Ingredients for Successful System Level Automation & Design Methodology

Hiren Patel
hiren@vt.edu

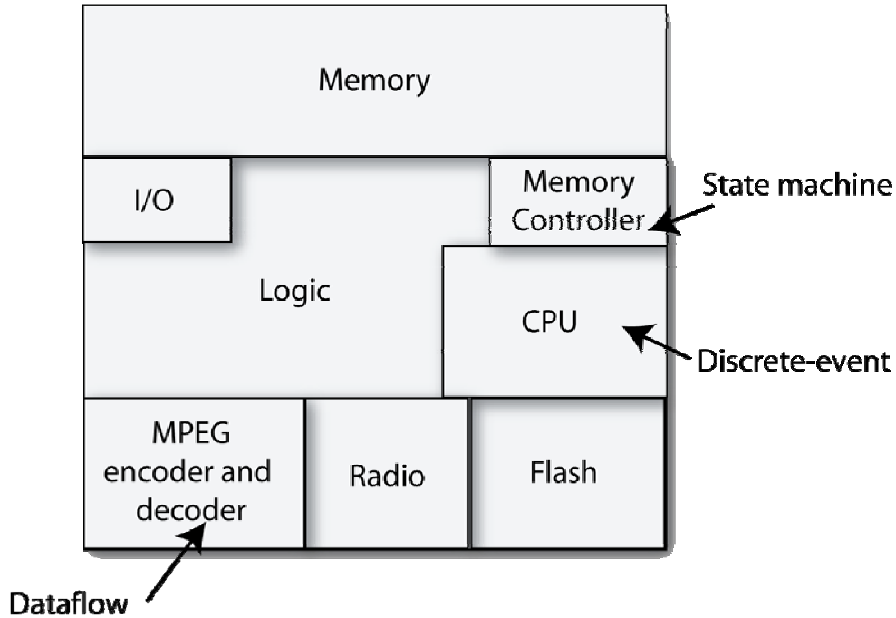
Formal Engineering Research with Models, Abstractions, and Transformations (FERMAT) Research Lab.
http://fermat.ece.vt.edu/

Mentor: Sandeep Shukla
shukla@vt.edu

Presentation for Chess seminar

Setting the Scene

- **Embedded system design**
  - **Increasingly complex and heterogeneous**
  - **Multi-functioned**
    - **Voice/data communication, music players, video players, and cameras**
  - **Using a variety of components with different functionalities**
    - **Microcontrollers, memories, and DSP cores**



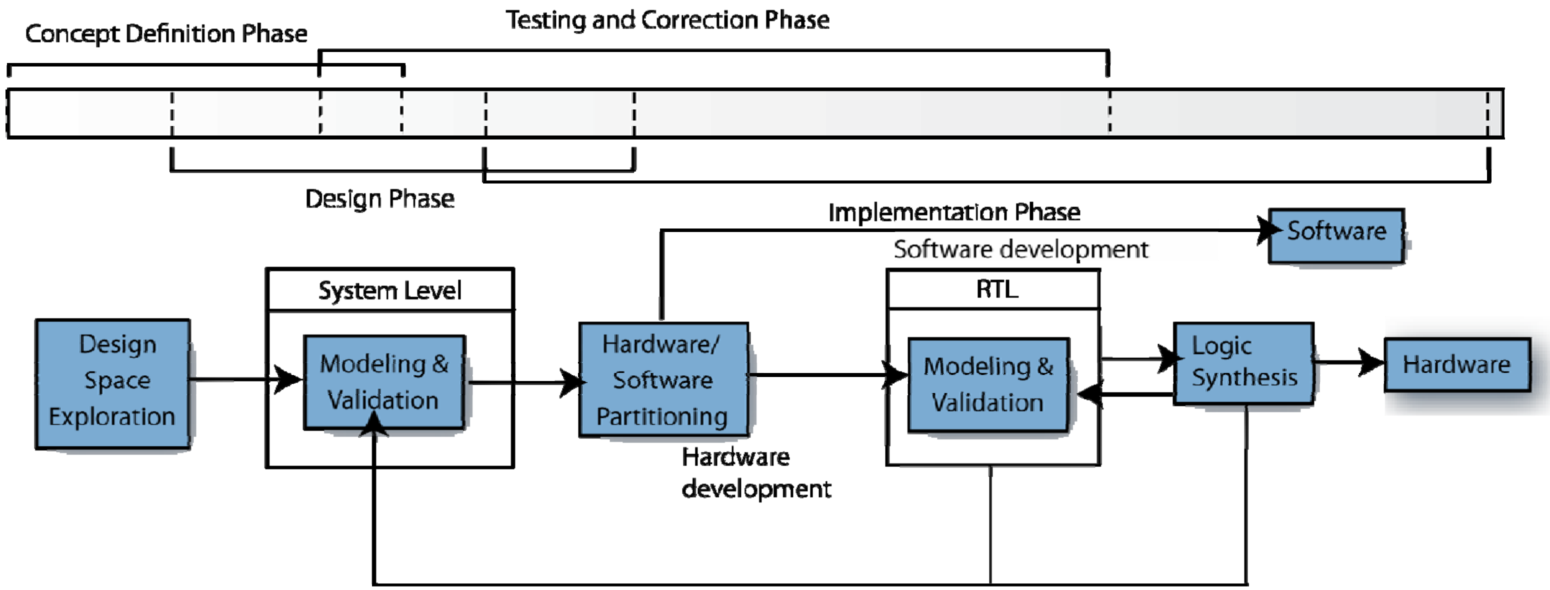
H. D. Patel and S. K. Shukla, *"Ingredients for Successful System Level Design Methodology and Automation"*, to appear in Springer, 2008.

