EECS 222A: System-on-Chip Description and Modeling Lecture 7

Rainer Dömer

doemer@uci.edu

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

Lecture 7: Overview

- Homework Assignment 4
  - Discussion
- Modeling of Hardware in SoC Design
  - Hardware Refinement
  - Register Transfer Level (RTL) Model
- RTL Modeling in SpecC
  - **`bit`**, **`bit4`** bit-vector types
  - **`buffered`** type modifier
  - **`signal`** type modifier
  - **`fsmd`** statement
- Homework Assignment 5
  - Creating behaviors from C code
  - ....

Homework Assignment 4

- Task
  - Creating behaviors in C code
    - initial system consists of
      - Test bench behavior Main
      - Stimulus behavior
      - Design Under Test (DUT)
      - Monitor behavior
    - we want to create more hierarchy in DUT
  - See posted detailed instructions!

Homework Assignment 4

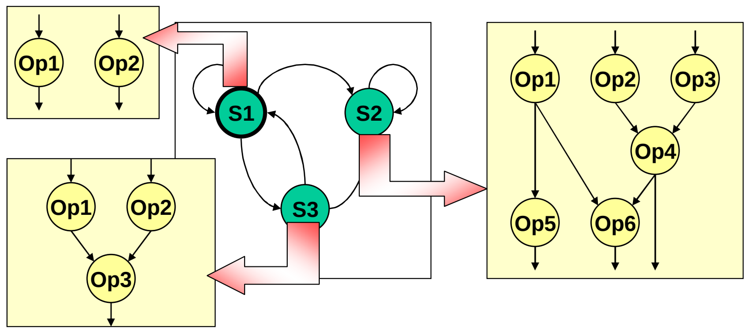
- Deliverables
  - 1-paragraph description about the two tasks
    - How far did you get?
    - What were the problems?
    - How did you solve it?
  - Report the time stamps
    - How long did it take to do the tasks?
- Due
  - Week 7 (this week)

Modeling of Hardware in SoC Design

- Hardware Refinement (for each HW block)
  - Allocation of hardware 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 transfers to busses
  - Result:
    - Clock-cycle accurate HW model

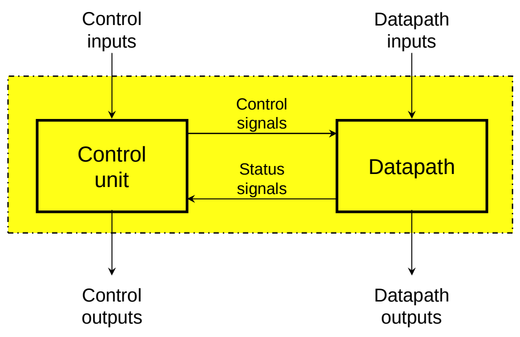
Modeling of Hardware in SoC Design

- Hardware Modeling
  - FSMD Model: Finite State Machine with Data
    - Combined model for control and computation
    - Implementation: controller plus datapath



Modeling of Hardware in SoC Design

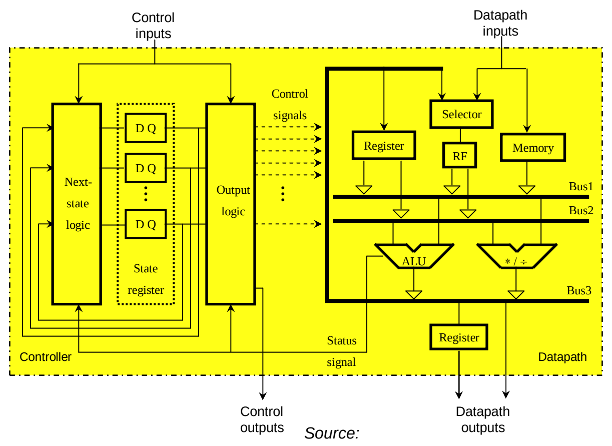
- Register Transfer Level (RTL) Modeling
  - Block diagram of generic RTL component (high level)



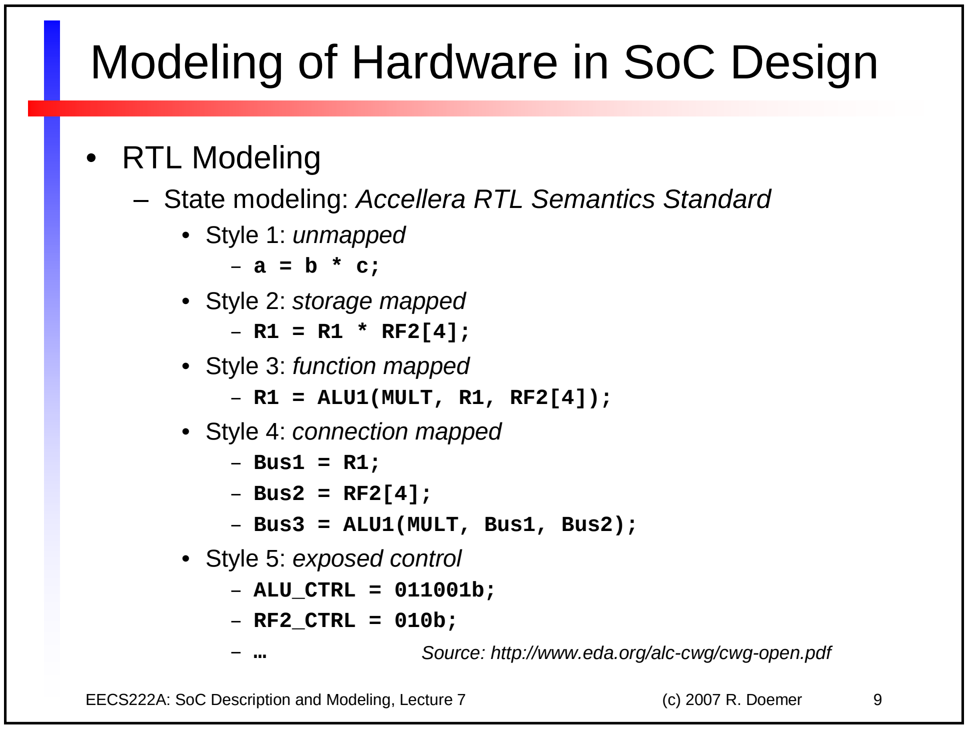
Source: http://www.eda.org/alc-cwg/cwg-open.pdf

