

ECPS 203

Embedded Systems Modeling and Design

Lecture 3

Rainer Dömer

doemer@uci.edu

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



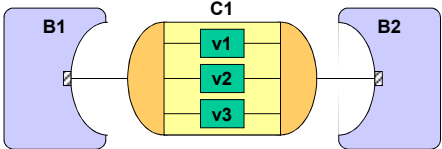
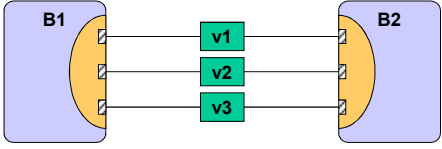
Lecture 3: Overview

- Separation of Concerns
 - Separating computation and communication
 - From system model to implementation model
- System Modeling Concepts
 - C/C++ foundation
 - Structural hierarchy
 - Behavioral hierarchy

Separation of Concerns

- System Model
 - Specification
 - Validation
 - Exploration
 - SystemC, SpecC
 - Computation in modules or behaviors
 - Communication in channels

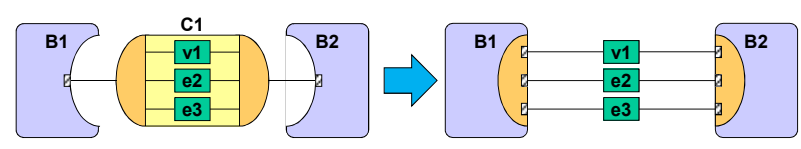
- Implementation Model
- Synthesis
 - SystemC, SpecC
 - or traditional Verilog, VHDL
 - Communication protocol is *inlined*
 - Channel disappears, signals get exposed

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

Separation of Concerns

- Communication Protocol Inlining



– SystemC example:

```
SC_MODULE (M1)
{
  [...]
  c.send(1);

  xy = x + 2 * y;

  c.send(xy);

  v1 = 0;
  [...]
}
```

```
SC_CHANNEL (C1)
{
  [...]
  send(int d)
  { v1 = d;
    e2.notify();
    wait (e3);
  }
  [...]
}
```

➔

```
SC_MODULE (M1)
{
  [...]
  v1 = 1;
  e2.notify();
  wait (e3);
  xy = x + 2 * y;
  v1 = xy;
  e2.notify();
  wait (e3);
  v1 = 0;
  [...]}

```

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

System Modeling Concepts

- System-Level Description Language (SLDL)
 - SpecC: Original concept language (used to explain concepts)
 - SystemC: IEEE and de-facto standard (taught in detail)
- Foundation: C/C++
 - Software requirements are fully covered
 - SpecC is a superset of ANSI-C (with its own compiler)
 - SystemC is a superset of C++ (class library for system modeling)
 - Leverage of large set of existing programs
 - Every C/C++ program is a SLDL program
 - Well-known, well-established, solid foundation
- Extensions needed for hardware and system models
 - Hardware types, simulation, synthesis
 - System composition and simulation

ECPS203: Embedded Systems Modeling and Design, Lecture 3

(c) 2018 R. Doemer

5

System Modeling Concepts

- SLDL Data Types
 - Support for all C/C++ basic types
 - predefined types (`char`, `int`, `float`, `double`, ...)
 - composite and user-defined types (`array`, `struct`, `union`, `enum`)
 - Logical types: Truth values, multi-value logic (0, 1, X, Z)
 - `bool b1 = true; sc_logic b2 = 'Z';`
 - Bit vector and fixed-point types: Types with arbitrary length
 - `sc_bv<16> bv = "1111000011110000";`
 - `sc_lv<16> lv = "11110000xxxxzzzz";`
 - `const sc_ufixed<19,3> PI("3.141592654");`
 - Event type: Synchronization of concurrent threads
 - `sc_event e;`
 - Signal type: RTL modeling
 - `sc_signal< sc_bv<16> > address_bus;`
 - `sc_signal_rv<16> data_bus;`

ECPS203: Embedded Systems Modeling and Design, Lecture 3

(c) 2018 R. Doemer

6

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 3 (c) 2018 R. Doemer 7

System Modeling Concepts

- Structural Hierarchy (SpecC syntax)

```

interface I1
{
    bit[63:0] Read(void);
    void Write(bit[63:0]);
};

channel C1 implements I1;

behavior B1(in int, I1, out int);

behavior B(in int p1, out int p2)
{
    int v1;
    C1 c1;
    B1 b1(p1, c1, v1),
    b2(v1, c1, p2);

    void main(void)
    { par {
        b1;
        b2;
    }
};
    
```