Hiren D. Patel
Virginia Tech.

Setting the Scene

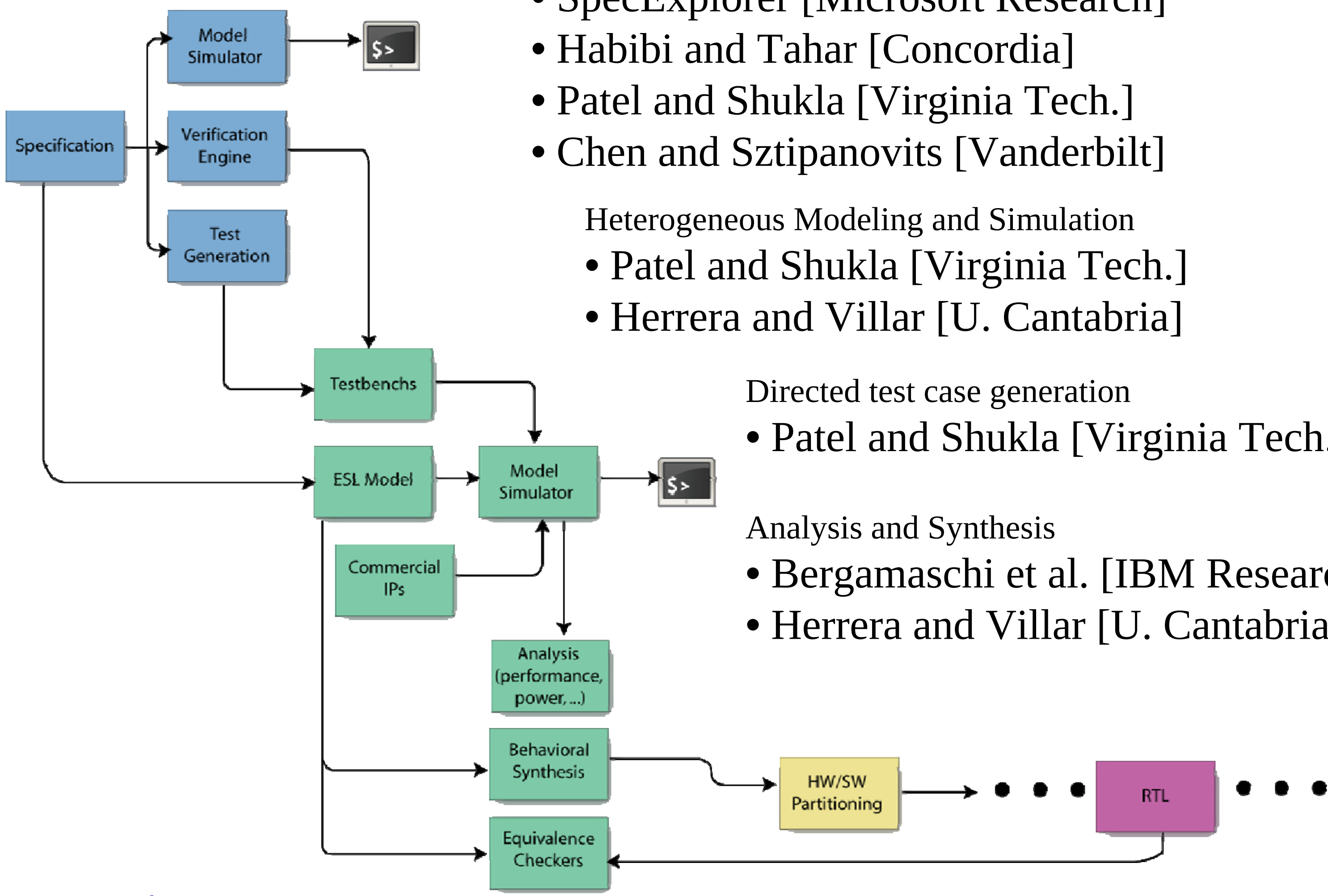
- Designers struggle with increasing requirements
    - Tools and methodologies do not scale
- Possible approach for mitigation:
    - Raise the abstraction layer to Electronic System Level (ESL)
- Electronic system level (ESL)
    - "a level above RTL (register transfer level) including hardware and software design" [ITRS 2004]
- But what is the next appropriate abstraction layer?
    - Lack of consensus in the EDA community
    - Proliferation of methodologies and tools

