EECS 222A: System-on-Chip Description and Modeling Lecture 8

Rainer Dömer

doemer@uci.edu

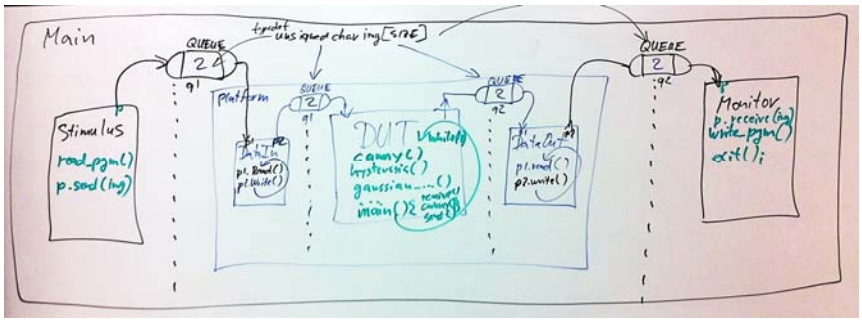
The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 8: Overview

- Project Discussion: Canny Edge Detector
  - Assignment 5
- Modeling with SystemC SLDL
  - SystemC Overview
  - Introduction to SystemC
  - Presentation by Stuart Swan, Cadence
- Producer / Consumer Example in SystemC
  - Assignment 6

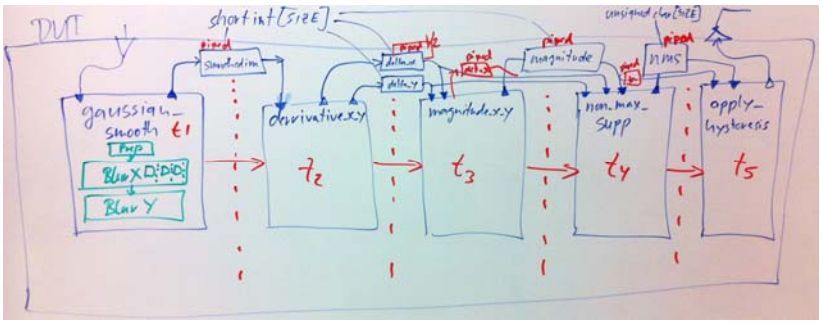
Project Discussion

- Structural Hierarchy for Canny Edge Detector
  - Test bench Structure



Project Discussion

- Additional Level of Hierarchy inside DUT



  - Potential for parallelism
    - 5 pipeline stages in DUT (red color)
    - Parallel decomposition of BlurX and BlurY blocks in Gaussian Smooth behavior (green color)

Project Discussion

- Parallel Hierarchy of DUT for Canny Edge Detector
  - `sir_tree –blt canny_a4_ref.sir DUT`
  - `B i s  behavior DUT`
  - `B i l  |------ Apply_Hysteresis apply_hysteresis`
  - `B i l  |------ Derivative_X_Y derivative_x_y`
  - `B i s  |------ Gaussian_Smooth gaussian_smooth`
  - `B i c  |       |------ BlurX_par blurX_par`
  - `B i l  |       |       |------ BlurX blurX1`
  - `B i l  |       |       |------ BlurX blurX2`
  - `B i l  |       |       |------ BlurX blurX3`
  - `B i l  |       |       \------ BlurX blurX4`
  - `B i c  |       |------ BlurY_par blurY_par`
  - `B i l  |       |       |------ BlurY blurY1`
  - `B i l  |       |       |------ BlurY blurY2`
  - `B i l  |       |       |------ BlurY blurY3`
  - `B i l  |       |       \------ BlurY blurY4`
  - `B i l  |       \------ Prep prep`
  - `B i l  |------ Magnitude_X_Y magnitude_x_y`
  - `B i l  \------ Non_Max_Supp non_max_supp`

Homework Assignment 5

- Task: Refine the Canny Edge Detector using SCE
  - Setup: use latest **`sce`** to evaluate and refine
    - **`source /opt/sce/bin/setup.csh`**
  - Provided Files
    - canny_a5_start.sc, Makefile
    - golfcart.pgm, ref_golfcart.pgm_s_0.60_l_0.30_h_0.80.pgm
  - Tasks
    - Architecture and Scheduling Refinement (ARM7TDMI plus HW accelerators)
    - Network and Communication Refinement (AMBA_AHB plus DblHsk busses)
    - Instruction Level Model (cycle-accurate instruction-set simulation, optional)
- Deliverables
  - Source file: **`canny.tree`**
  - Description: **`canny.txt`**
- Due
  - By next week: May 25, 2012, 12pm (noon!)

