

ECPS 203

Embedded Systems Modeling and Design

Lecture 18

Rainer Dömer

doemer@uci.edu

Center for Embedded and Cyber-physical Systems
University of California, Irvine



Lecture 18: Overview

- Course Administration
 - Final course evaluation
 - Final report
- Hardware Modeling in Embedded Systems
 - Register Transfer Level (RTL) abstraction
 - RTL modeling in SystemC
 - Example modules

Course Administration

- Final Course Evaluation
 - Open until end of 10th week (Sunday night)
 - Nov. 28, 2017, through Dec. 10, 2017, 11pm
 - Online via EEE Evaluation application
- Mandatory Evaluation of Course and Instructor
 - Voluntary
 - Anonymous
 - Very valuable
- Please spend 5 minutes for this survey!
 - Your feedback is appreciated!

ECPS203: Embedded Systems Modeling and Design, Lecture 18

(c) 2017 R. Doemer

3

Course Administration

- Final Report (in lieu of Final Exam)
 - Allocated time and room for final exam
 - Monday, December 11, 8:00-10:00am (DBH 1420)
 - *Not applicable, we use electronic submission instead!*
 - Format: Final Project Report
 - Submission script: `~ecps203/bin/turnin.sh`
 - Directory name: `final`
 - Deliverables: `ECPS203_Report.pdf`
`Canny.cpp`
 - Soft deadline: Draft report (for early feedback)
 - Monday, December 11, 2017, 10am
 - Hard deadline: Final report (graded!)
 - Wednesday, December 13, 2017, 6pm

ECPS203: Embedded Systems Modeling and Design, Lecture 18

(c) 2017 R. Doemer

4

Final Project Report

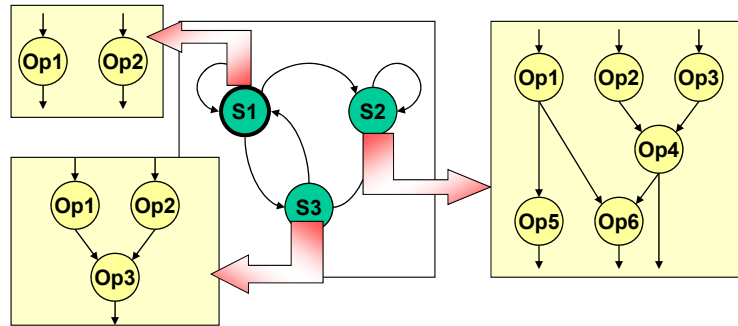
- Technical Report about the Course Project
 - Title
 - *Specification and Modeling of a Canny Edge Detector for Embedded Systems Design*
 - Contents
 - “Story” of the Canny Edge Detector project
 - From downloading the initial C reference code
 - Via modeling and simulating in SystemC
 - To performance optimization for real-time video
 - Describe the project assignments 1 through 9
 - Focus on the reasoning and the optimization results
 - Length
 - About 12 pages (including title page, figures, and bibliography)

Final Project Report

1. Title page
 - Project title, author, date, course number and title
 - Abstract
2. Introduction
 - Embedded system modeling and design concepts
 - The IEEE SystemC language
3. Case Study of a Canny Edge Detector for Real-time Video
 - Structure of the Canny edge detection algorithm
 - Modeling and simulation in IEEE SystemC
 - Model refinement for pipelining and parallelization
 - Performance estimation and throughput optimization
 - Real-time video performance results
4. Summary and Conclusion
 - Lessons learned
 - Future work
5. References

Hardware Modeling in Embedded Systems

- Register Transfer Level (RTL) Abstraction
 - FSM Model: Finite State Machine with Data
 - Combined model for control and computation
 - Implementation: controller plus datapath



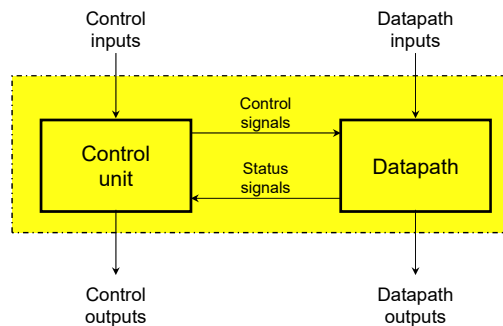
ECPS203: Embedded Systems Modeling and Design, Lecture 18

(c) 2017 R. Doemer

7

Hardware Modeling in Embedded Systems

- Register Transfer Level (RTL) Abstraction
 - Block diagram of a generic RTL component (high level)