ESL Design Flow



- Exciting opportunities for research
  - Specification, analysis, IP Composition, validation, verification, behavioral synthesis, equivalence checking
- Required: recipe for ESL
  - Prescription [Martin et al., "ESL Design and Verification"]

Possible Recipe: By leveraging other works



Abstract State Machines and Verification

- SpecExplorer [Microsoft Research]
- Habibi and Tahar [Concordia]
- Patel and Shukla [Virginia Tech.]
- Chen and Sztipanovits [Vanderbilt]

Heterogeneous Modeling and Simulation

- Patel and Shukla [Virginia Tech.]
- Herrera and Villar [U. Cantabria]

Directed test case generation

- Patel and Shukla [Virginia Tech.]

Analysis and Synthesis

- Bergamaschi et al. [IBM Research]
- Herrera and Villar [U. Cantabria]

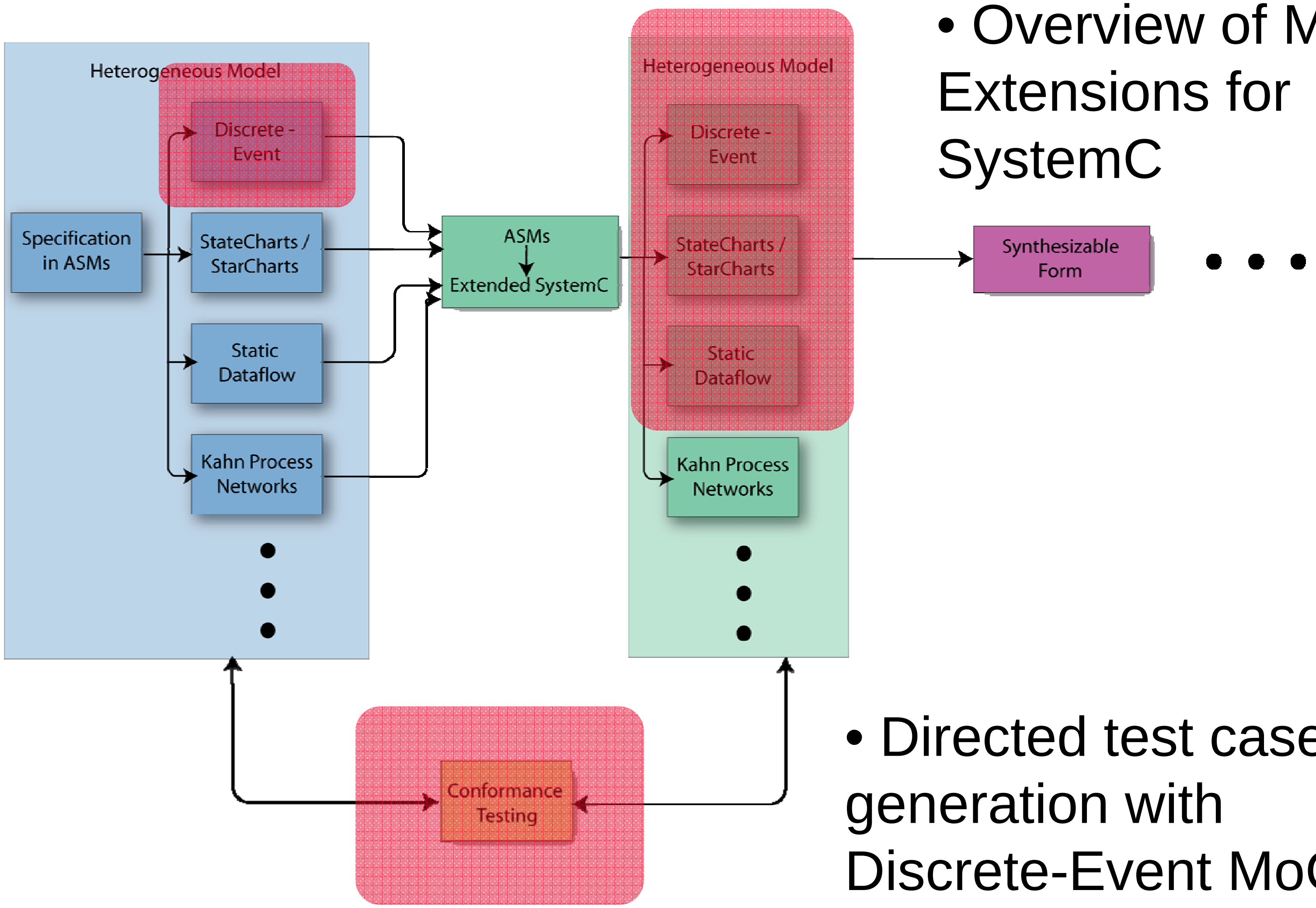
Focus

1. **Managing the prevalent need for heterogeneity in today's designs**
   - **Heterogeneous models of computation (MoCs)**

2. **Generating directed test cases for heterogeneous system level designs**
   - **Model-driven approach for validation**

3. **Integrating multiple tools and methodologies**
   - **Service-oriented architecture**

Today’s Outline



- Overview of MoC Extensions for SystemC

- Directed test case generation with Discrete-Event MoC

Hiren D. Patel
Virginia Tech.

