



# Example Design in VHDL / SystemC

```
•  ENTITY counter IS
•    PORT(count_val: OUT integer;
•    clk: INOUT BOOLEAN);
•  END ENTITY counter;

•  ARCHITECTURE proc OF counter IS
•    SIGNAL cnt: integer;
•  BEGIN
•    p: PROCESS
•    BEGIN
•      WAIT ON clk´event and clk=´1´;
•      cnt <= cnt+1;
•    END PROCESS p;

•    count_val <= cnt;
•  END ARCHITECTURE proc;
```

```
•  class counter: sc_module {
•    sc_out<int> count_val;
•    sc_in<bool> clk;
•    int cnt;

•    do_count() {
•      cnt = cnt + 1;
•      count_val.write(cnt);
•    };

•    SC_HAS_METHOD(counter);
•    counter() {
•      SC_METHOD(do_count)
•      sensitive_pos << clk;
•    }
•  };
```



Search for text: uart

Consider: Only Signals/Variables

With value: Any value

Look in: All Available Data

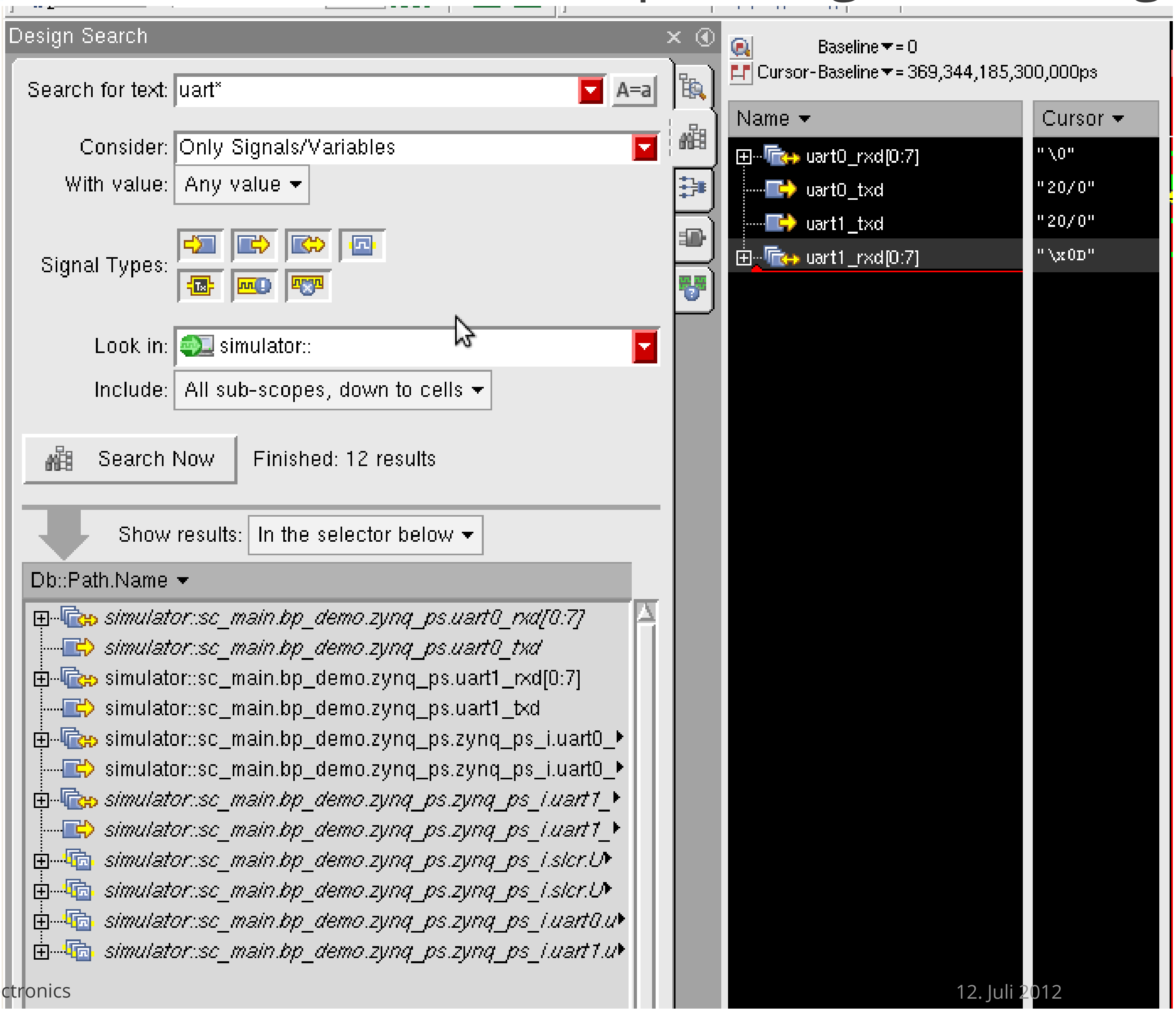
Include: All sub-scopes, down to cells

<< Click here to search

Start/stop searching through all design data

Show results: In the selector below

