

# ECPS 203

## Embedded Systems Modeling and Design

### Lecture 4

Rainer Dömer

doemer@uci.edu

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



## Lecture 4: Overview

- Review: System Modeling Concepts
  - Structural hierarchy
  - Behavioral hierarchy
- System Modeling Concepts
  - Communication
  - Synchronization
  - Time
- Project Discussion
  - Assignment 1
  - Assignment 2

## System Modeling Concepts

- Structural Hierarchy
  - *Classes and instances*
  - Top behavior/module
  - Child behavior/module
  - Channel
  - Interface
  - Variable (wire)
  - Port

ECPS203: Embedded Systems Modeling and Design, Lecture 4 (c) 2018 R. Doemer 3

## System Modeling Concepts

- Behavioral hierarchy (in SpecC)
 

**Sequential execution**

**FSM execution**

**Concurrent execution**

**Pipelined execution**

**Exception handling, abortion**

**Exception handling, interrupt**

ECPS203: Embedded Systems Modeling and Design, Lecture 4 (c) 2018 R. Doemer 4

## System Modeling Concepts

- Behavioral hierarchy: Limited support in SystemC

**Sequential execution**

**B\_seq**

**FSM execution**

**B\_fsm**

**Concurrent execution**

**B\_par**

**Pipelined execution**

**B\_pipe**

**Exception handling, abortion**

**B1**

**Exception handling, interrupt**

**B2**

ECPS203: Embedded Systems Modeling and Design, Lecture 4
(c) 2018 R. Doemer
5

## System Modeling Concepts

- Communication and Synchronization
  - via shared variables and events
  - via channels with interfaces
  - via hierarchical channels

**Shared memory**

**Message passing**

**Protocol stack**

ECPS203: Embedded Systems Modeling and Design, Lecture 4
(c) 2018 R. Doemer
6

## System Modeling Concepts

- Synchronization (SpecC)
  - Event type
    - `event <event_List>;`
  - Synchronization primitives
    - `wait <event_list>;`
    - `notify <event_list>;`

```

behavior S(out event Req,
           out float Data,
           in event Ack)
{
  float X;
  void main(void)
  { ...
    Data = X;
    notify Req;
    wait Ack;
    ...
  }
};

behavior R(in event Req,
           in float Data,
           out event Ack)
{
  float Y;
  void main(void)
  { ...
    wait Req;
    Y = Data;
    notify Ack;
    ...
  }
};
                    
```

ECPS203: Embedded Systems Modeling and Design, Lecture 4
(c) 2018 R. Doemer
7

## System Modeling Concepts

- Communication (SpecC)
  - Interface class
    - `interface <name>`  
`{ <declarations>;`
  - Channel class
    - `channel <name>`  
`implements <interfaces>`  
`{ <implementations>;`

```

interface IS
{
  void Send(float);
};

interface IR
{
  float Receive(void);
};

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;
  }
};

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

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

ECPS203: Embedded Systems Modeling and Design, Lecture 4
(c) 2018 R. Doemer
8

## System Modeling Concepts

- Hierarchical Channel
  - Virtual channel implemented by standard bus protocol
    - SpecC example: simplified PCI bus

```

interface PCI_IF
{
  void Transfer(
    enum Mode,
    int NumBytes,
    int Address);
};

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

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

```

interface IS
{
  void Send(float);
};

interface IR
{
  float Receive(void);
};

channel PCI
  implements PCI_IF;

channel C2
  implements IS, IR
{
  PCI Bus;
  void Send(float X)
  { Bus.Transfer(
    PCI_WRITE,
    sizeof(X), &X);
  }

  float Receive(void)
  { float Y;
    Bus.Transfer(
    PCI_READ,
    sizeof(Y), &Y);
    return Y;
  }
};
                    
```

ECPS203: Embedded Systems Modeling and Design, Lecture 4
(c) 2018 R. Doemer
9

## System Modeling Concepts

- Time
  - Timing delay
    - `waitfor <delay>;`

**Example: Stimulus for a test bench**

**Example (SpecC)**

```

behavior Stimulus
  (inout int a,
   inout bit[4] b,
   out event e1,
   out event e2)
{
  void main(void)
  {
    waitfor 5;
    a = 42;
    notify e1;

    waitfor 5;
    b = 1010b;
    notify e2;

    waitfor 10;
    a++;
    b |= 0101b;
    notify e1, e2;

    waitfor 10;
    b = 0;
    notify e2;
  }
};
                    
```

ECPS203: Embedded Systems Modeling and Design, Lecture 4
(c) 2018 R. Doemer
10

## System Modeling Concepts

- Time
  - Timing delay
    - **waitfor** <delay>;
  - Timing constraints
    - **do** { <actions> }  
**timing** {<constraints>}