Modeling of Hardware in SoC Design

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



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

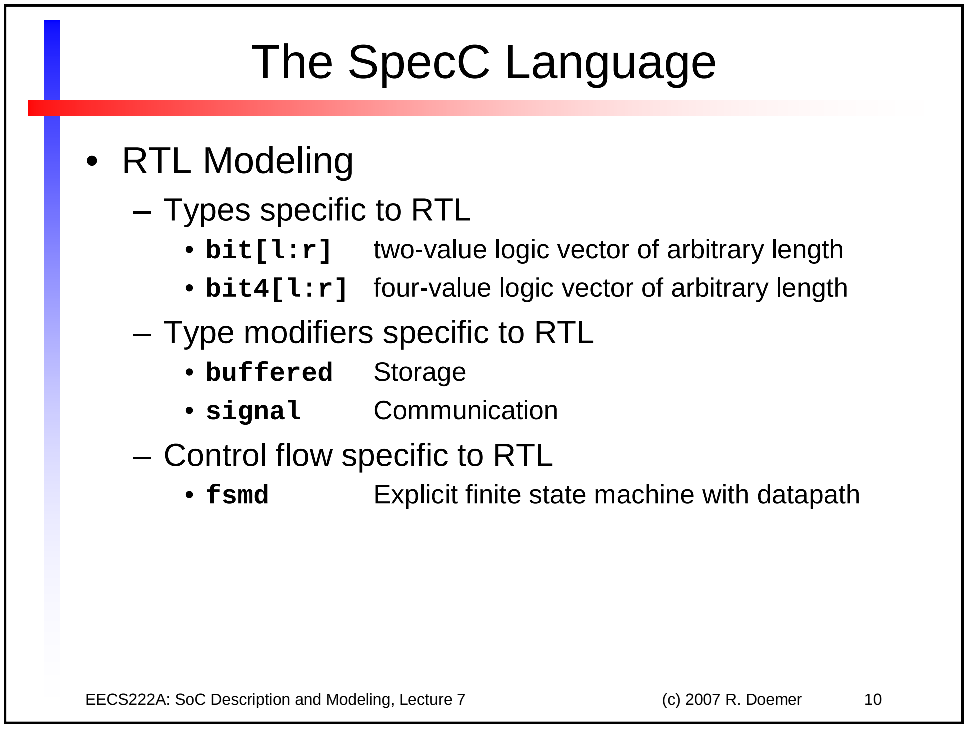


Modeling of Hardware in SoC Design

- RTL Modeling
  - State modeling: *Accellera RTL Semantics Standard*
    - Style 1: *unmapped*
      - **`a = b * c;`**
    - Style 2: *storage mapped*
      - **`R1 = R1 * RF2[4];`**
    - Style 3: *function mapped*
      - **`R1 = ALU1(MULT, R1, RF2[4]);`**
    - Style 4: *connection mapped*
      - **`Bus1 = R1;`**
      - **`Bus2 = RF2[4];`**
      - **`Bus3 = ALU1(MULT, Bus1, Bus2);`**
    - Style 5: *exposed control*
      - **`ALU_CTRL = 011001b;`**
      - **`RF2_CTRL = 010b;`**
      - **…**

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

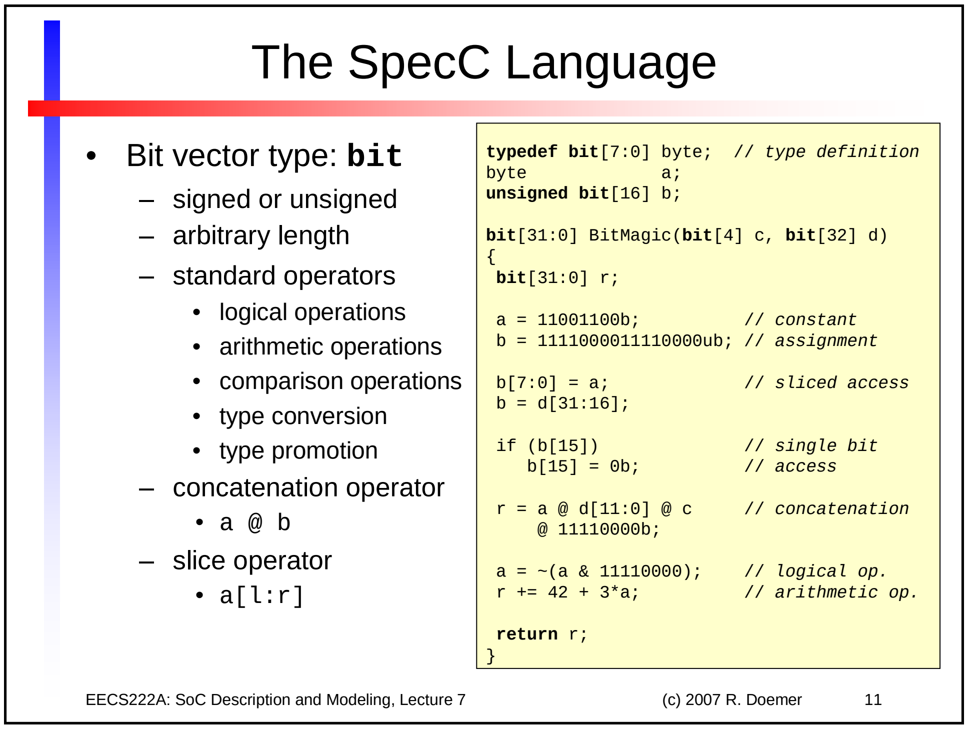
EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 9



The SpecC Language

- RTL Modeling
  - Types specific to RTL
    - **`bit[l:r]`** two-value logic vector of arbitrary length
    - **`bit4[l:r]`** four-value logic vector of arbitrary length
  - Type modifiers specific to RTL
    - **`buffered`** Storage
    - **`signal`** Communication
  - Control flow specific to RTL
    - **`fsmd`** Explicit finite state machine with datapath

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 10

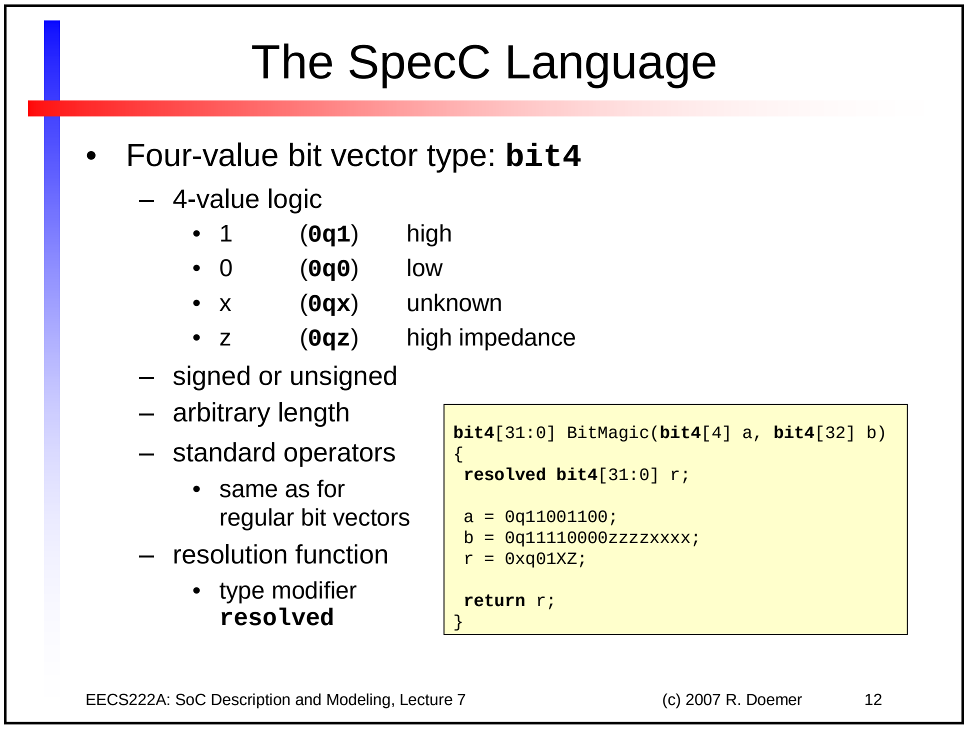


The SpecC Language

- Bit vector type: **`bit`**
  - signed or unsigned
  - arbitrary length
  - standard operators
    - logical operations
    - arithmetic operations
    - comparison operations
    - type conversion
    - type promotion
  - concatenation operator
    - `a @ b`
  - slice operator
    - `a[l:r]`

```
typedef bit[7:0] byte;  // type definition
byte             a;
unsigned bit[16] b;

bit[31:0] BitMagic(bit[4] c, bit[32] d)
{
 bit[31:0] r;

 a = 11001100b;          // constant
 b = 1111000011110000ub; // assignment

 b[7:0] = a;             // sliced access
 b = d[31:16];

 if (b[15])              // single bit
    b[15] = 0b;          // access

 r = a @ d[11:0] @ c     // concatenation
     @ 11110000b;

 a = ~(a & 11110000);    // logical op.
 r += 42 + 3*a;          // arithmetic op.

 return r;
}
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 11



The SpecC Language

- Four-value bit vector type: **`bit4`**
  - 4-value logic
    - 1 (**`0q1`**) high
    - 0 (**`0q0`**) low
    - x (**`0qx`**) unknown
    - z (**`0qz`**) high impedance
  - signed or unsigned
  - arbitrary length
  - standard operators
    - same as for regular bit vectors
  - resolution function
    - type modifier **`resolved`**

