EECS 222: Embedded System Modeling Lecture 17

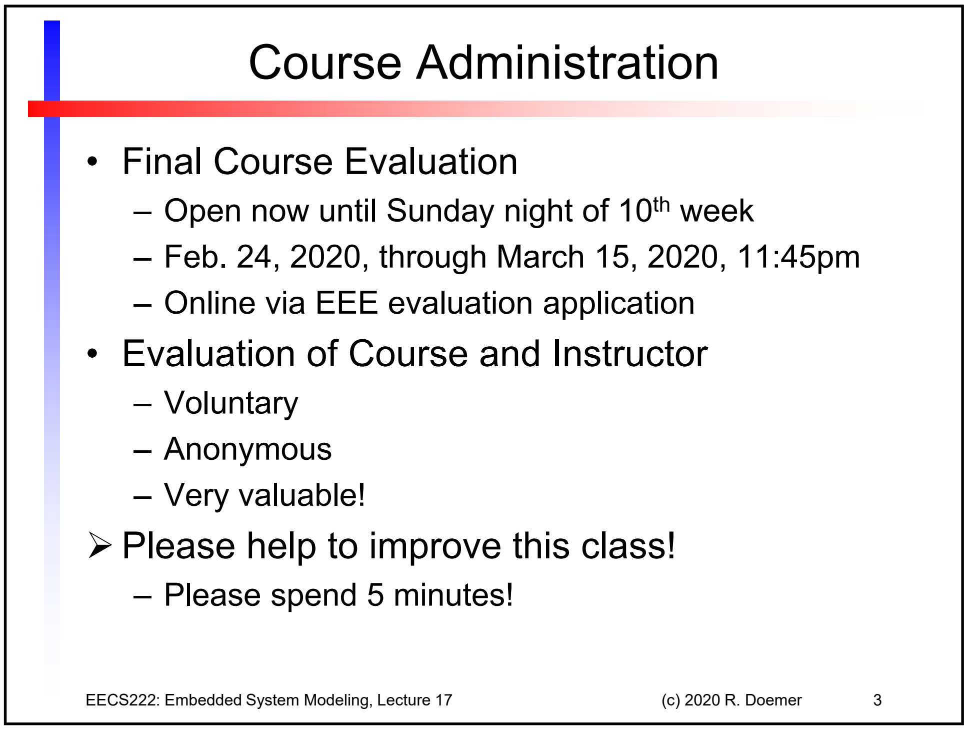
Rainer Dömer

doemer@uci.edu

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

Lecture 17: Overview

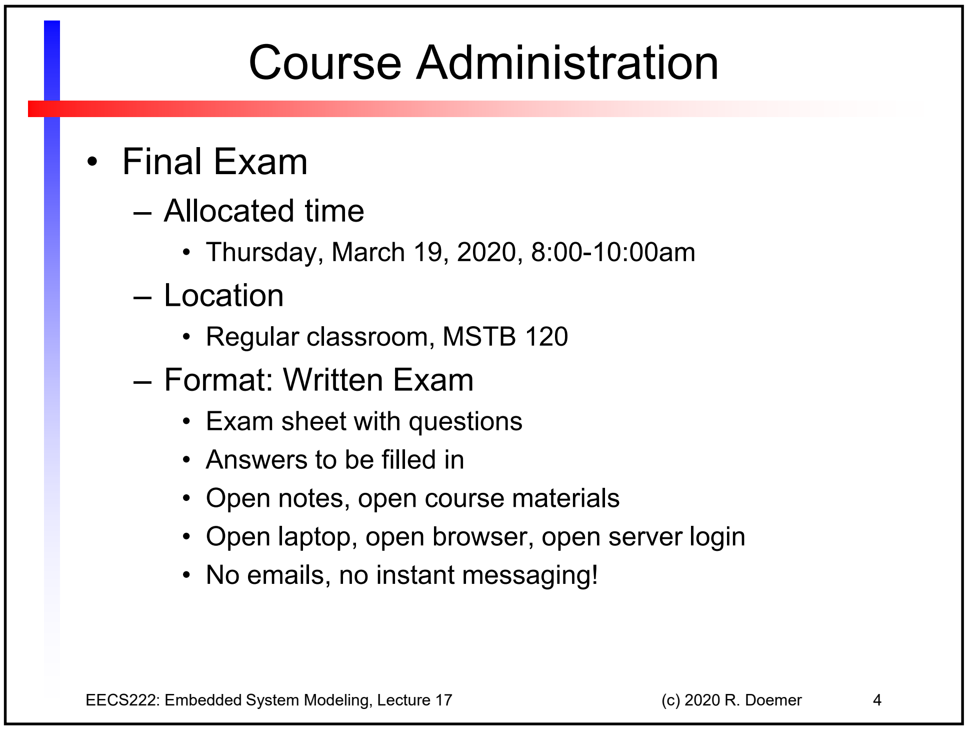
- Course Administration
  - Course evaluation
  - Final exam
- Modeling of Hardware in Embedded Systems
  - RTL component modeling in SystemC
- Communication Abstractions
  - Bus Functional Model (BFM)
  - Transaction Level Modeling (TLM)
  - Transaction Level Modeling, TLM-2.0
- SystemC TLM-2.0
  - *"The Definitive Guide to SystemC: TLM-2.0 and the IEEE 1666-2011 Standard"*
  - by David Black, Doulos