Heterogeneity in SystemC

Why SystemC?

- Strong industrial backing
  - Large users community
- Language based on a library of C++ classes
- Recent IEEE standard
- Free modeling and simulation framework
  - Allows for experimentation
- Compiled with open-source compilers such as GCC

Heterogeneity in SystemC

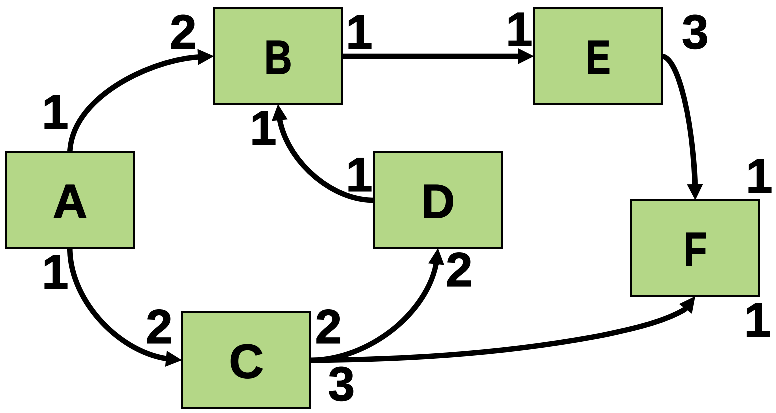
- **Introduce heterogeneous Models of Computation (MoCs)**
  - **Synchronous Data Flow (SDF)**
  - **Finite State Machine (FSM)**
  - **Communicating Sequential Processes (CSP)**
  - **Bluespec's Rule-based Paradigm**
- **Extend SystemC simulation framework to support simulation semantics of these MoCs**
- **Interoperable with Discrete-Event reference implementation of SystemC**

Related work

- **Ptolemy Project [UC Berkeley]**
  - **Heterogeneity for embedded software synthesis**
- **Metropolis Project [UC Berkeley]**
  - **Heterogeneity and platform-based methodology**
- **ForSyDE [KTH] & SML-Sys [Virginia Tech.]**
  - **Heterogeneous modeling and simulation**
- **HetSC [U. Cantabria]**
  - **Heterogeneous SystemC**

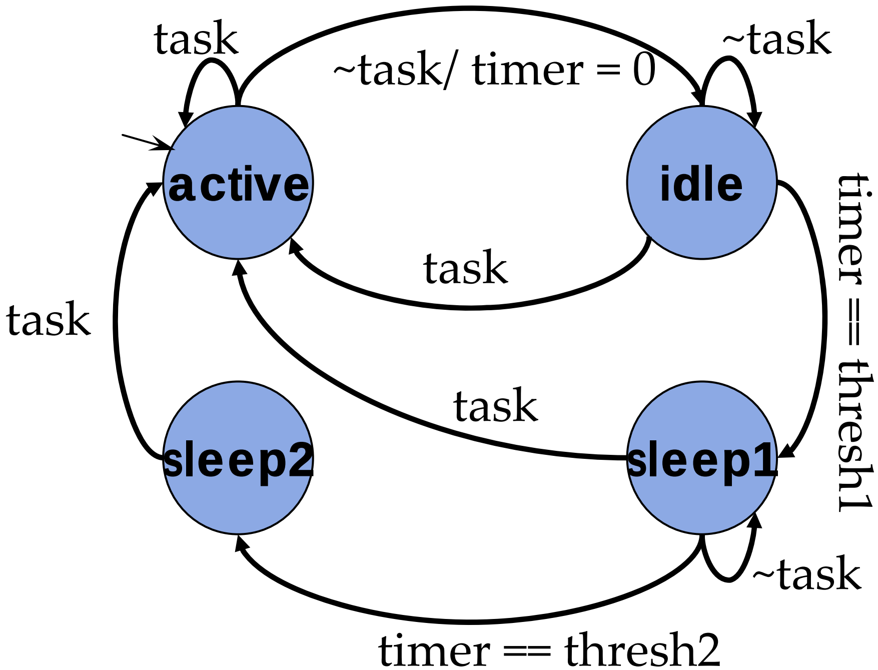
Heterogeneity in SystemC

SDF



Example from: H. D. Patel and S. K. Shukla, "*Towards A Heterogeneous Simulation Kernel for System Level Models: A SystemC Kernel for Synchronous Data Flow Models*", IEEE Transactions in Computer-Aided Design, Vol. 24, No. 8, August 2005