```
bit4[31:0] BitMagic(bit4[4] a, bit4[32] b)
{
 resolved bit4[31:0] r;

 a = 0q11001100;
 b = 0q11110000zzzzxxxx;
 r = 0xq01XZ;

 return r;
}
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 12

The SpecC Language

- **`buffered`** type modifier
  - Representation of storage in RTL models
    - register
    - register file
    - memory
  - Update at notification of specified events
    - synchronized with explicit clock

```
event Clk1, Clk2;                // system clock
buffered[Clk1] bit[32] R1;       // register
buffered[Clk1] bit[32] R2;

buffered[CLK2] bit[16] RF[64];   // register file
buffered[CLK2] bit[ 8] M1[1024]; // memory

R1 = R2;      // swap contents of R1 and R2
R2 = R1;
wait CLK1;

RF[2] = RF[0] + RF[1];
...
wait CLK2;
```

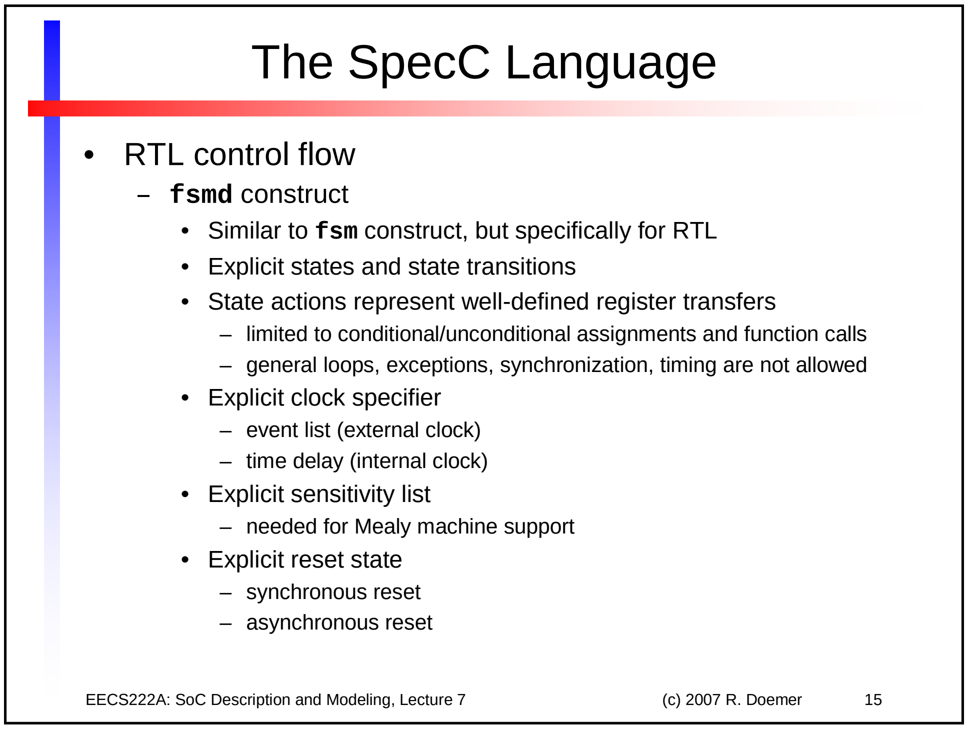
The SpecC Language

- **`signal`** type modifier
  - Representation of wires and busses in RTL models
  - Semantics as in VHDL, Verilog

```
signal bit[31:0] addr;   // address bus
signal bit[31:0] data;   // data bus
buffered[CLK] M[1024];

wait addr;               // memory read access
data = M[addr];
...

wait addr && data;
M[addr] = data;          // memory write access
...
```

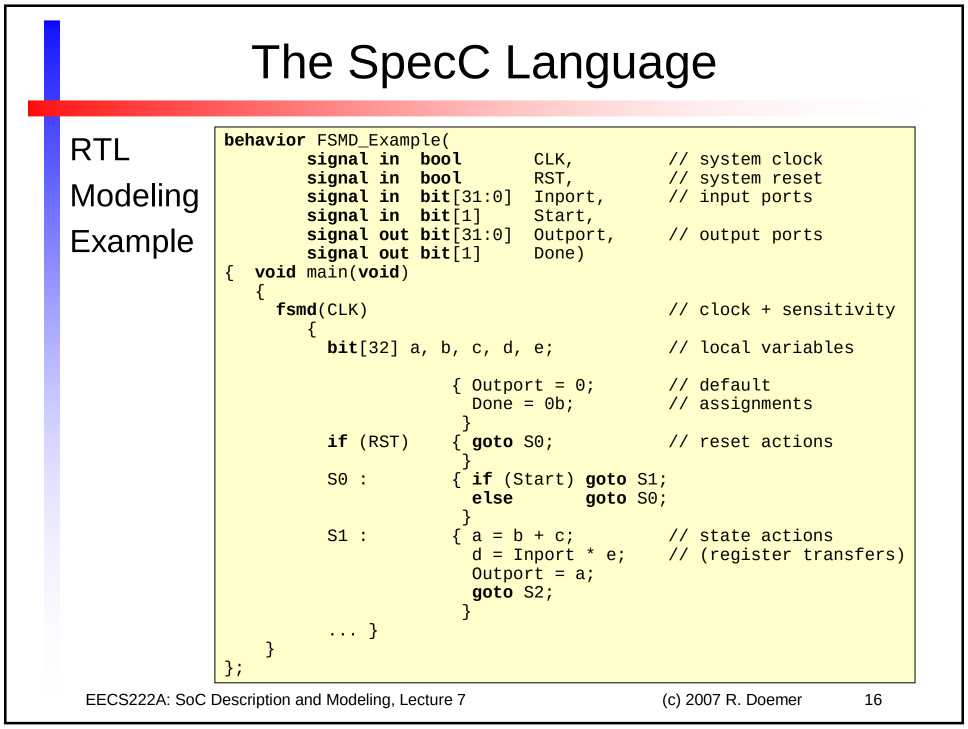
  - Implemented as buffered variables with associated event

```
signal int x;     ⇔  buffered int x_v; event x_e;
x = 55;           ⇔  x_v = 55; notify x_e;
y = x + 2;        ⇔  y = x_v + 2;
wait x;           ⇔  wait x_e;
notify x;         ⇔  notify x_e;
```