Course Administration

- Final Course Evaluation
  - Open now until Sunday night of 10th week
  - Feb. 24, 2020, through March 15, 2020, 11:45pm
  - Online via EEE evaluation application
- Evaluation of Course and Instructor
  - Voluntary
  - Anonymous
  - Very valuable!
- ➢ Please help to improve this class!
  - Please spend 5 minutes!

EECS222: Embedded System Modeling, Lecture 17 (c) 2020 R. Doemer 3



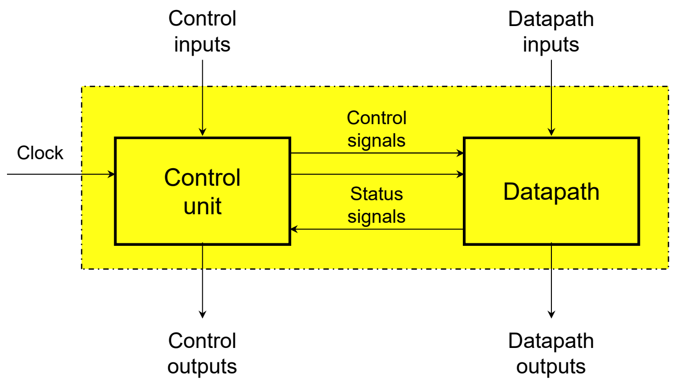
Course Administration

- Final Exam
  - Allocated time
    - Thursday, March 19, 2020, 8:00-10:00am
  - Location
    - Regular classroom, MSTB 120
  - Format: Written Exam
    - Exam sheet with questions
    - Answers to be filled in
    - Open notes, open course materials
    - Open laptop, open browser, open server login
    - No emails, no instant messaging!

EECS222: Embedded System Modeling, Lecture 17 (c) 2020 R. Doemer 4

Hardware Modeling in Embedded Systems

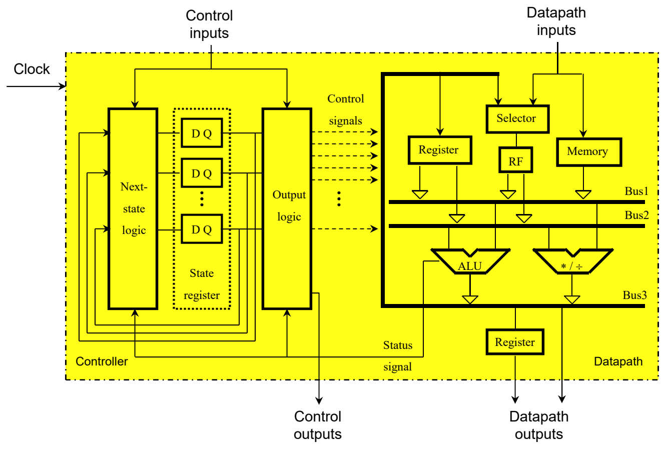
- Register Transfer Level (RTL) Abstraction
  - Block diagram of generic RTL component (high level)



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

Hardware Modeling in Embedded Systems

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



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

RTL Modeling in SystemC

- Example Modules of RTL Components
  - D-FF
  - D-FF with Asynchronous Reset
  - Memory
  - Shifter
  - Counter
  - State machine

➢ 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
  - 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)

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 = “ZZZZZZZZZZZZZZZZ”;
  }
}

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

- Source: Aleksandar Milenkovic (ECE at UAH)

