# EECS 222: Embedded System Modeling Lecture 16

Rainer Dömer

doemer@uci.edu

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

## Lecture 16: Overview

- Homework Assignment 7
  - Pipelined/parallel model the Canny Edge Detector
- Simulator run-time facilities
  - Observing simulated time in SystemC
  - Observing simulated time in SpecC
  - Additional tools available for SpecC modeling
    - SIR tools
    - Debugging and tracing support
- System-on-Chip Environment (SCE)
  - Top-down refinement-based design flow
  - Interactive demonstration (part 2)
  - Experimental results

# Homework Assignment 7

- Task: Pipelining and Parallelization of the Canny Model
  - Back-annotate estimated delays to observe timing in the model
  - Pipeline and parallelize the model to maximize throughput
- Steps
  1. Instrument model with logging of simulation time and frame delay
  2. Back-annotate estimated timing in DUT components
  3. Pipeline the DUT into stages for each component
  4. Integrate Gaussian Smooth components into pipeline stages
  5. Slice the BlurX and BlurY blocks into parallel components
- Deliverables
  - **`Canny.sc`** or **`Canny.cpp`** (choose one!)
  - **`Canny.txt`** (with observed timing and frame delays)
- Due: By next week: May 31, 2017, 12pm (Wednesday, noon!)



# Homework Assignment 7

- Task: Pipelining and Parallelization of the Canny Model
  - Expected instance tree

```
DUT canny
|------ Gaussian_Smooth gaussian_smooth
|       |------ Receive_Image receive
|       \------ Gaussian_Kernel gauss
|------ BlurX blurX
|       |------ BlurX_Slice sliceX1
|       |------ BlurX_Slice sliceX2
|       |       [...]
|       \------ BlurX_Slice sliceX8
|------ BlurY blurY
|       |------ BlurY_Slice sliceY1
|       |       [...]
|       \------ BlurY_Slice sliceY8
|------ Derivative_X_Y derivative_x_y
|------ Magnitude_X_Y magnitude_x_y
|------ Non_Max_Supp non_max_supp
\------ Apply_Hysteresis apply_hysteresis
```

# Homework Assignment 7

- Task: Pipelining and Parallelization of the Canny Model
  - Expected execution log with timing (after step 2)

```
          0: Stimulus sent frame  1.
          0: Stimulus sent frame  2.
          0: Stimulus sent frame  3.
          0: Stimulus sent frame  4.
          0: Stimulus sent frame  5.
          0: Stimulus sent frame  6.
    6547500: Monitor received frame  1 with  6547500 us delay.
    6547500: Stimulus sent frame  7.
   13095000: Monitor received frame  1 with 13095000 us delay.
   13095000: Stimulus sent frame  8.
   19642500: Monitor received frame  2 with 19642500 us delay.
   19642500: Stimulus sent frame  9.
  [...]
  111307500: Monitor received frame 16 with 39285000 us delay.
  117855000: Monitor received frame 17 with 39285000 us delay.
  124402500: Monitor received frame 18 with 39285000 us delay.
  130950000: Monitor received frame 19 with 39285000 us delay.
  130950000: Monitor exits simulation.
```



# Homework Assignment 7

- Task: Pipelining and Parallelization of the Canny Model
  - Expected timing results observed after each step:

| Model | Frame Delay | Total simulation time |
|---|---|---|
| CannyA7_step1 | ... us | ... us |
| CannyA7_step2 | ... us | ... us |
| CannyA7_step3 | ... us | ... us |
| CannyA7_step4 | ... us | ... us |
| CannyA7_step5 | ... us | ... us |

# SystemC Simulation

- Compilation and Simulation
  - `g++ DesignName.cpp -I$SYSTEMC/include \ -L$SYSTEMC/lib-linux64 \ -Xlinker -R -Xlinker $SYSTEMC/lib-linux64 \ -lsystemc -o simple_fifo`
  - `./DesignName`
  - Header file `systemc.h`
    - Access to simulation time
      - Time units: `enum sc_time_unit {SC_FS, SC_PS, SC_NS, SC_US, SC_MS, SC_SEC};`
      - Constructor: `sc_time(double, sc_time_unit)`
      - Current simulation time: `sc_time_stamp()`, `sc_delta_count()`
      - Conversion functions: `.to_string().c_str()`
    - Reference: Doulos SystemC Training (part 1, slide 40)



# SystemC Simulation

- Observing Simulated Time in SystemC
- Example: Print the current simulation time

```
#include "systemc.h"
...
sc_time t;
uint64  d;
...
t = sc_time_stamp();  d = sc_delta_count();
printf("Time is now %s pico seconds.\n", t.to_string().c_str());
printf("(delta count is %ull)\n", d);
wait(42000, SC_NS);
printf("Time is now %s pico seconds.\n", t.to_string().c_str());
printf("Time is now %s nano seconds.\n",
                                    (t/1000).to_string().c_str());
...
```