Source:
<http://www.eda.org/alc-cwg/cwg-open.pdf>

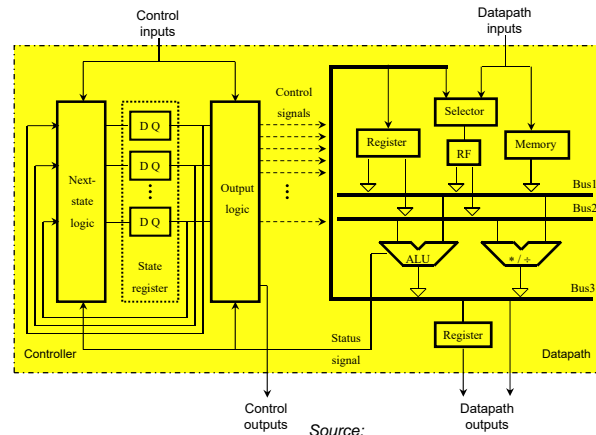
ECPS203: Embedded Systems Modeling and Design, Lecture 18

(c) 2017 R. Doemer

8

Hardware Modeling in Embedded Systems

- Register Transfer Level (RTL) Abstraction
 - Block diagram of a generic RTL component (low level)



ECPS203: Embedded Systems Modeling and Design, Lecture 18

(c) 2017 R. Doemer

9

Hardware Modeling in Embedded Systems

- Register Transfer Level (RTL) Synthesis
 - Allocation of RTL components
 - Type and number of functional units
 - Type and number of storage units
 - Type and number of interconnecting busses
 - Scheduling
 - Representation of basic blocks as super-states
 - Scheduling of operations to clock cycles
 - Binding
 - Bind functional operations to functional units
 - Bind variables to storage units
 - Bind assignments/transfers to busses
 - Result:
 - Clock-cycle and pin accurate model at RTL abstraction

ECPS203: Embedded Systems Modeling and Design, Lecture 18

(c) 2017 R. Doemer

10

RTL Modeling in SystemC

- Example Modules of RTL Components
 - D-FF
 - D-FF with Asynchronous Reset
 - Shifter
 - Counter
 - State machine
 - Memory
- Source: Aleksandar Milenkovic (ECE at UAH)
- CPE 626: *The SystemC Language – VHDL, Verilog Designer's Guide*
- homepages.cae.wisc.edu/~ece734/SystemC/cpe626-SystemC-L2.ppt

RTL Modeling in SystemC

- Example Modules of RTL Components
 - D-FF

```

// dff.h
#include "systemc.h"
SC_MODULE(dff)
{
    sc_in<bool> din;
    sc_in<bool> clock;
    sc_out<bool> dout;

    void doit()
    {
        dout = din;
    };
    SC_CTOR(dff)
    {
        SC_METHOD(doit);
        sensitive_pos << clock;
    }
};

```

- Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components

- D-FF with asynchronous reset

```
// dffa.h
#include "systemc.h"
SC_MODULE(dffa)
{
  sc_in<bool> clock;
  sc_in<bool> reset;
  sc_in<bool> din;
  sc_out<bool> dout;
  void do_ffa()
  {
    if (reset) {
      dout = false;
    } else if (clock.event()) {
      dout = din;
    }
  };
  SC_CTOR(dffa) {
    SC_METHOD(do_ffa);
    sensitive(reset);
    sensitive_pos(clock);
  }
};
```

- Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components

- Shifter

```
// shift.h
#include "systemc.h"
SC_MODULE(shift)
{
  sc_in<sc_bv<8> > din;
  sc_in<bool> clk;
  sc_in<bool> load;
  sc_in<bool> LR;
  sc_out<sc_bv<8> > dout;
  sc_bv<8> shiftval;
  void shifty();
  SC_CTOR(shift)
  {
    SC_METHOD(shifty);
    sensitive_pos (clk);
  }
};
```

```
// shift.cc
#include "shift.h"
void shift::shifty()
{
  if (load) {
    shiftval = din;
  } else if (!LR) {
    shiftval.range(6,0) = shiftval.range(7,1);
    shiftval[7] = '0';
  } else if (LR) {
    shiftval.range(7,1) = shiftval.range(6,0);
    shiftval[0] = '0';
  }
  dout = shiftval;
}
```

- Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components
 - Counter

```
#include "systemc.h"
SC_MODULE(counter)
{
    sc_in<bool> clock;
    sc_in<bool> load;
    sc_in<bool> clear;
    sc_in<sc_int<8> > din;
    sc_out<sc_int<8> > dout;
    int countval;
    void onetwothree();

    SC_CTOR(counter)
    {
        SC_METHOD(onetwothree);
        sensitive_pos (clock);
    }
};
```

```
// counter.cc
#include "counter.h"
void counter::onetwothree()
{
    if (clear) {
        countval = 0;
    } else if (load) {
        countval = din.read(); // use read when a
        // type conversion is happening
        // from an input port
    } else {
        countval++;
    }
    dout = countval;
}
```

- Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components
 - State Machine
 - Voicemail controller
 - States
 - Main
 - Send
 - Review
 - Repeat, Erase, Record
 - Outputs
 - play, recrd, erase, save and address
 - Two SC_METHOD processes
 - getnextst – calculates the next state based on input and current state
 - setstate – copies the calculated next_state to the current_state every positive clock edge on input clk
 - Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components
 - State Machine

```
// stmach.h
#include "systemc.h"
enum vm_state {
    main_st, review_st, repeat_st,
    save_st, erase_st, send_st,
    address_st, record_st,
    begin_rec_st, message_st
};
SC_MODULE(stmach)
{
    sc_in<bool> clk;
    sc_in<char> key;
    sc_out<sc_logic> play;
    sc_out<sc_logic> recrd;
    sc_out<sc_logic> erase;
    sc_out<sc_logic> save;
    sc_out<sc_logic> address;
    sc_signal<vm_state> next_state;
    sc_signal<vm_state> current_state;
};
```

```
void getnextst();
void setstate();

SC_CTOR(stmach)
{
    SC_METHOD(getnextst);
    sensitive << key << current_state;
    SC_METHOD(setstate);
    sensitive_pos (clk);
}
};
```

- Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components
 - State Machine (cont'd)

```
// stmach.cc
#include "stmach.h"
void stmach::getnextst()
{
    play = sc_logic_0;
    recrd = sc_logic_0;
    erase = sc_logic_0;
    save = sc_logic_0;
    address = sc_logic_0;
    switch (current_state) {
        case main_st:
            if (key == '1') {
                next_state = review_st;
            } else {
                if (key == '2') {
                    next_state = send_st;
                } else {
                    next_state = main_st;
                }
            }
    }
}
```

```
case review_st:
    if (key == '1') {
        next_state = repeat_st;
    } else {
        if (key == '2') {
            next_state = save_st;
        } else {
            if (key == '3') {
                next_state = erase_st;
            } else {
                if (key == '#') {
                    next_state = main_st;
                } else {
                    next_state = review_st;
                }
            }
        }
    }
}
```

- Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components
 - State Machine (cont'd)

```

case repeat_st:
    play = sc_logic_1;
    next_state = review_st;
case save_st:
    save = sc_logic_1;
    next_state = review_st;
case erase_st:
    erase = sc_logic_1;
    next_state = review_st;
case send_st:
    next_state = address_st;
case address_st:
    address = sc_logic_1;
    if (key == '#') {
        next_state = record_st;
    } else {
        next_state = address_st;
    }
}

```

- Source: Aleksandar Milenkovic (ECE at UAH)

```

case record_st:
    if (key == '5') {
        next_state = begin_rec_st;
    } else {
        next_state = record_st;
    }
case begin_rec_st:
    recrd = sc_logic_1;
    next_state = message_st;
case message_st:
    recrd = sc_logic_1;
    if (key == '#') {
        next_state = send_st;
    } else {
        next_state = message_st;
    }
} // end switch
} // end method
void stmach::setstate()
{
    current_state = next_state;
}

```

ECPS203: Embedded Systems Modeling and Design, Lecture 18

(c) 2017 R. Doemer

19

RTL Modeling in SystemC

- Example Modules of RTL Components
 - Memory

```

// ram.h
#include "systemc.h"
SC_MODULE(ram)
{
    sc_in<sc_int<8> > addr;
    sc_in<bool> enable;
    sc_in<bool> readwr;
    sc_inout_rv<16> data;

    void read_data();
    void write_data();
    sc_lv<16> ram_data[256];

    SC_CTOR(ram)
    {
        SC_METHOD(read_data);
        sensitive << addr << enable << readwr;
        SC_METHOD(write_data);
        sensitive << addr << enable << readwr << data;
    }
};

```

- Source: Aleksandar Milenkovic (ECE at UAH)

ECPS203: Embedded Systems Modeling and Design, Lecture 18

(c) 2017 R. Doemer

20

RTL Modeling in SystemC

- Example Modules of RTL Components

- Memory (cont'd)

```

// ram.cc
#include "ram.h"
void ram::read_data()
{
    if (enable && ! readwr ) {
        data = ram_data[addr.read()];
    } else {
        data = "ZZZZZZZZZZZZZZZZZZ";
    }
}

void ram::write_data()
{
    if (enable && readwr) {
        ram_data[addr.read()] = data;
    }
}
    
```

- Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components

- D-FF
 - D-FF with Asynchronous Reset
 - Shifter
 - Counter
 - State machine
 - Memory

- Source: Aleksandar Milenkovic (ECE at UAH)