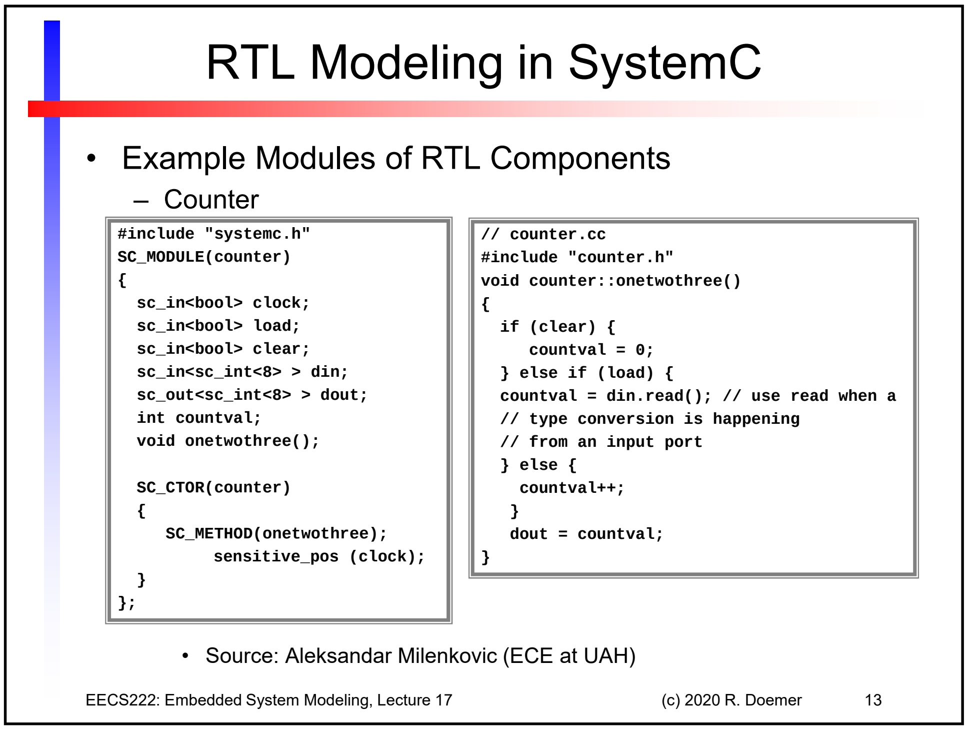
RTL Modeling in SystemC

- Example Modules of RTL Components
  - 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