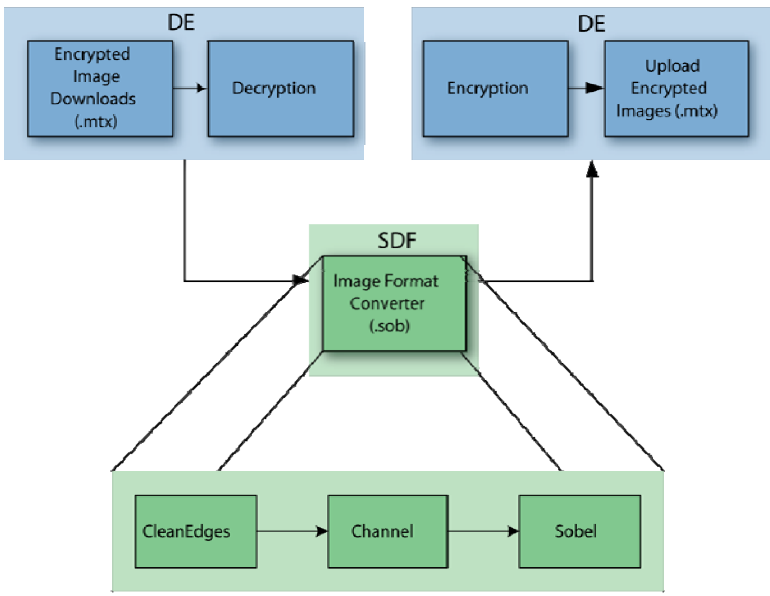
FSM



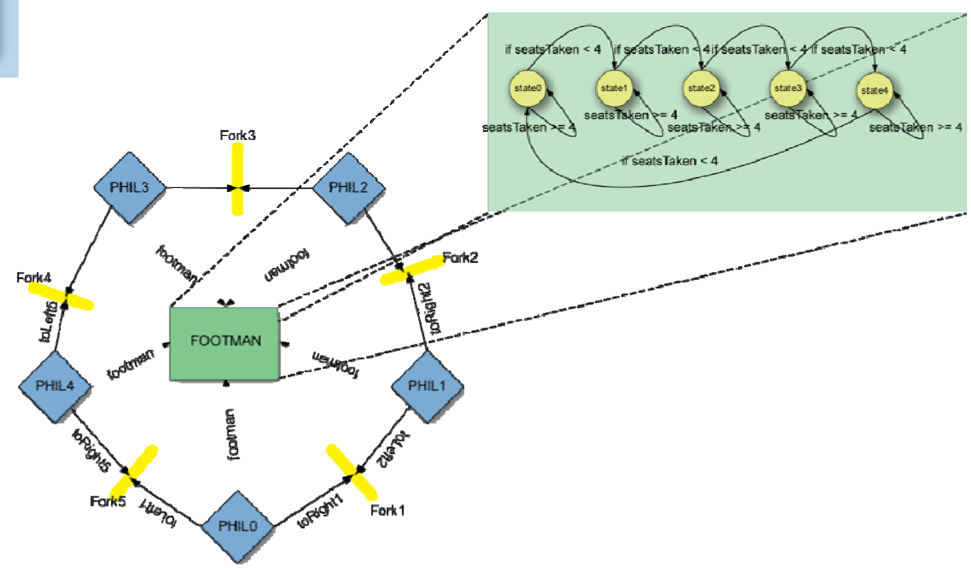
Example from: H. D. Patel and S. K. Shukla, "*Towards Behavioral Hierarchy Extensions for SystemC*", In Proceedings of Forum on Design and Specification Languages (FDL '05), Lausanne, Switzerland, September 2005