# SpecC Simulation

- Compilation and Simulation
  - **`scc DesignName -sc2out -vv -ww`**
  - **`./DesignName`**
  - Header file **`sim.sh`**
    - Access to simulation time
      - macros **`PICO_SEC`**, **`NANO_SEC`**, **`MICRO_SEC`**, **`MILLI_SEC`**, **`SEC`**
      - typedef **`sim_time`**, **`sim_delta`**, **`sim_time_string`**
      - function **`now()`**, **`delta()`**
      - conversion functions **`time2str()`**, **`str2time()`**
    - Handling of bit vectors
      - conversion functions **`bit2str()`**, **`ubit2str()`**, **`str2bit()`**, **`str2ubit()`**
    - Handling of long-long values
      - conversion functions **`ll2str()`**, **`ull2str()`**, **`str2ll()`**, **`str2ull()`**



# SpecC Simulation

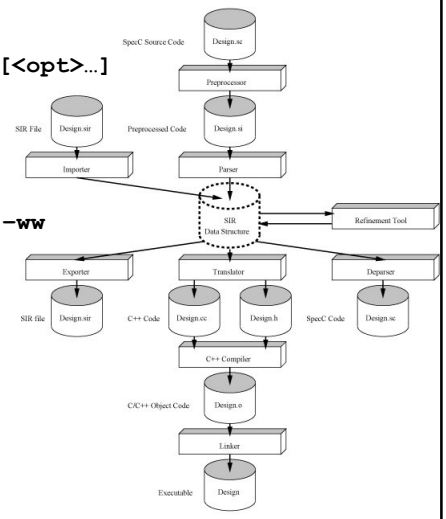
- SpecC Simulation Time
- Example: Print the current simulation time

```
#include <sim.sh>
...
sim_time t;
sim_delta d;
sim_time_string buffer;
...
t = now();  d = delta();
printf("Time is now %s pico seconds.\n", time2str(buffer, t));
printf("(delta count is %s)\n", time2str(buffer, d);
waitfor 42000 NANO_SEC;
printf("Time is now %s pico seconds.\n", time2str(buffer, t));
printf("Time is now %s nano seconds.\n",
            time2str(buffer, t/(1 NANO_SEC)));
...
```

# SpecC Compiler and Simulator

- SpecC Compiler
  - Command line interface
  - Usage: `scc <design> [<cmd>] [<opt>…]`
  - Help: `scc -h`
    `man scc`
  - Example:

```
% scc HelloWorld -sc2out -v -ww
scc: SpecC Compiler V 2.2.1
(c)2012 CECS, UC Irvine
Preprocessing...
Parsing...
Translating...
Compiling...
Linking…
Done.
```






# SpecC Compiler and Simulator

- SpecC SIR Tools
  - Tools working with SpecC Internal Representation (SIR) files
  - Example:

```
% scc Adder -sc2sir -o Adder.sir
% sir_list -t Adder.sir
behavior ADD8
behavior AND2
behavior FA
behavior HA
behavior Main
behavior XOR2
% sir_tree -bt Adder.sir FA
behavior FA
|------ HA ha1
|       |------ AND2 and1
|       \------ XOR2 xor1
|------ HA ha2
|       |------ AND2 and1
|       \------ XOR2 xor1
\------ OR2 or1
```

## SpecC Compiler and Simulator

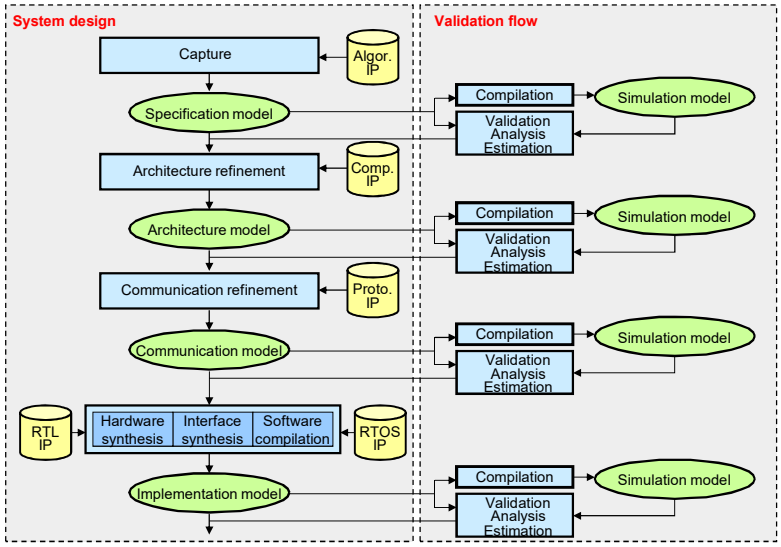
- Debugging
  - **`scc DesignName -sc2out -vv -ww -g -G`**
    **`gdb ./DesignName`**
    **`ddd ./DesignName`**
  - Header file **`sim.sh`**
    - Access to simulation engine state
      - functions **`ready_queue()`**, **`running_queue()`**, etc.
      - functions **`_print_ready_queue()`**, **`_print_running_queue()`**, etc.
      - function **`_print_process_states()`**
      - function **`_print_simulator_state()`**
    - Access to current instance
      - functions **`active_class()`**, **`active_instance()`**
      - functions **`current_class()`**, **`current_instance()`**
      - functions **`print_active_path()`**, **`print_current_path()`**
      - ...