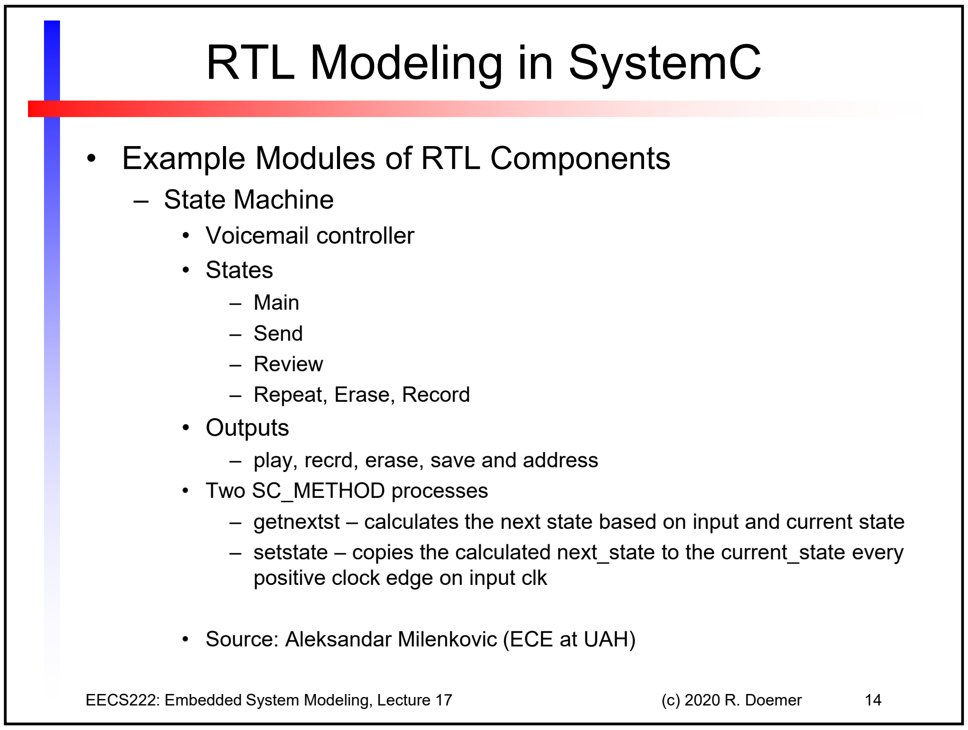
```
#include "systemc.h"
SC_MODULE(counter)
{
  sc_in<bool> clock;
  sc_in<bool> load;
  sc_in<bool> clear;
  sc_in<sc_int<8> > din;
  sc_out<sc_int<8> > dout;
  int countval;
  void onetwothree();

  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)

EECS222: Embedded System Modeling, Lecture 17 (c) 2020 R. Doemer 13

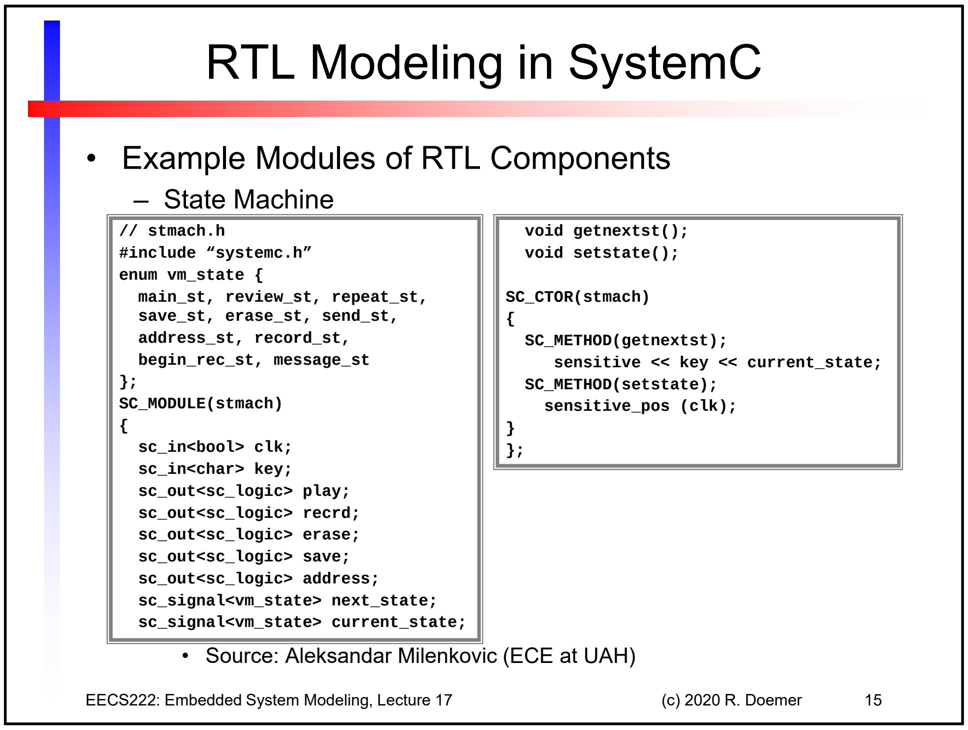


RTL Modeling in SystemC

- 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)

EECS222: Embedded System Modeling, Lecture 17 (c) 2020 R. Doemer 14



RTL Modeling in SystemC

- Example Modules of RTL Components
  - State Machine

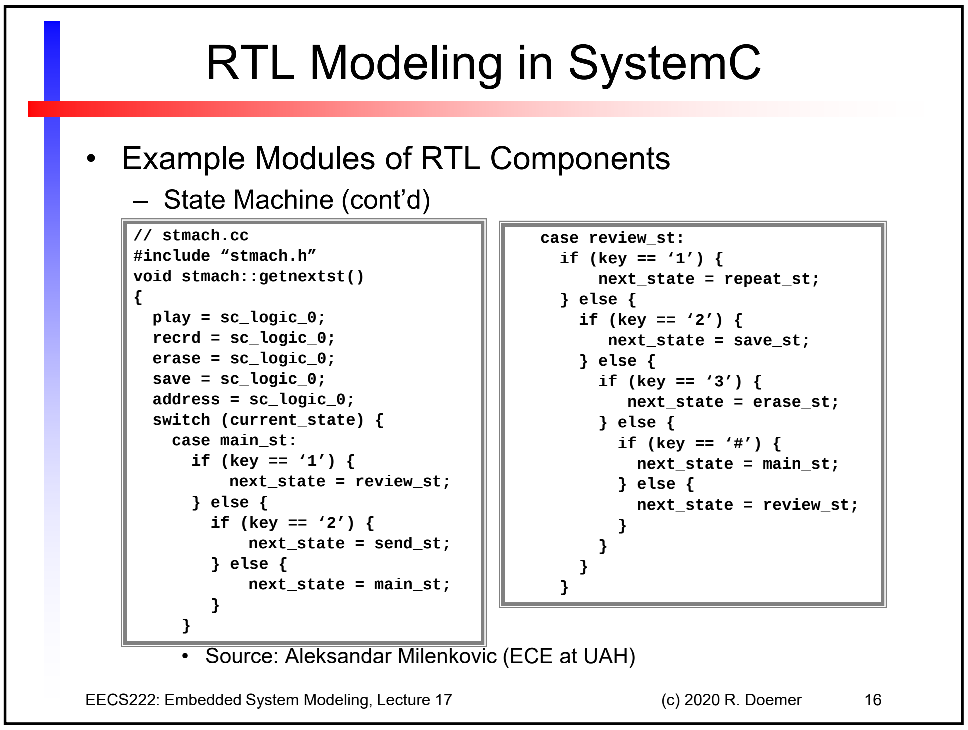
```
// stmach.h
#include "systemc.h"
enum vm_state {
  main_st, review_st, repeat_st,
  save_st, erase_st, send_st,
  address_st, record_st,
  begin_rec_st, message_st
};
SC_MODULE(stmach)
{
  sc_in<bool> clk;
  sc_in<char> key;
  sc_out<sc_logic> play;
  sc_out<sc_logic> recrd;
  sc_out<sc_logic> erase;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

EECS222: Embedded System Modeling, Lecture 17 (c) 2020 R. Doemer 15



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)