SystemC Overview

- Goals
  - Common C++ Modeling Platform
    - System Level Design
    - HW/SW Codesign
    - RTL
  - Seamless Co-Simulation of HW and SW
  - IP Reuse
  - Free licensing, Open Source
  - De-facto Standard
- Open SystemC Initiative (OSCI)
  - Consortium of many EDA companies
    - Synopsys, Cadence, CoWare, Frontier, ...
  - Open Community (very large!)

SystemC Overview

- Language
  - C++ class library (layered SW architecture)
  - Hierarchy of Modules connected by Ports
  - Communication via Interfaces and Channels
  - Discrete-Event Simulation
- Methodology
  - Untimed Model
  - Transaction-level Model
  - Bus-functional Model
  - Cycle-accurate Model

Introduction to SystemC

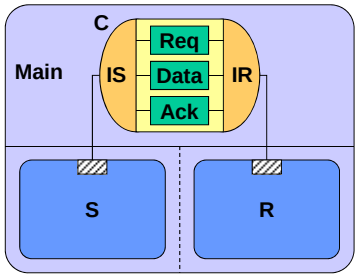
- Presentation by Stuart Swan, Cadence, 2002
  - Goals and Requirements
  - History and Organization
  - Versions, Contents, Coverage
  - Language Architecture
  - Modeling, Models of Computation, Examples
  - Communication Refinement
  - Outlook

Review: Homework Assignment 1

- Task: Introduction to SpecC Compiler and Simulator
  - Become familiar with **`scc`**
    - See **`man scc`** for manual page
  - Use **`scc`** to compile and simulate the examples in
    - **`/opt/sce-20100908/examples/simple/`**
  - Build and simulate a Producer-Consumer example
    - See Slide 25! (behavior **`B`** should be **`Main`**)
    - Producer **`Prod`** should send string **`"Apples and Oranges"`** character by character to the consumer **`Cons`** which prints the received characters to the screen
- Deliverables
  - Source file: **`ProdCons.sc`**
  - Simulation log: **`ProdCons.log`**
- Due
  - By next week: April 20, 2012, 12pm (noon!)

Review: Homework Assignment 1

- Add behavior **`Main`**
- Add loop to **`S`**, name **`Prod`**
- Add loop to **`R`**, name **`Cons`**
- Compile and Simulate
- Create log file




```
interface IS
{
 void Send(float);
};
interface IR
{
 float Receive(void);
};
```

```
behavior S(IS Port)
{
 float X;
 void main(void)
 { ...
   Port.Send(X);
   ...
 }
};
```

```
behavior R(IR Port)
{
 float Y;
 void main(void)
 {...
  Y=Port.Receive();
  ...
 }
};
```

```
channel C
    implements IS, IR
{
 event Req;
 float Data;
 event Ack;

 void Send(float X)
 { Data = X;
   notify Req;
   wait Ack;
 }
 float Receive(void)
 { float Y;
   wait Req;
   Y = Data;
   notify Ack;
   return Y;
 }
};
```


Homework Assignment 6

- Producer/Consumer Example in SystemC
  - Review the FIFO example by Stuart Swan
    - See **`Lecture8_SystemC_Intro.pdf`**
  - Compile and simulate the example using SystemC
    - **`mkdir hw6; cd hw6`**
    - **`cp /opt/pkg/systemc-2.2.0/examples/sysc/simple_fifo/simple_fifo.cpp .`**
    - **`g++ simple_fifo.cpp -I/opt/pkg/systemc-2.2.0/include -L/opt/pkg/systemc-2.2.0/lib-linux –lsystemc -o simple_fifo`**
    - **`./simple_fifo`**
  - Translate the producer/consumer example from Assignment 1 to SystemC and simulate it
    - Reference: **`/home/eecs222/EECS222A_S12/ProdCons.sc`**
- Deliverables
  - Source file: **`SystemC_ProdCons.cpp`**
  - Simulation log: **`SystemC_ProdCons.log`**
- Due
  - By next week: June 1, 2012, 12pm (noon!)