## SpecC Compiler and Simulator

- Tracing
  - **`scc DesignName -sc2out -vv -ww -Tvcds`**
    **`./DesignName`**
    **`gtkwave DesignName.vcd`**
  - Trace instructions in file **`DesignName.do`**
  - Trace log in file **`DesignName.vcd`**
  - Waveform display **`gtkwave`**
    - available as **`/opt/gtkwave/bin/gtkwave`**
- Documentation:
  - E. Johnson, A. Gerstlauer, R. Dömer:
    *"Efficient Debugging and Tracing of System Level Designs"*,
    CECS Technical Report 06-08, May 2006.
  - http://www.cecs.uci.edu/~doemer/publications/CECS_TR_06_08.pdf

## Refinement-based System Design Flow






## SCE Demonstration

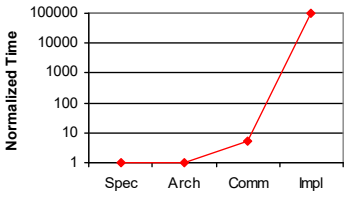
- Application Example: GSM Vocoder
  - Enhanced full-rate voice codec
  - GSM standard for mobile telephony (GSM 06.10)
  - Lossy voice encoding/decoding
    - Incoming speech samples @ 104 kbit/s
    - Encoded bit stream @ 12.2 kbit/s
    - Frames of 4 x 40 = 160 samples (4 x 5ms = 20ms of speech)
  - Real-time constraint:
    - max. 20ms per speech frame
      (max. total of 3.26s for sample speech file)
  - SpecC specification model
    - 29 hierarchical behaviors (9 par, 10 seq, 10 fsm)
    - 73 leaf behaviors
    - 9139 formatted lines of SpecC code
      (~13000 lines of original C code, including comments)

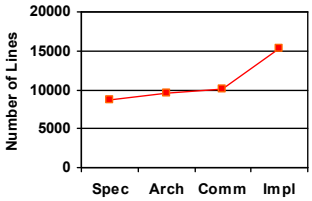
# SCE Experimental Results

- Design Example 1: GSM Vocoder

Simulation Speed




Code Size




Refinement Effort

| | Modified lines | Manual effort | Automated User / Refine |
|---|---|---|---|
| Spec → Arch | 3,275 | 3~4 month | 15 min / < 1 min |
| Arch → Comm | 914 | 1~2 month | 5 min / < 0.5 min |
| Comm → Impl | 6,146 | 5~6 month | 30 min / < 2 min |
| Total | 10,355 | 9~12 month | 50 min / < 4 min |

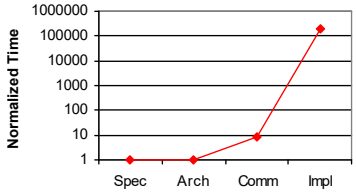
➢ Productivity gain: 12 months vs. 1 hour = 2000x !

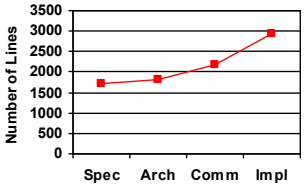
# SCE Experimental Results

- Design Example 2: JPEG Encoder

Simulation Speed




Code Size




Refinement Effort

| | Modified lines | Manual effort | Automated User / Refine |
|---|---|---|---|
| Spec → Arch | 751 | 1~2 month | 5 min / < 0.5 min |
| Arch → Comm | 492 | ~1 month | 3 min / < 0.5 min |
| Comm → Impl | 1,278 | 3~4 month | 20 min / < 1 min |
| Total | 2,521 | 5~7 month | 28 min / < 2 min |

➢ Productivity gain: 6 months vs. 1/2 hour = 2000x !