EECS222: Embedded System Modeling, Lecture 17 (c) 2020 R. Doemer 16

RTL Modeling in SystemC

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

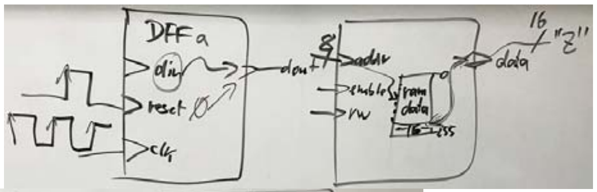
```
  case repeat_st:
     play = sc_logic_1;
     next_state = review_st;
  case save_st:
     save = sc_logic_1;
     next_state = review_st;
  case erase_st:
     erase = sc_logic_1;
     next_state = review_st;
  case send_st:
     next_state = address_st;
  case address_st:
     address = sc_logic_1;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

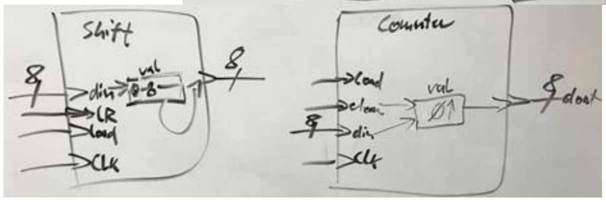
```
   case record_st:
      if (key == '5') {
        next_state = begin_rec_st;
      } else {
        next_state = record_st;
      }
    case begin_rec_st:
      recrd = sc_logic_1;
      next_state = message_st;
    case message_st:
      recrd = sc_logic_1;
      if (key == '#') {
         next_state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

- Source: Aleksandar Milenkovic (ECE at UAH)

RTL Modeling in SystemC

- Example Modules of RTL Components
  - Source: Aleksandar Milenkovic (ECE at UAH)
  - D-FF
  - D-FF with Asynchronous Reset
  - Memory
  - Shifter
  - Counter
  - State machine



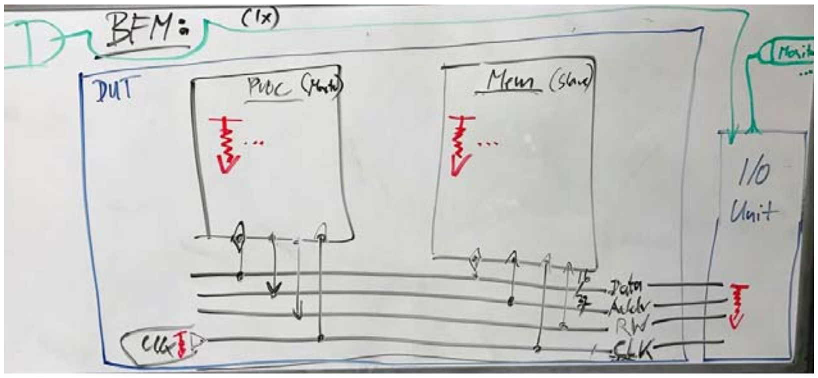


Communication Abstractions

- Communication Modeling
  - Bus Functional Model (BFM)
    - Cycle-accurate, pin-accurate communication
    - ➢ Ports, signal assignments
    - ➢ Slow (1x)
  - Transaction Level Modeling (TLM)
    - Modules/behaviors, channels, interfaces
    - ➢ Method calls
    - ➢ Fast (100x)
  - New Transaction Level Modeling: SystemC TLM-2.0
    - Initiators, targets, sockets
    - ➢ Method calls with Direct Memory Interface (DMI)
    - ➢ Very fast (1000x)

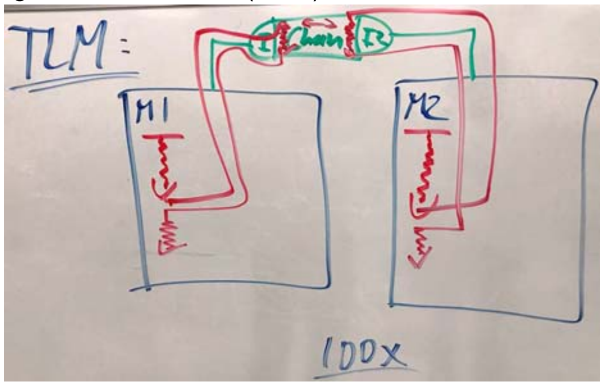
Communication Abstractions

- Traditional Bus Functional Model (BFM)
  - Cycle-accurate, pin-accurate communication
  - ➢ Ports, signal assignments
  - ➢ Lowest abstraction, slow (1x)