# Virtual Platform Example – Signal Tracing



Design Search

Search for text: uart*

Consider: Only Signals/Variables

With value: Any value

Signal Types:

Look in: simulator::

Include: All sub-scopes, down to cells

Finished: 12 results

Show results: In the selector below

Db::Path.Name

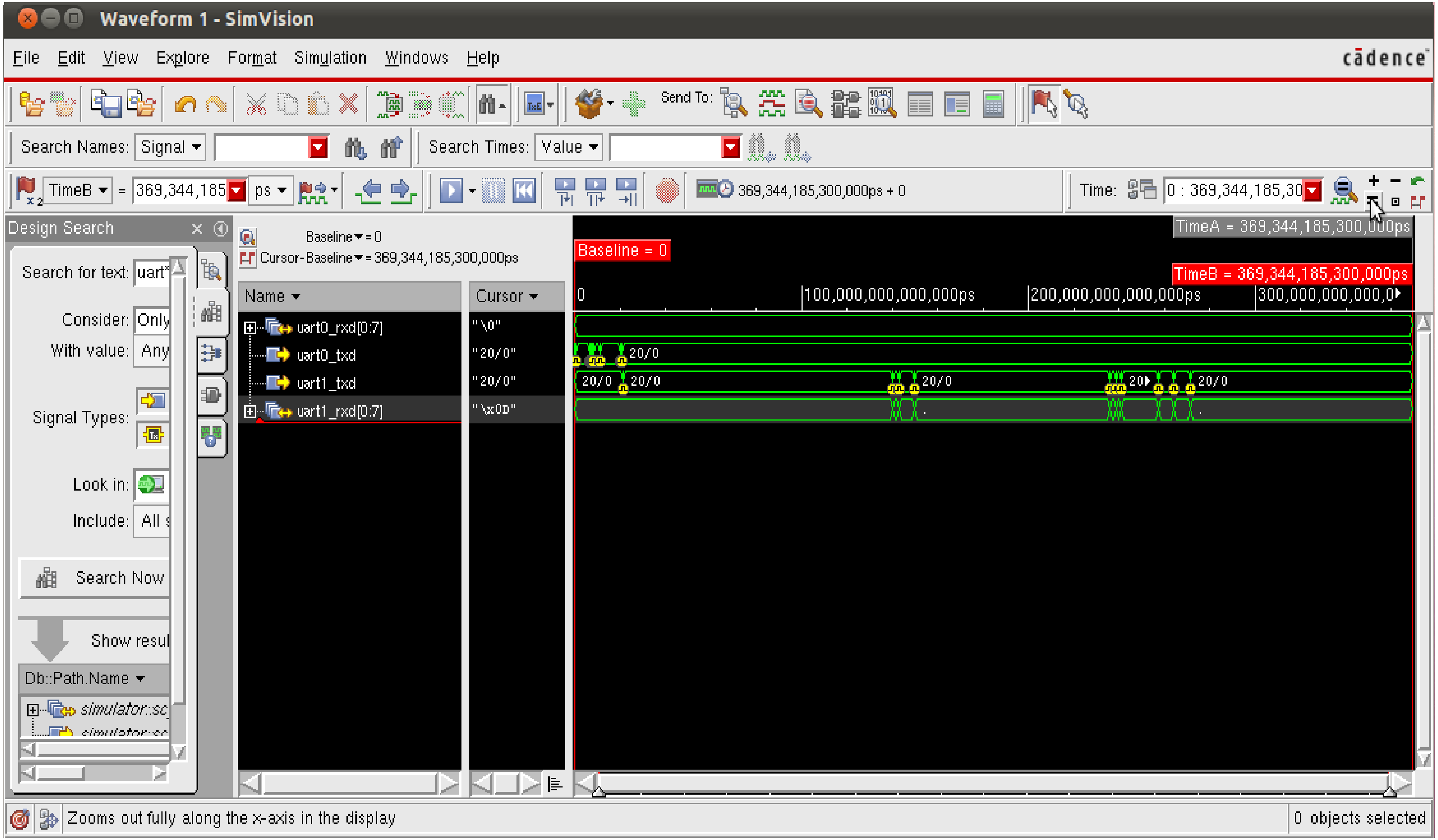
- simulator::sc_main.bp_demo.zynq_ps.uart0_rxd[0:7]
- simulator::sc_main.bp_demo.zynq_ps.uart0_txd
- simulator::sc_main.bp_demo.zynq_ps.uart1_rxd[0:7]
- simulator::sc_main.bp_demo.zynq_ps.uart1_txd
- simulator::sc_main.bp_demo.zynq_ps.zynq_ps_i.uart0_
- simulator::sc_main.bp_demo.zynq_ps.zynq_ps_i.uart0_
- simulator::sc_main.bp_demo.zynq_ps.zynq_ps_i.uart1_
- simulator::sc_main.bp_demo.zynq_ps.zynq_ps_i.uart1_
- simulator::sc_main.bp_demo.zynq_ps.zynq_ps_i.slcr.U
- simulator::sc_main.bp_demo.zynq_ps.zynq_ps_i.slcr.U
- simulator::sc_main.bp_demo.zynq_ps.zynq_ps_i.uart0.u
- simulator::sc_main.bp_demo.zynq_ps.zynq_ps_i.uart1.u