The SpecC Language

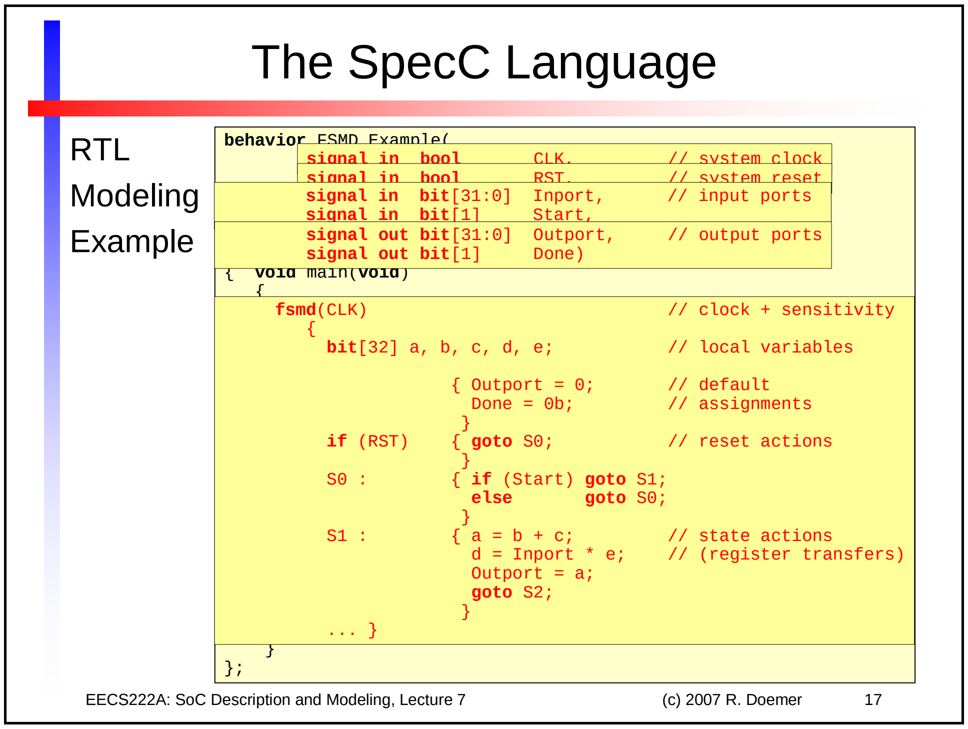
- RTL control flow
  - **`fsmd`** construct
    - Similar to **`fsm`** construct, but specifically for RTL
    - Explicit states and state transitions
    - State actions represent well-defined register transfers
      - limited to conditional/unconditional assignments and function calls
      - general loops, exceptions, synchronization, timing are not allowed
    - Explicit clock specifier
      - event list (external clock)
      - time delay (internal clock)
    - Explicit sensitivity list
      - needed for Mealy machine support
    - Explicit reset state
      - synchronous reset
      - asynchronous reset



The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
        {
          bit[32] a, b, c, d, e;           // local variables