SpecC 2.0:
 if *b* is a behavior instance,
b; is equivalent to *b.main()* ;

ECPS203: Embedded Systems Modeling and Design, Lecture 3 (c) 2018 R. Doemer 8

System Modeling Concepts

- Structural Hierarchy: Top-level behavior/module
 - Test bench, typically called **Main** or **Top**
 - Stimulus provides test vectors
 - Design under test (DUT) represents the target design
 - Monitor observes and validates DUT outputs

ECPS203: Embedded Systems Modeling and Design, Lecture 3 (c) 2018 R. Doemer 9

System Modeling Concepts

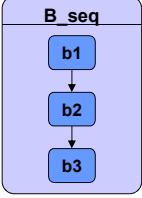
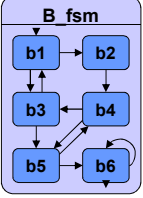
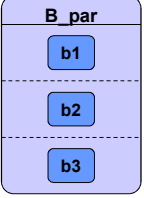
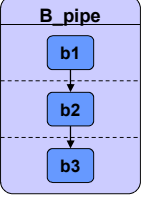
- Behavioral hierarchy (SpecC)

Sequential execution	FSM execution	Concurrent execution	Pipelined execution
<pre>behavior B_seq { B b1, b2, b3; void main(void) { b1; b2; b3; } };</pre>	<pre>behavior B_fsm { B b1, b2, b3, b4, b5, b6; void main(void) { fsm { b1: {...} b2: {...} ... } } };</pre>		

ECPS203: Embedded Systems Modeling and Design, Lecture 3 (c) 2018 R. Doemer 10

System Modeling Concepts

- Behavioral hierarchy (SpecC)

Sequential execution	FSM execution	Concurrent execution	Pipelined execution
			
<pre>behavior B_seq { B b1, b2, b3; void main(void) { b1; b2; b3; } };</pre>	<pre>behavior B_fsm { B b1, b2, b3, b4, b5, b6; void main(void) { fsm { b1: {...} b2: {...} ... } } };</pre>	<pre>behavior B_par { B b1, b2, b3; void main(void) { par { b1; b2; b3; } } };</pre>	<pre>behavior B_pipe { B b1, b2, b3; void main(void) { pipe { b1; b2; b3; } } };</pre>

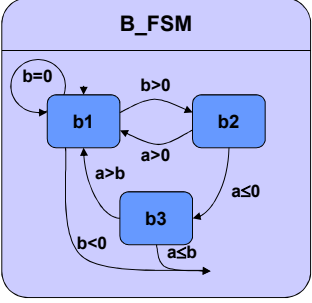
ECPS203: Embedded Systems Modeling and Design, Lecture 3
(c) 2018 R. Doemer
11

System Modeling Concepts

- Behavioral Hierarchy: Finite State Machine (SpecC)
 - Explicit state transitions
 - triple $\langle \text{current_state}, \text{condition}, \text{next_state} \rangle$
 - `fsm { <current_state> : { if <condition> goto <next_state> } ... }`
 - Moore-type FSM
 - Mealy-type FSM

```
behavior B_FSM(in int a, in int b)
{
  B b1, b2, b3;

  void main(void)
  {
    fsm { b1: { if (b<0) break;
                if (b==0) goto b1;
                if (b>0) goto b2; }
          b2: { if (a>0) goto b1; }
          b3: { if (a>b) goto b1; }
        }
  }
};
```



ECPS203: Embedded Systems Modeling and Design, Lecture 3
(c) 2018 R. Doemer
12

System Modeling Concepts

- Behavioral Hierarchy: Pipeline (SpecC)
 - Explicit execution in pipeline fashion
 - `pipe { <instance_list> };`

Pipeline

```
behavior Pipeline
{
    Stage1 b1;
    Stage2 b2;
    Stage3 b3;

    void main(void)
    {
        pipe
        {
            b1;
            b2;
            b3;
        }
    }
};
```