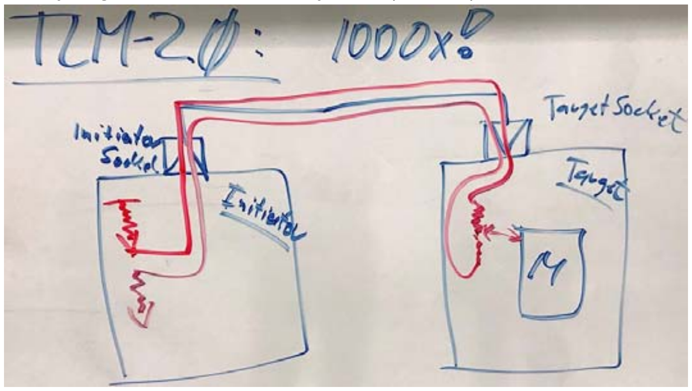
Communication Abstractions

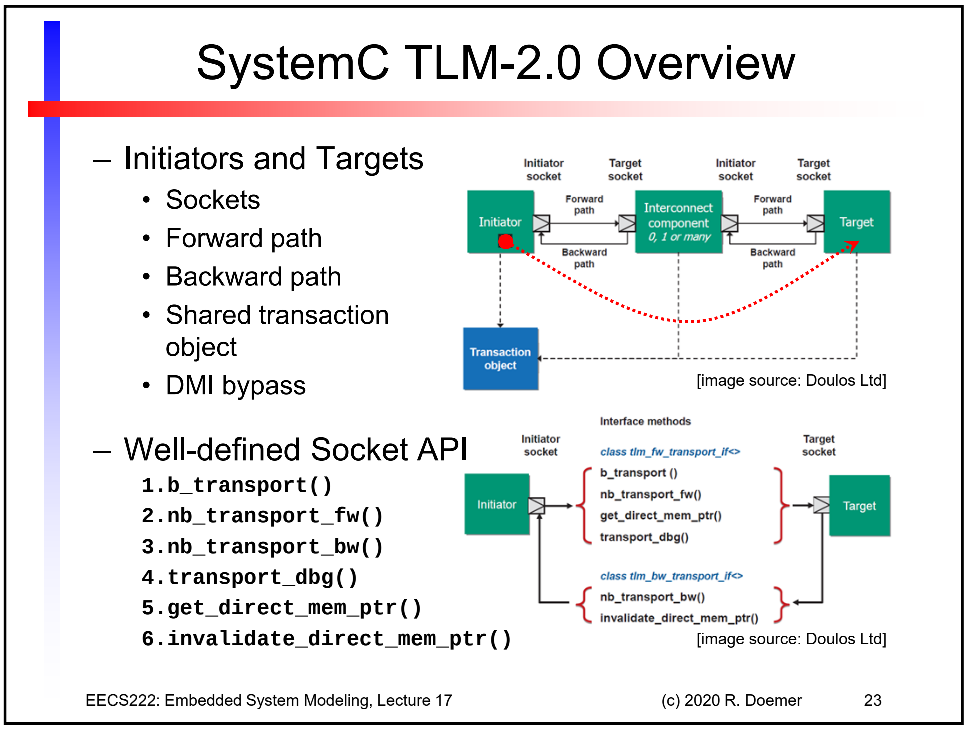
- Regular Transaction Level Modeling: TLM 1.0
  - Modules/behaviors, channels, interfaces
  - ➢ Method calls
  - ➢ High abstraction, fast (100x)



Communication Abstractions

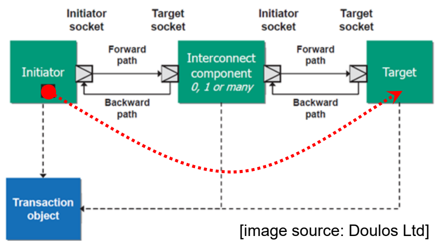
- New Transaction Level Modeling: TLM-2.0
  - Initiators, targets, sockets
  - ➢ Method calls with Direct Memory Interface (DMI)
  - ➢ Very high abstraction, very fast (1000x)





SystemC TLM-2.0 Overview

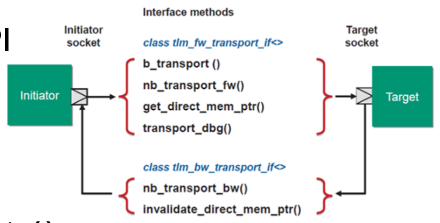
- Initiators and Targets
  - Sockets
  - Forward path
  - Backward path
  - Shared transaction object
  - DMI bypass



[image source: Doulos Ltd]