                        { Outport = 0;     // default
                          Done = 0b;       // assignments
                         }
          if (RST)      { goto S0;         // reset actions
                         }
          S0 :          { if (Start) goto S1;
                          else       goto S0;
                         }
          S1 :          { a = b + c;       // state actions
                          d = Inport * e;  // (register transfers)
                          Outport = a;
                          goto S2;
                         }
          ... }
    }
};
```

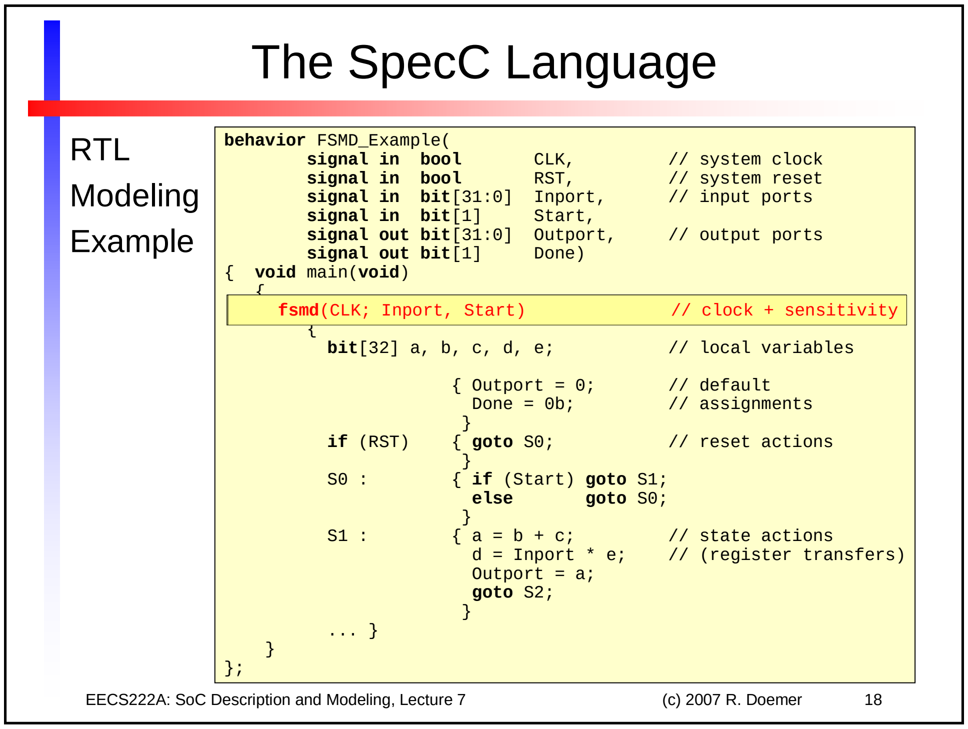


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
        {
          bit[32] a, b, c, d, e;           // local variables

                      { Outport = 0;       // default
                        Done = 0b;         // assignments
                       }
          if (RST)    { goto S0;           // reset actions
                       }
          S0 :        { if (Start) goto S1;
                        else       goto S0;
                       }
          S1 :        { a = b + c;         // state actions
                        d = Inport * e;    // (register transfers)
                        Outport = a;
                        goto S2;
                       }
          ... }
     }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 17

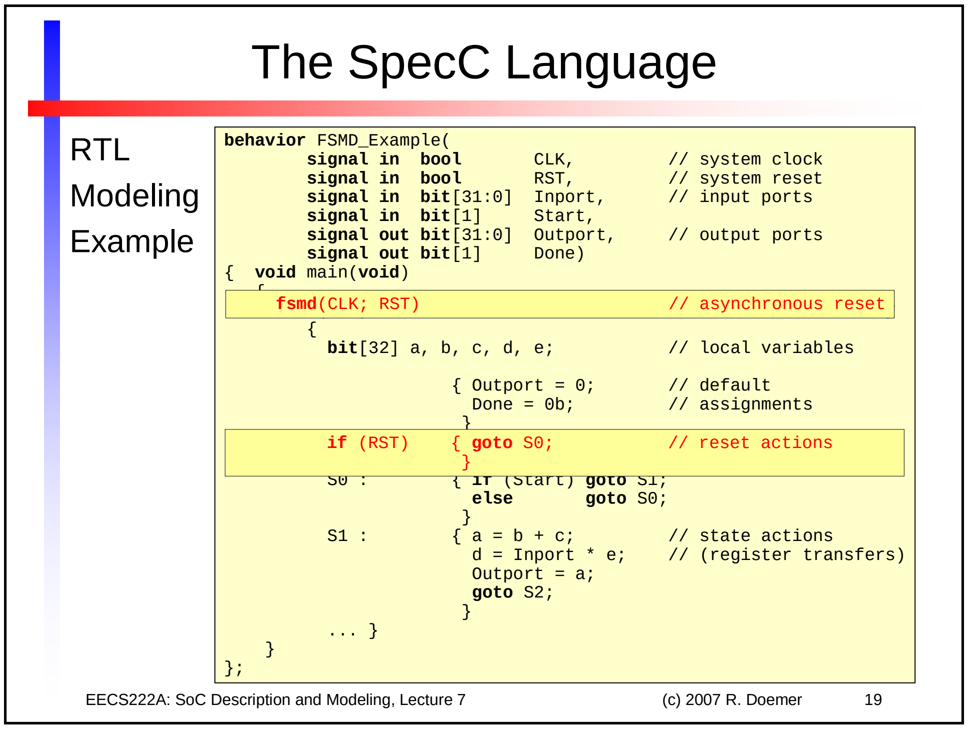


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK; Inport, Start)              // clock + sensitivity
        {
          bit[32] a, b, c, d, e;           // local variables

                      { Outport = 0;       // default
                        Done = 0b;         // assignments
                       }
          if (RST)    { goto S0;           // reset actions
                       }
          S0 :        { if (Start) goto S1;
                        else       goto S0;
                       }
          S1 :        { a = b + c;         // state actions
                        d = Inport * e;    // (register transfers)
                        Outport = a;
                        goto S2;
                       }
          ... }
     }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 18

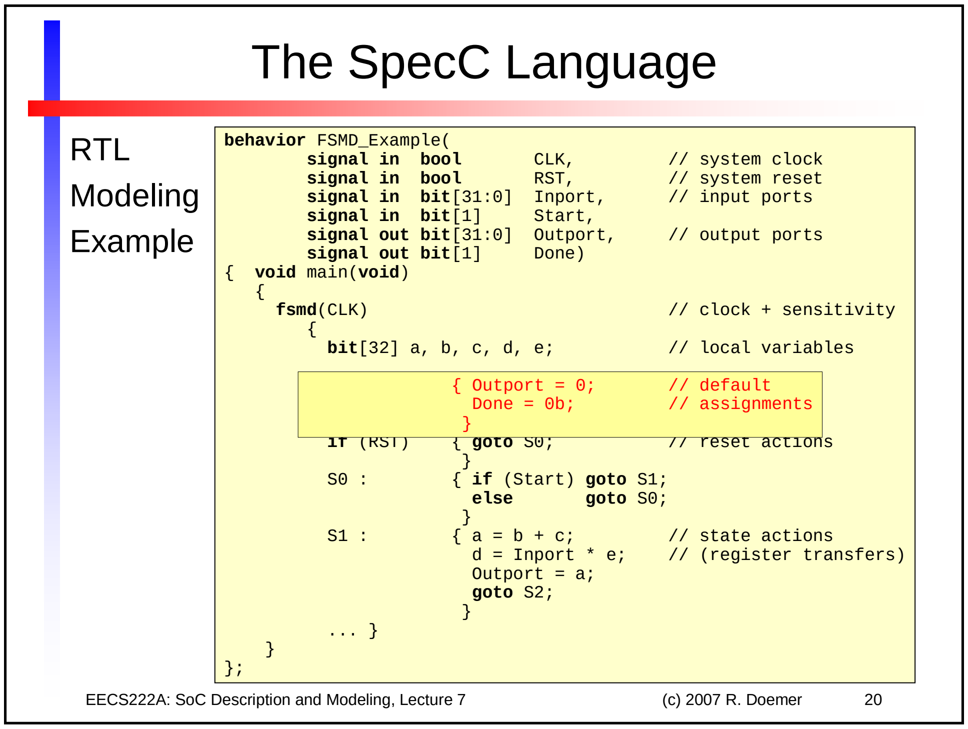


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,          // system clock
        signal in  bool       RST,          // system reset
        signal in  bit[31:0]  Inport,       // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,      // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK; RST)                         // asynchronous reset
        {
          bit[32] a, b, c, d, e;            // local variables

                      { Outport = 0;        // default
                        Done = 0b;          // assignments
                       }
          if (RST)    { goto S0;            // reset actions
                       }
          S0 :        { if (Start) goto S1;
                        else       goto S0;
                       }
          S1 :        { a = b + c;          // state actions