Some Examples

- **DE with SDF**

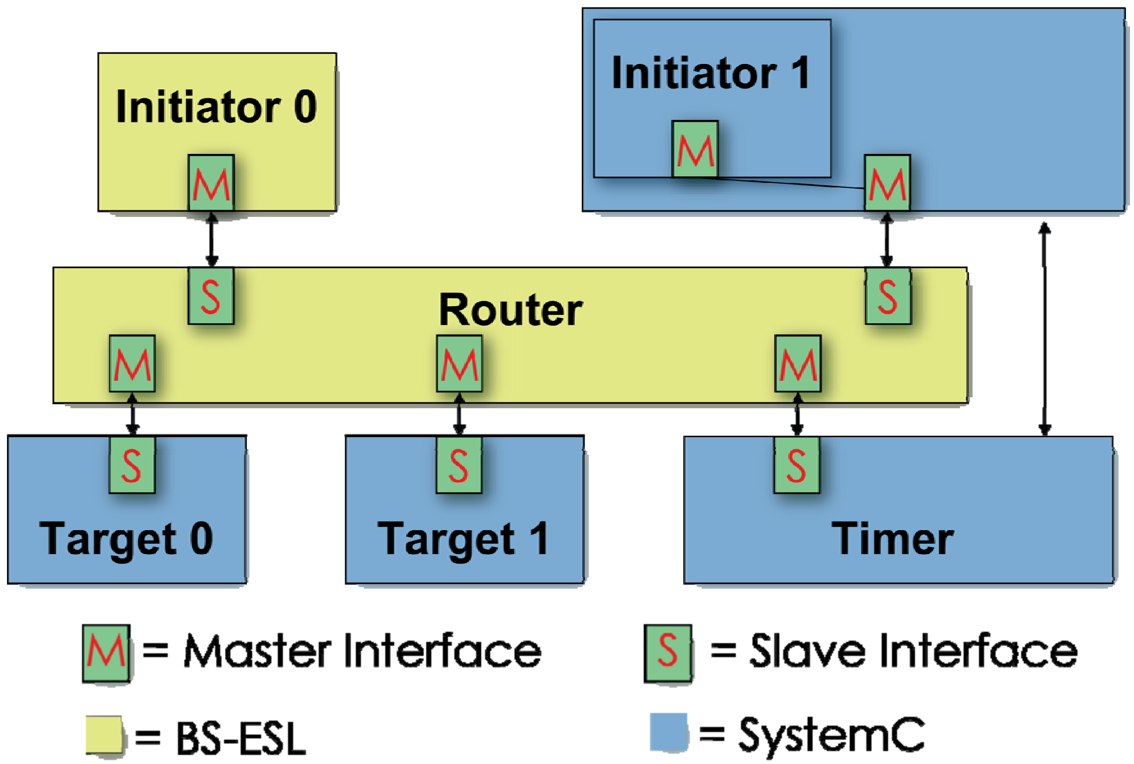


- **CSP with FSM**



Examples from: H. D. Patel and S. K. Shukla, *"SystemC Kernel Extensions for Heterogeneous System Modeling"*, Kluwer Academic Publishers, 2004.