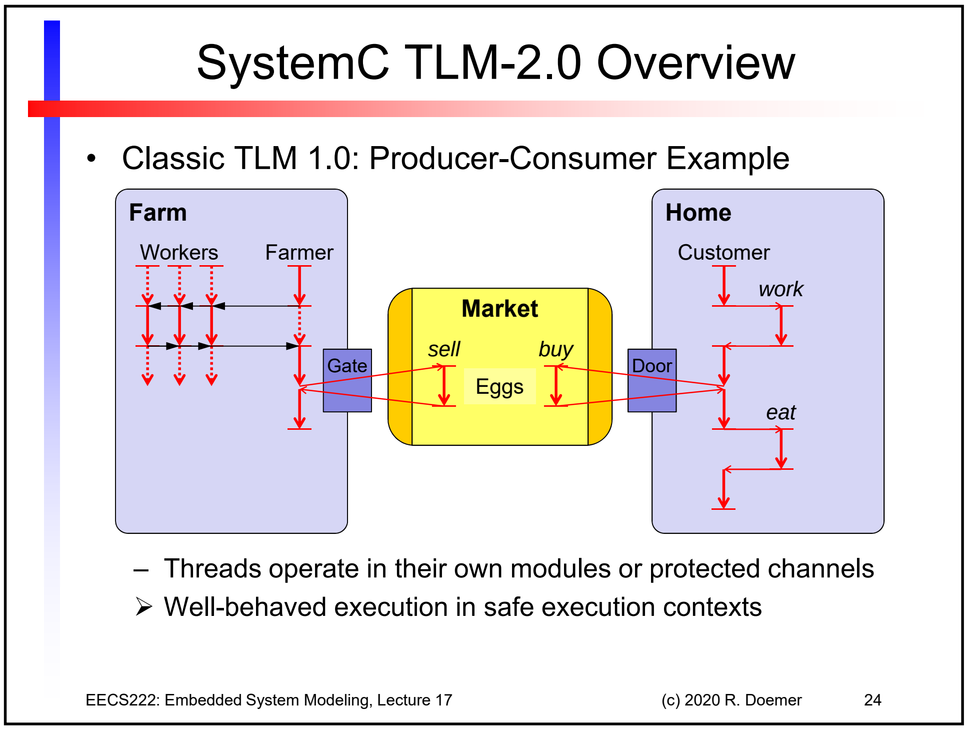
- Well-defined Socket API

```
1.b_transport()
2.nb_transport_fw()
3.nb_transport_bw()
4.transport_dbg()
5.get_direct_mem_ptr()
6.invalidate_direct_mem_ptr()
```



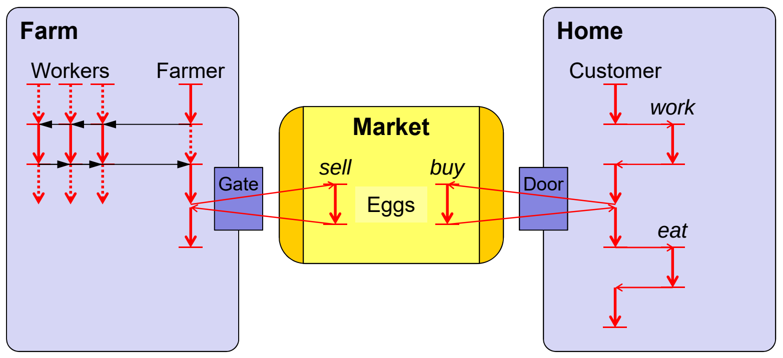
[image source: Doulos Ltd]

EECS222: Embedded System Modeling, Lecture 17 (c) 2020 R. Doemer 23



SystemC TLM-2.0 Overview

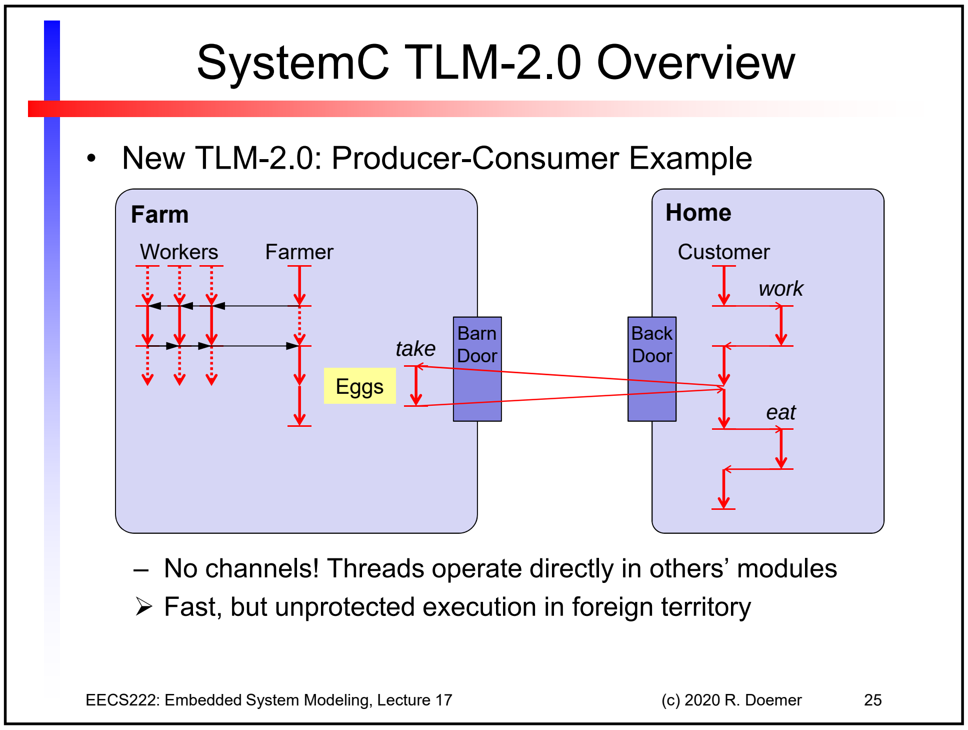
- Classic TLM 1.0: Producer-Consumer Example