                        d = Inport * e;     // (register transfers)
                        Outport = a;
                        goto S2;
                       }
          ... }
    }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 19

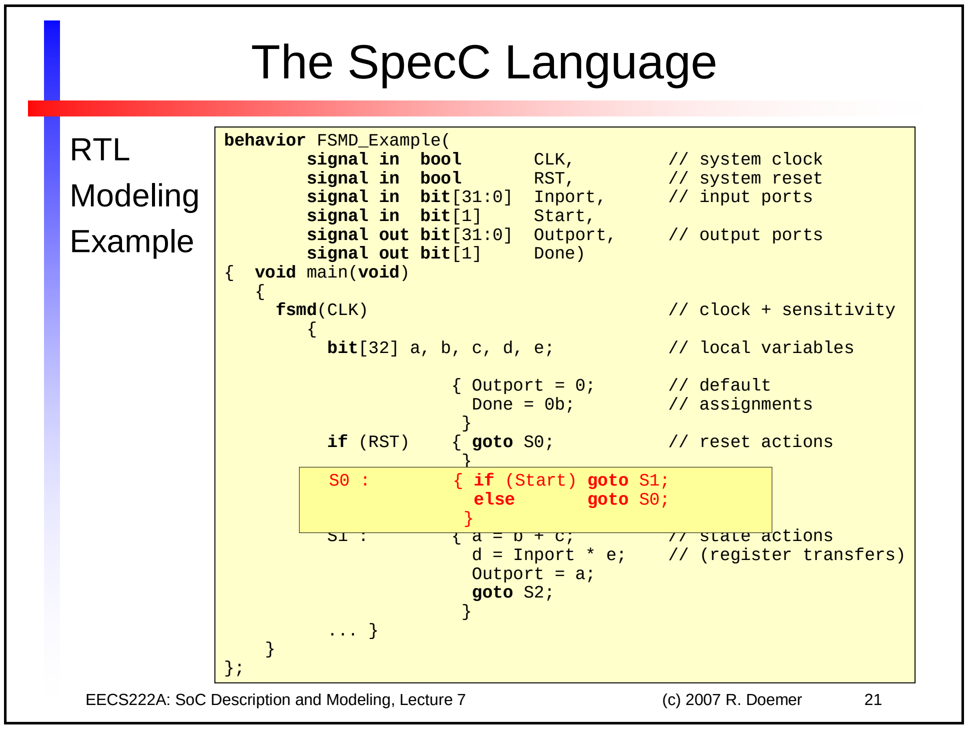


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,          // system clock
        signal in  bool       RST,          // system reset
        signal in  bit[31:0]  Inport,       // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,      // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                              // clock + sensitivity
        {
          bit[32] a, b, c, d, e;            // local variables

                      { Outport = 0;        // default
                        Done = 0b;          // assignments
                       }
          if (RST)    { goto S0;            // reset actions
                       }
          S0 :        { if (Start) goto S1;
                        else       goto S0;
                       }
          S1 :        { a = b + c;          // state actions
                        d = Inport * e;     // (register transfers)
                        Outport = a;
                        goto S2;
                       }
          ... }
    }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 20

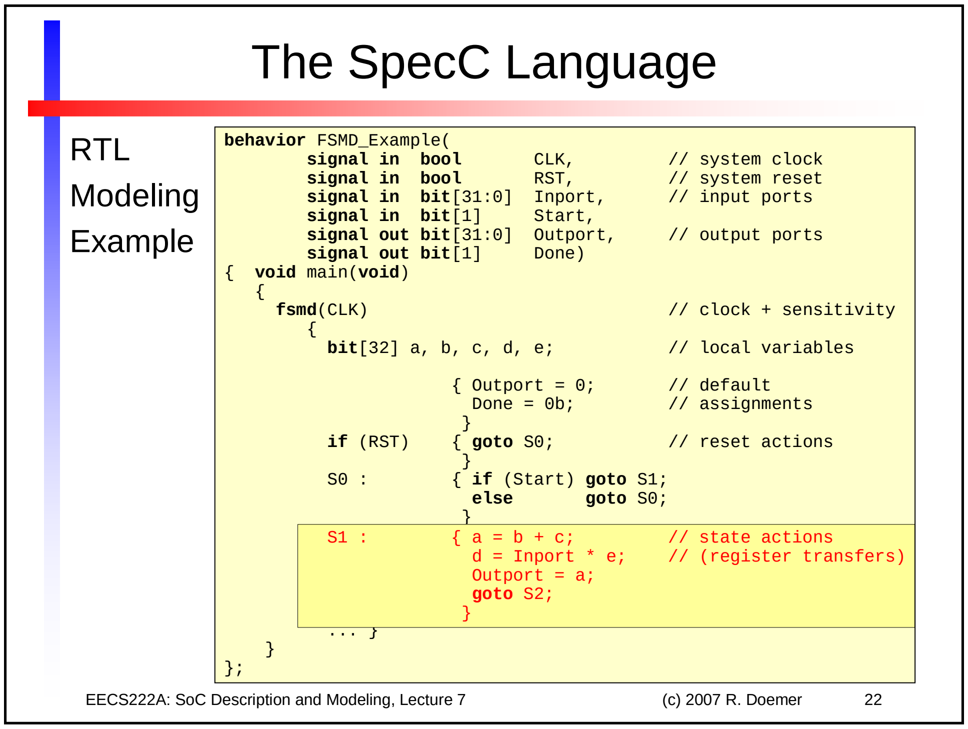


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
        {
          bit[32] a, b, c, d, e;           // local variables

                        { Outport = 0;     // default
                          Done = 0b;       // assignments
                         }
          if (RST)      { goto S0;         // reset actions
                         }
          S0 :          { if (Start) goto S1;
                          else       goto S0;
                         }
          S1 :          { a = b + c;       // state actions
                          d = Inport * e;  // (register transfers)
                          Outport = a;
                          goto S2;
                         }
          ... }
     }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 21

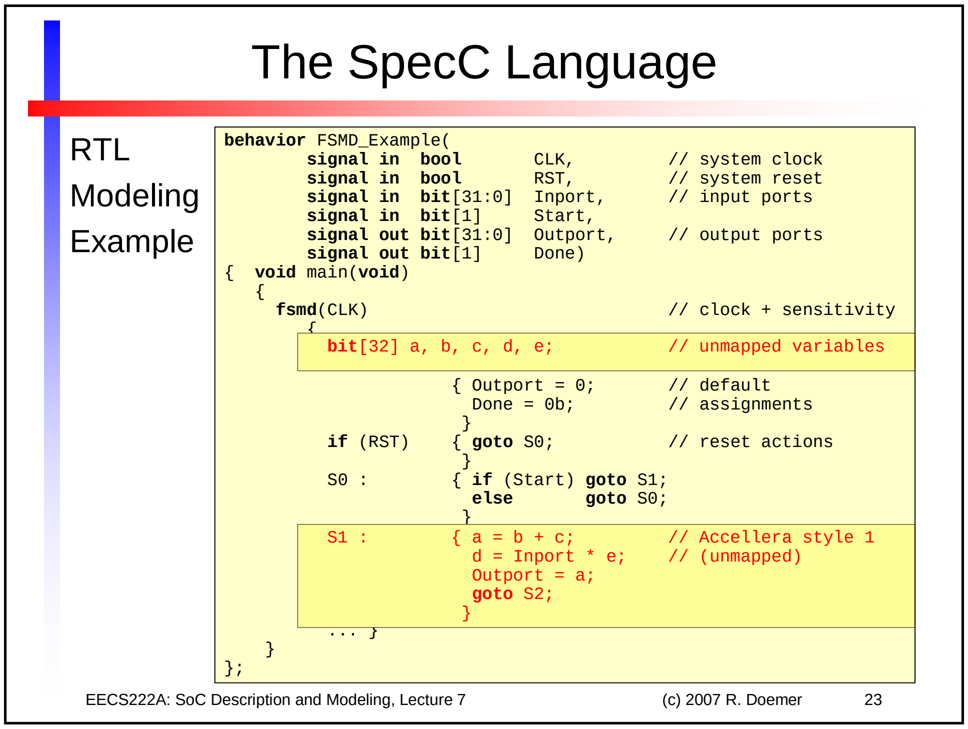


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
        {
          bit[32] a, b, c, d, e;           // local variables

                        { Outport = 0;     // default
                          Done = 0b;       // assignments
                         }
          if (RST)      { goto S0;         // reset actions
                         }
          S0 :          { if (Start) goto S1;
                          else       goto S0;
                         }
          S1 :          { a = b + c;       // state actions
                          d = Inport * e;  // (register transfers)
                          Outport = a;
                          goto S2;
                         }
          ... }
     }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 22

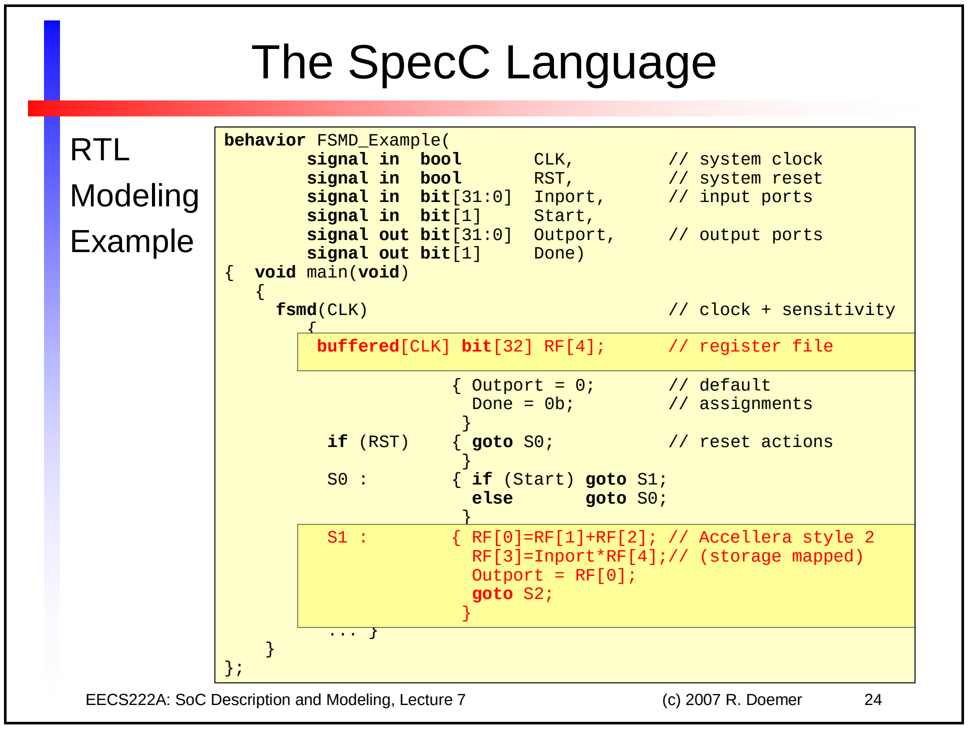


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
        {
          bit[32] a, b, c, d, e;           // unmapped variables