Example: Rule-based MoC with SystemC

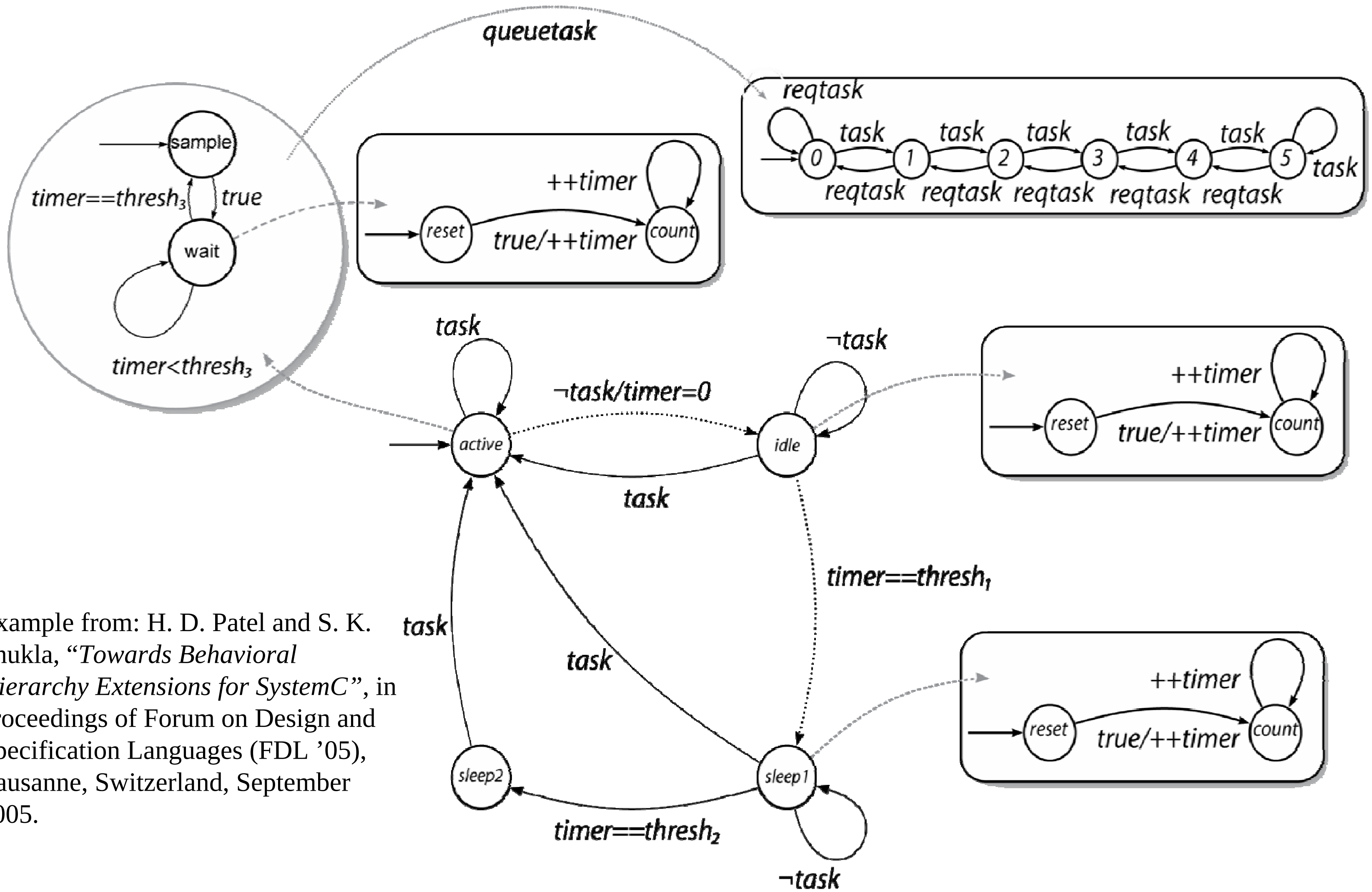


Example from:
1. H. D. Patel, S. K. Shukla, E. Mednick and R. S. Nikhil, *"A Rule-based Model of Computation for SystemC: Integrating SystemC and Bluespec for Co-design"*, in proceedings of Formal Methods and Models for Codesign, pp. 39-48, Napa Valley, California, 2006.
2. H. D. Patel and S. K. Shukla, *"On Co-simulating Multiple Abstraction Level System Level Models"*, to appear in IEEE Transactions in Computer-Aided Design, 2007.

Why Behavioral Hierarchy?

- Natural to decompose behaviors into small behaviors and compose/reuse small behaviors to realize larger behavior
- Traditional HDLs
  - Structural Hierarchy: components within other components connected via ports/signals
    - Encapsulation benefits during modeling and reuse
    - Hierarchy information flattened during simulation
- Behaviors realized by MoCs
- Simulation kernel responsible for simulating one level of hierarchy at a time
  - Refinements invoke another instance of the simulation kernel
  - Example for hierarchical FSM: Starchart semantics [Ptolemy II Group]

Example of Power State Model

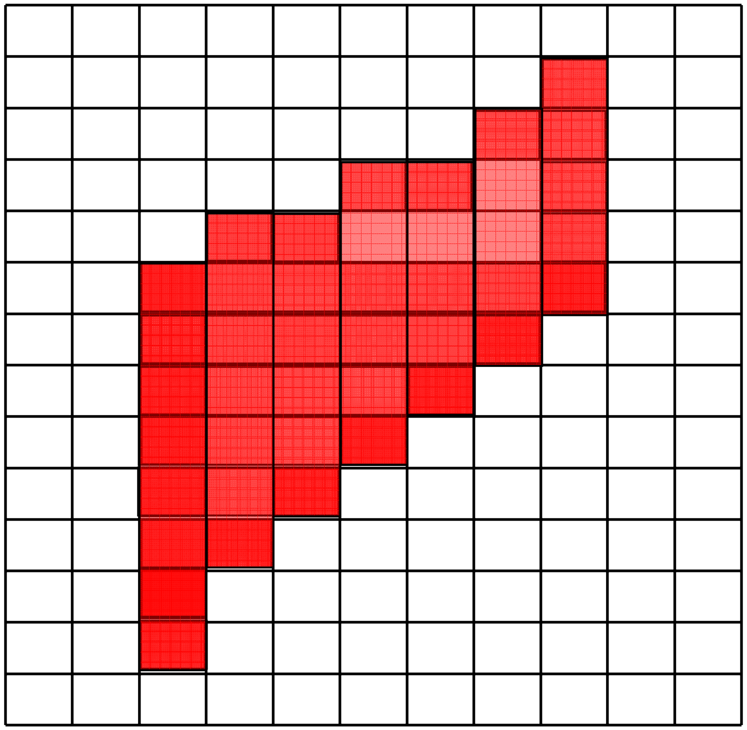


Example from: H. D. Patel and S. K. Shukla, "*Towards Behavioral Hierarchy Extensions for SystemC*", in proceedings of Forum on Design and Specification Languages (FDL '05), Lausanne, Switzerland, September 2005.