**Example: SRAM read protocol**

**Specification (SpecC)**

```

bit[7:0] Read_SRAM(bit[15:0] a)
{
  bit[7:0] d;

  do { t1: {ABus = a; }
      t2: {RMode = 1;
           WMode = 0; }
      t3: { }
      t4: {d = Dbus; }
      t5: {ABus = 0; }
      t6: {RMode = 0;
           WMode = 0; }
      t7: { }
    }
  timing { range(t1; t2; 0; );
           range(t1; t3; 10; 20);
           range(t2; t3; 10; 20);
           range(t3; t4; 0; );
           range(t4; t5; 0; );
           range(t5; t7; 10; 20);
           range(t6; t7; 5; 10);
         }
  return(d);
}
                
```

ECPS203: Embedded Systems Modeling and Design, Lecture 4
(c) 2018 R. Doemer
11

## System Modeling Concepts

- Time
  - Timing delay
    - **waitfor** <delay>;
  - Timing constraints
    - **do** { <actions> }  
**timing** {<constraints>}

**Example: SRAM read protocol**

**Implementation 1 (SpecC)**

```

bit[7:0] Read_SRAM(bit[15:0] a)
{
  bit[7:0] d;

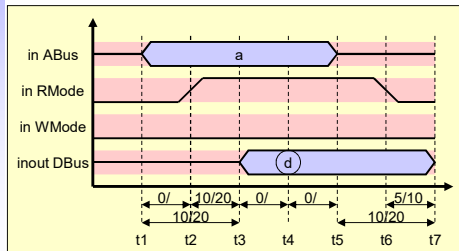
  do { t1: {ABus = a; waitfor( 2);}
      t2: {RMode = 1;
           WMode = 0; waitfor(12);}
      t3: { waitfor( 5);}
      t4: {d = Dbus; waitfor( 5);}
      t5: {ABus = 0; waitfor( 2);}
      t6: {RMode = 0;
           WMode = 0; waitfor(10);}
      t7: { }
    }
  timing { range(t1; t2; 0; );
           range(t1; t3; 10; 20);
           range(t2; t3; 10; 20);
           range(t3; t4; 0; );
           range(t4; t5; 0; );
           range(t5; t7; 10; 20);
           range(t6; t7; 5; 10);
         }
  return(d);
}
                
```

ECPS203: Embedded Systems Modeling and Design, Lecture 4
(c) 2018 R. Doemer
12

## System Modeling Concepts

- Time
  - Timing delay
    - **waitfor** <delay>;
  - Timing constraints
    - **do** { <actions> }  
**timing** {<constraints>}

Example: SRAM read protocol



### Implementation 2 (SpecC)

```

bit[7:0] Read_SRAM(bit[15:0] a)
{
    bit[7:0] d; // ASAP Schedule

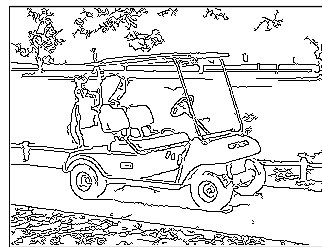
    do { t1: {ABus = a; }
        t2: {RMode = 1;
            WMode = 0; waitfor(10);}
        t3: { }
        t4: {d = Dbus; }
        t5: {ABus = 0; }
        t6: {RMode = 0;
            WMode = 0; waitfor(10);}
        t7: { }
    }
    timing { range(t1; t2; 0; );
            range(t1; t3; 10; 20);
            range(t2; t3; 10; 20);
            range(t3; t4; 0; );
            range(t4; t5; 0; );
            range(t5; t7; 10; 20);
            range(t6; t7; 5; 10);
        }
    return(d);
}
    
```

## ECPS 203 Project

- Application Example: Canny Edge Detector
  - Embedded system model for image processing:  
Automatic edge detection in a digital camera



golfcart.pgm



golfcart.pgm\_s\_0.60\_l\_0.30\_h\_0.80.pgm

- Application source and documentation:
  - [http://marathon.csee.usf.edu/edge/edge\\_detection.html](http://marathon.csee.usf.edu/edge/edge_detection.html)
  - [http://en.wikipedia.org/wiki/Canny\\_edge\\_detector](http://en.wikipedia.org/wiki/Canny_edge_detector)

## Project Assignment 1

- Task: Introduction to Application Example
  - Canny Edge Detector
  - Algorithm for edge detection in digital images
- Steps
  1. Setup your Linux programming environment
  2. Download, adjust, and compile the application C code with the GNU C compiler (`gcc`)
  3. Study the application, determine function-call tree
- Deliverables
  - Source code and text file: `canny.c`, `canny.txt`
- Due
  - Wednesday, October 10, 2018, 6pm

ECPS203: Embedded Systems Modeling and Design, Lecture 4

(c) 2018 R. Doemer

15

## Project Assignment 2

- Task: Clean C++ model with static memory allocation
  - Prepare the C++ source code for modeling in SystemC
  - Configure parameters for specific application
  - Apply static memory allocation
- Steps
  1. Fix the off-by-one bug in the `non_max_supp` function
  2. Clean-up the code for compilation without warnings
  3. Fix configuration parameters to compile-time constants
  4. Remove or replace dynamic memory allocation
- Deliverables
  - Source code and text file: `canny.cpp`, `canny.txt`
- Due
  - Wednesday, next week: October 17, 2018, 6pm

ECPS203: Embedded Systems Modeling and Design, Lecture 4

(c) 2018 R. Doemer

16