                      { Outport = 0;       // default
                        Done = 0b;         // assignments
                       }
          if (RST)    { goto S0;           // reset actions
                       }
          S0 :        { if (Start) goto S1;
                        else       goto S0;
                       }
          S1 :        { a = b + c;         // Accellera style 1
                        d = Inport * e;    // (unmapped)
                        Outport = a;
                        goto S2;
                       }
          ... }
    }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 23

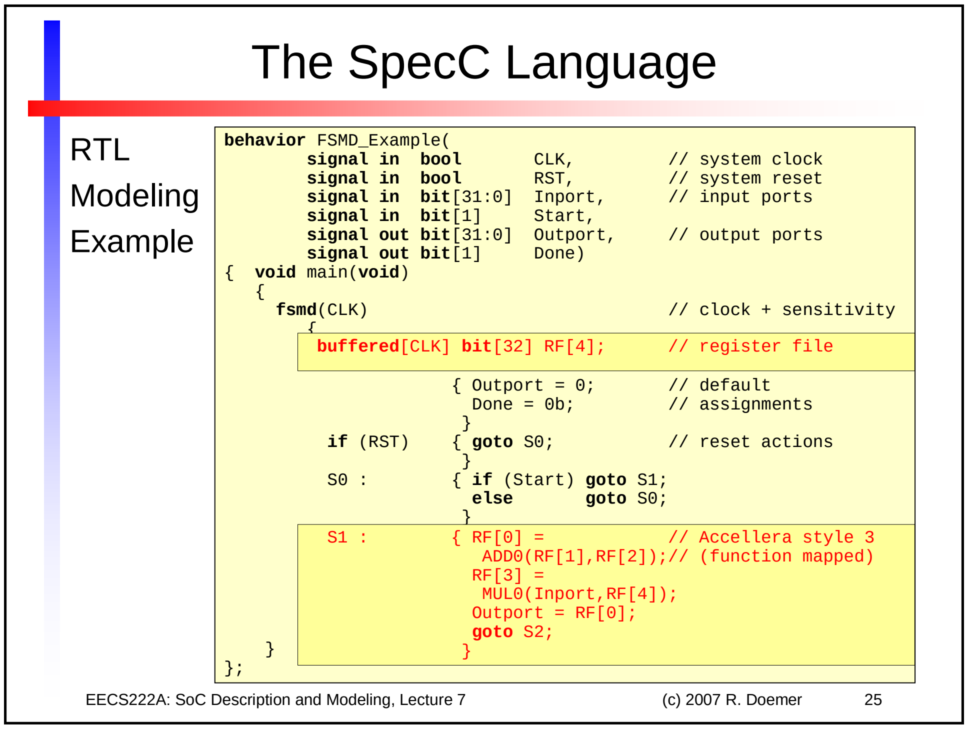


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
        {
         buffered[CLK] bit[32] RF[4];      // register file

                      { Outport = 0;       // default
                        Done = 0b;         // assignments
                       }
          if (RST)    { goto S0;           // reset actions
                       }
          S0 :        { if (Start) goto S1;
                        else       goto S0;
                       }
          S1 :        { RF[0]=RF[1]+RF[2]; // Accellera style 2
                        RF[3]=Inport*RF[4];// (storage mapped)
                        Outport = RF[0];
                        goto S2;
                       }
          ... }
    }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 24

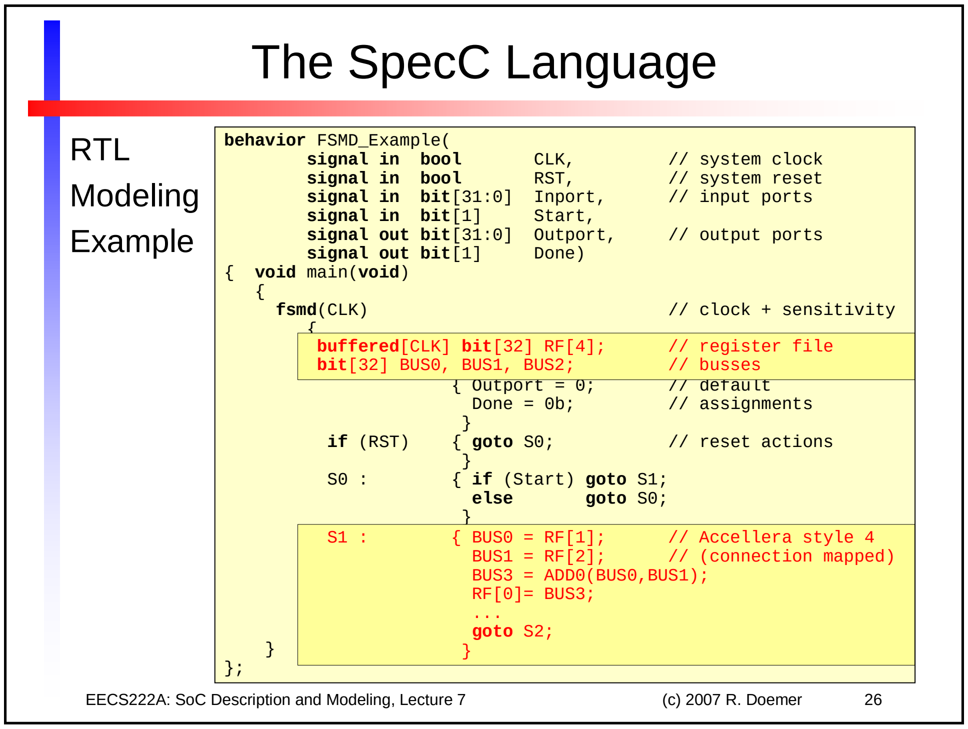


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
        {
        buffered[CLK] bit[32] RF[4];       // register file

                       { Outport = 0;      // default
                         Done = 0b;        // assignments
                        }
         if (RST)      { goto S0;          // reset actions
                        }
         S0 :          { if (Start) goto S1;
                         else       goto S0;
                        }
         S1 :          { RF[0] =           // Accellera style 3
                          ADD0(RF[1],RF[2]);// (function mapped)
                         RF[3] =
                          MUL0(Inport,RF[4]);
                         Outport = RF[0];
                         goto S2;
     }                  }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 25

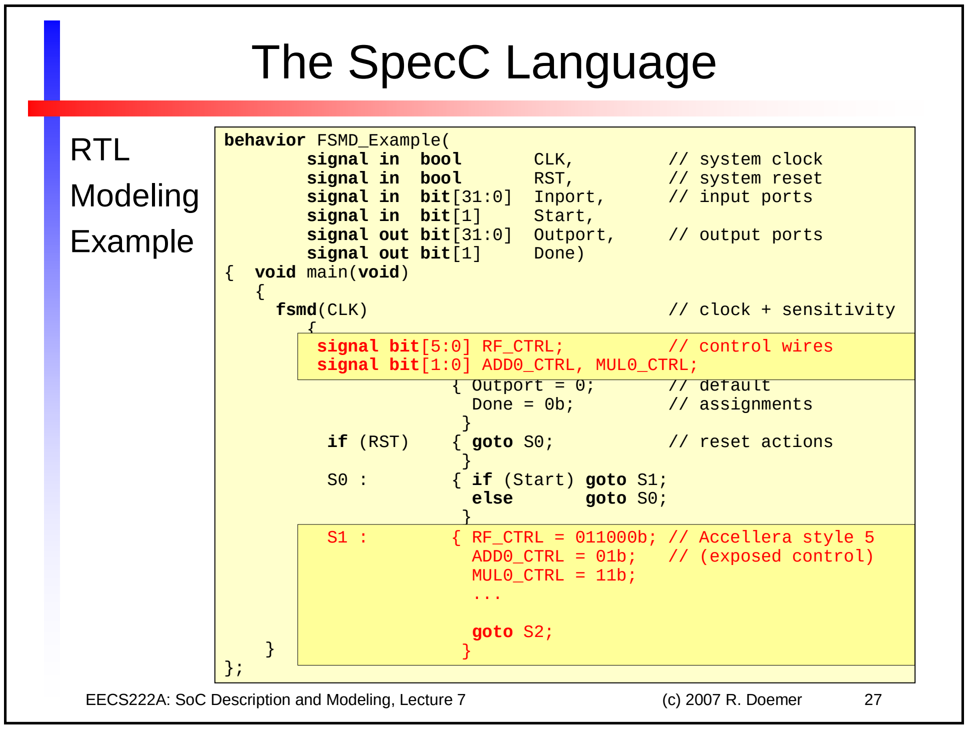


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
        {
        buffered[CLK] bit[32] RF[4];       // register file
        bit[32] BUS0, BUS1, BUS2;          // busses
                       { Outport = 0;      // default
                         Done = 0b;        // assignments
                        }
         if (RST)      { goto S0;          // reset actions
                        }
         S0 :          { if (Start) goto S1;
                         else       goto S0;
                        }
         S1 :          { BUS0 = RF[1];     // Accellera style 4
                         BUS1 = RF[2];     // (connection mapped)
                         BUS3 = ADD0(BUS0,BUS1);
                         RF[0]= BUS3;
                          ...
                         goto S2;
     }                  }
};
```