– Threads operate in their own modules or protected channels

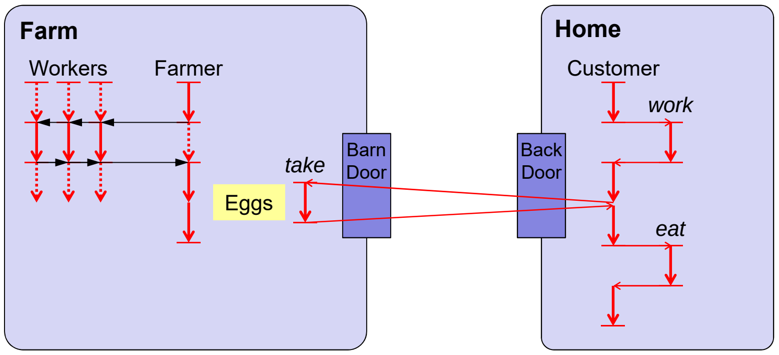
➢ Well-behaved execution in safe execution contexts

EECS222: Embedded System Modeling, Lecture 17 (c) 2020 R. Doemer 24

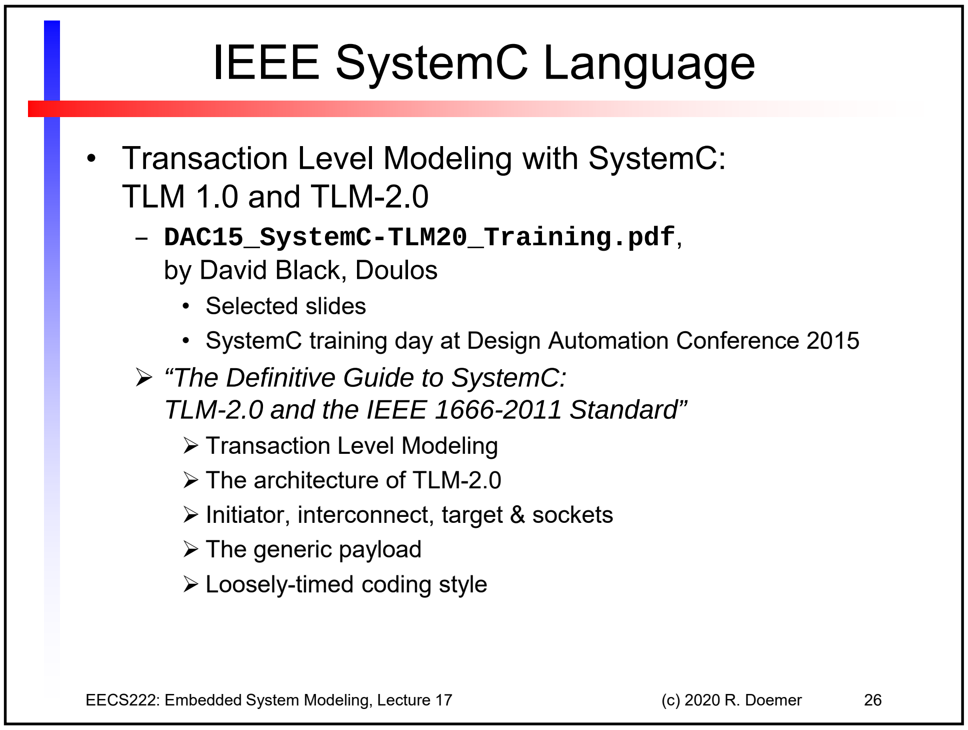


SystemC TLM-2.0 Overview

- New TLM-2.0: Producer-Consumer Example



  – No channels! Threads operate directly in others’ modules
  ➢ Fast, but unprotected execution in foreign territory



IEEE SystemC Language

- Transaction Level Modeling with SystemC: TLM 1.0 and TLM-2.0
  – **`DAC15_SystemC-TLM20_Training.pdf`**, by David Black, Doulos
    - Selected slides
    - SystemC training day at Design Automation Conference 2015
  ➢ *“The Definitive Guide to SystemC: TLM-2.0 and the IEEE 1666-2011 Standard”*
    ➢ Transaction Level Modeling
    ➢ The architecture of TLM-2.0
    ➢ Initiator, interconnect, target & sockets
    ➢ The generic payload
    ➢ Loosely-timed coding style