ECPS203: Embedded Systems Modeling and Design, Lecture 3
(c) 2018 R. Doemer
13

System Modeling Concepts

- Behavioral Hierarchy: Pipeline (SpecC)
 - Explicit execution in pipeline fashion
 - `pipe { <instance_list> };`
 - `pipe (<init>; <cond>; <incr>){ ... }`

Pipeline

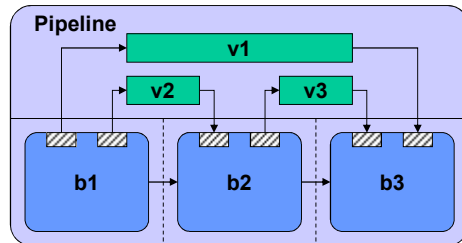
```
behavior Pipeline
{
    Stage1 b1;
    Stage2 b2;
    Stage3 b3;

    void main(void)
    {
        int i;
        pipe (i=0; i<10; i++)
        {
            b1;
            b2;
            b3;
        }
    }
};
```

ECPS203: Embedded Systems Modeling and Design, Lecture 3
(c) 2018 R. Doemer
14

System Modeling Concepts

- Behavioral Hierarchy: Pipeline (SpecC)
 - Explicit execution in pipeline fashion
 - `pipe { <instance_list> };`
 - `pipe (<init>; <cond>; <incr>){ ... }`
 - Support for automatic buffering



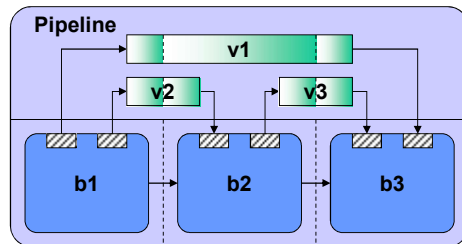
```
behavior Pipeline
{
  int v1;
  int v2;
  int v3;

  Stage1 b1(v1, v2);
  Stage2 b2(v2, v3);
  Stage3 b3(v3, v1);

  void main(void)
  {
    int i;
    pipe (i=0; i<10; i++)
    {
      b1;
      b2;
      b3;
    }
  }
};
```

System Modeling Concepts

- Behavioral Hierarchy: Pipeline (SpecC)
 - Explicit execution in pipeline fashion
 - `pipe { <instance_list> };`
 - `pipe (<init>; <cond>; <incr>){ ... }`
 - Support for automatic buffering
 - `piped [...] <type> <variable_list>;`



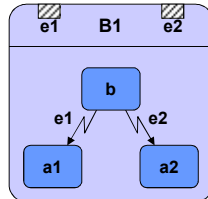
```
behavior Pipeline
{
  piped piped int v1;
  piped int v2;
  piped int v3;

  Stage1 b1(v1, v2);
  Stage2 b2(v2, v3);
  Stage3 b3(v3, v1);

  void main(void)
  {
    int i;
    pipe (i=0; i<10; i++)
    {
      b1;
      b2;
      b3;
    }
  }
};
```


System Modeling Concepts

- Behavioral Hierarchy: Exception handling (SpecC)
 - Abortion
 - Interrupt



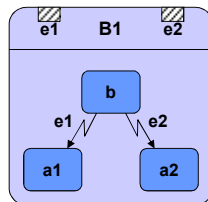
```

behavior B1(in event e1, in event e2)
{
  B b, a1, a2;

  void main(void)
  { try { b; }
    trap (e1) { a1; }
    trap (e2) { a2; }
  }
};
    
```

System Modeling Concepts

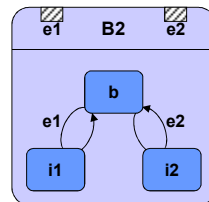
- Behavioral Hierarchy: Exception handling (SpecC)
 - Abortion
 - Interrupt



```

behavior B1(in event e1, in event e2)
{
  B b, a1, a2;

  void main(void)
  { try { b; }
    trap (e1) { a1; }
    trap (e2) { a2; }
  }
};
    
```



```

behavior B2(in event e1, in event e2)
{
  B b, i1, i2;

  void main(void)
  { try { b; }
    interrupt (e1) { i1; }
    interrupt (e2) { i2; }
  }
};
    
```

System Modeling Concepts

- Behavioral hierarchy (in SpecC)

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 3
(c) 2018 R. Doemer
19

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 3
(c) 2018 R. Doemer
20