Baseline = 0

Cursor-Baseline = 369,344,185,300,000ps

| Name | Cursor |
|---|---|
| uart0_rxd[0:7] | "\0" |
| uart0_txd | "20/0" |
| uart1_txd | "20/0" |
| uart1_rxd[0:7] | "\x0D" |

# Virtual Platform Example – Signal Tracing



Search Names: Signal

Search Times: Value

TimeB = 369,344,185 ps

369,344,185,300,000ps + 0

Time: 0 : 369,344,185,30

Search for text: uart*

Consider: Only

With value: Any

Signal Types:

Look in:

Include: All s

Baseline = 0

Cursor-Baseline = 369,344,185,300,000ps

TimeA = 369,344,185,300,000ps

TimeB = 369,344,185,300,000ps

Baseline = 0

| Name | Cursor |
|---|---|
| uart0_rxd[0:7] | "\0" |
| uart0_txd | "20/0" |
| uart1_txd | "20/0" |
| uart1_rxd[0:7] | "\x0D" |

Zooms out fully along the x-axis in the display

0 objects selected

# Virtual Platform Options for Zynq

- QEMU – "free" Open Source – great for SW development

  runs Linux, download @ xilinx.com
- VP System Designer – for SW development
  - License available from Cadence via Xilinx
- VP SystemCreator – for system realization (HW & SW)
  - License  available from Cadence via Xilinx

# References

- "System Design with SystemC", T. Groettker, Springer
- "CADENCE Virtual System Platform for the Xilinx Zynq-7000 EPP", www.Cadence.com
- "Designing Software Applications with an Extensible Virtual Platform", www.Xilinx.com
- "Managing Latency and Bandwidth in HW/SW Co-Processing", Endric Schubert, Embedded World Conference, 2012

# Die virtuelle Plattform:

## Der Einsatz von Zynq fuer die Verifikation und das Debugging von konfigurierbaren Systemen

Dr. Endric Schubert
Missing Link Electronics
Marlene-Dietrich-Straße 5
89231 Neu-Ulm

www.missinglinkelectronics.com
Tel: +49 (731) 141-149-0