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 26

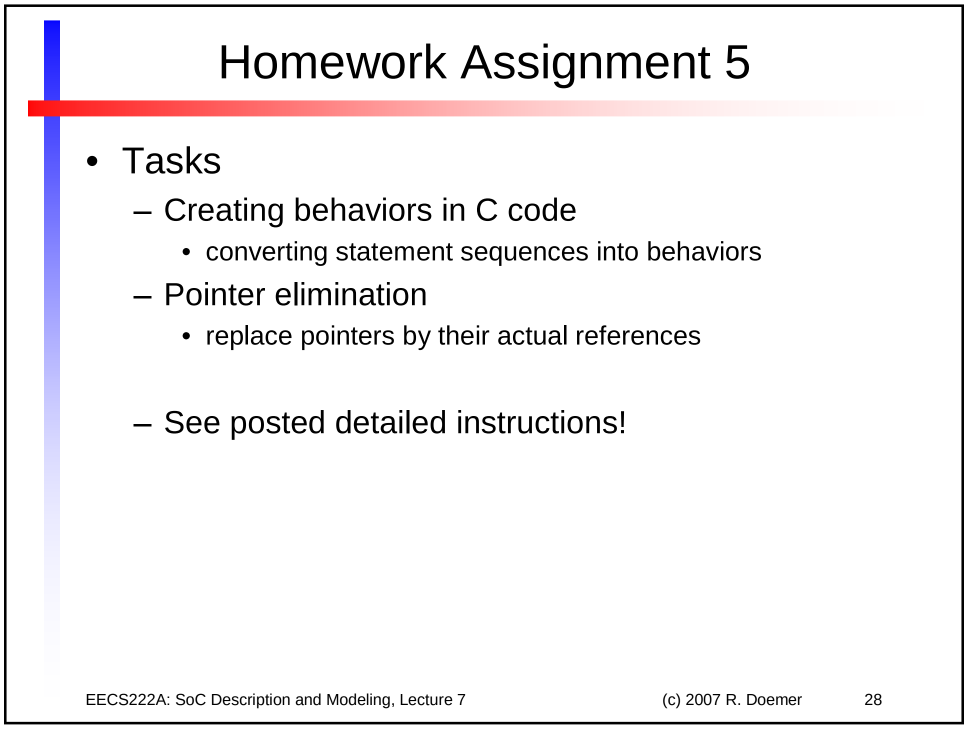


The SpecC Language

RTL Modeling Example

```
behavior FSMD_Example(
        signal in  bool       CLK,         // system clock
        signal in  bool       RST,         // system reset
        signal in  bit[31:0]  Inport,      // input ports
        signal in  bit[1]     Start,
        signal out bit[31:0]  Outport,     // output ports
        signal out bit[1]     Done)
{  void main(void)
   {
     fsmd(CLK)                             // clock + sensitivity
          {
       signal bit[5:0] RF_CTRL;            // control wires
       signal bit[1:0] ADD0_CTRL, MUL0_CTRL;
                      { Outport = 0;       // default
                        Done = 0b;         // assignments
                       }
          if (RST)    { goto S0;           // reset actions
                       }
          S0 :        { if (Start) goto S1;
                        else       goto S0;
                       }
          S1 :        { RF_CTRL = 011000b; // Accellera style 5
                        ADD0_CTRL = 01b;   // (exposed control)
                        MUL0_CTRL = 11b;
                        ...

                        goto S2;
     }                 }
};
```

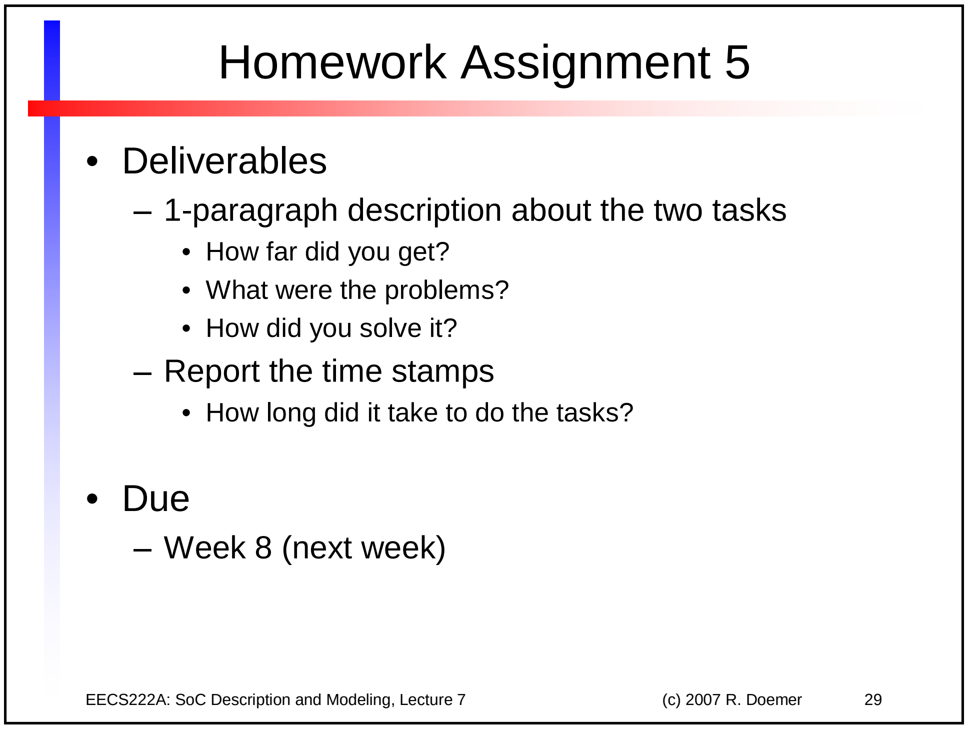
EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 27



Homework Assignment 5

- Tasks
  - Creating behaviors in C code
    - converting statement sequences into behaviors
  - Pointer elimination
    - replace pointers by their actual references
  - See posted detailed instructions!

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 28



Homework Assignment 5

- Deliverables
  - 1-paragraph description about the two tasks
    - How far did you get?
    - What were the problems?
    - How did you solve it?
  - Report the time stamps
    - How long did it take to do the tasks?
- Due
  - Week 8 (next week)

EECS222A: SoC Description and Modeling, Lecture 7 (c) 2007 R. Doemer 29