Heterogeneous Behavioral Hierarchy

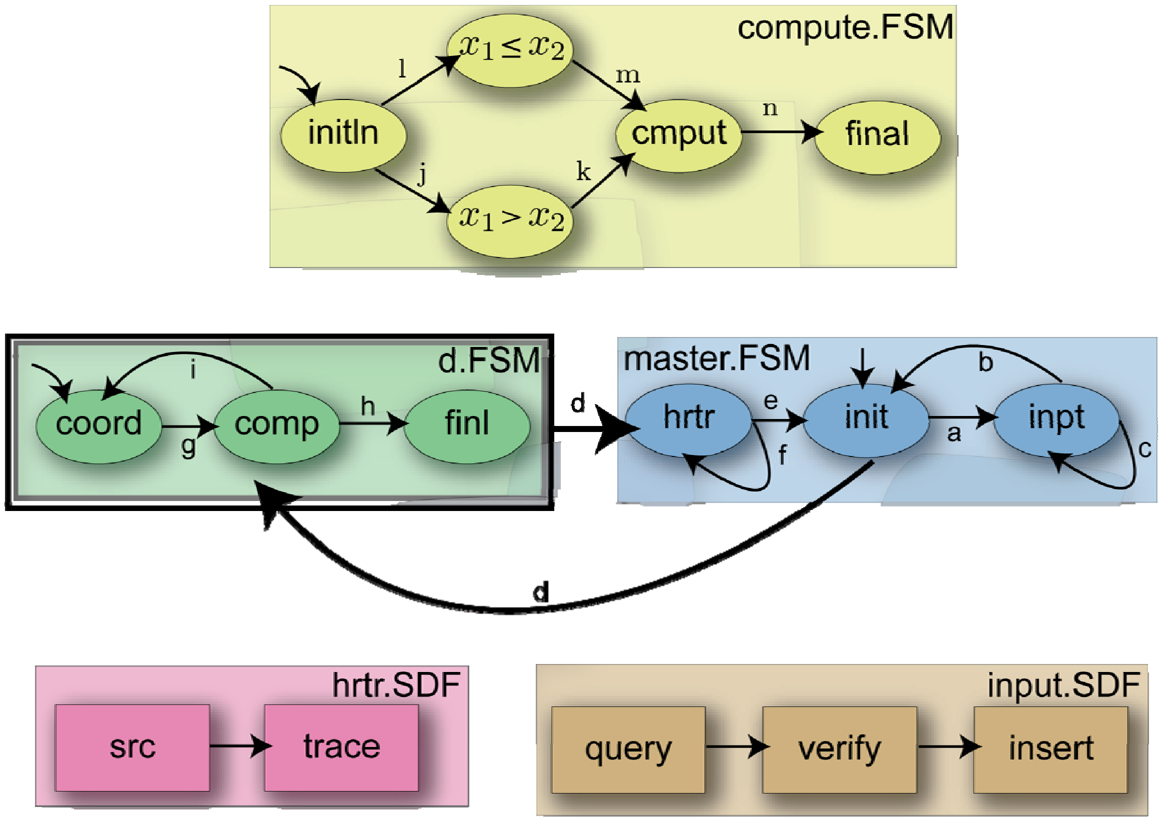
- Heterogeneous extensions of behavioral hierarchy
- Various behaviors realized by different MoCs are embedded in a variety of ways
  - Examples:
    - SDF embedded in an FSM state
    - FSM embedded in an SDF function block
- MoC-specific simulation kernel simulates each level of hierarchy according to MoC

Polygon Infill Processor Example



Example from: R. A. Bergamaschi, "*The Development of a High-Level Synthesis System for Concurrent VLSI Systems*," Ph.D. dissertation, University of Southampton, 1989.

Polygon Infill Processor

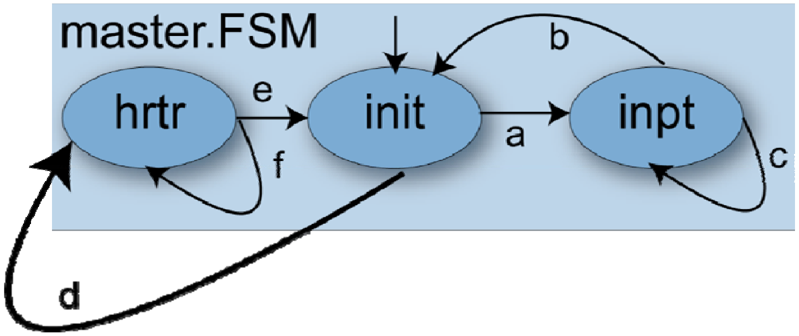


Example from:

1. H. D. Patel and S. K. Shukla, "*Heterogeneous Behavioral Hierarchy Extensions for SystemC*", in proceedings of IEEE Transactions in Computer-Aided Design, pp. 765-780, April 2007.
2. H. D. Patel and S. K. Shukla, "*Heterogeneous Behavioral Hierarchy for System Level Design*", in proceedings Design Automation and Test in Europe, pp. 565-570, Munich, Germany, March 2006.

Hiren D. Patel
Virginia Tech.

Modeling HFSM in SystemC



SystemC-H

```
SCH_FSM_STATE( init ) {
        SCH_FSM_STATE_CTOR( init )
        { /// … };

        SCH_FSM_ENTRY_ACTIONS( )
        {  /// … };

        SCH_FSM_EXIT_ACTIONS( )
        {  /// … };
};
```

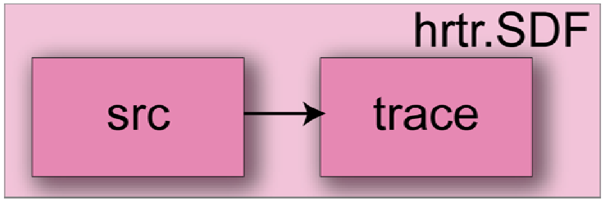

Hiren D. Patel
Virginia Tech.

Additional member functions for HFSM

- Adding more than one FSM / SDF refinements
  - add_fsm_refinement(), add_sdf_refinement()
- Three types of transitions
  - Run-to-complete: refinement must traverse from initial to final state before returning control
    - set_run_complete()
  - Preemptive: refinement is not executed
    - set_preemptive()
  - Reset: before changing state, resets the refinement of the destination state
    - set_reset()

Hiren D. Patel
Virginia Tech.

Modeling SDF



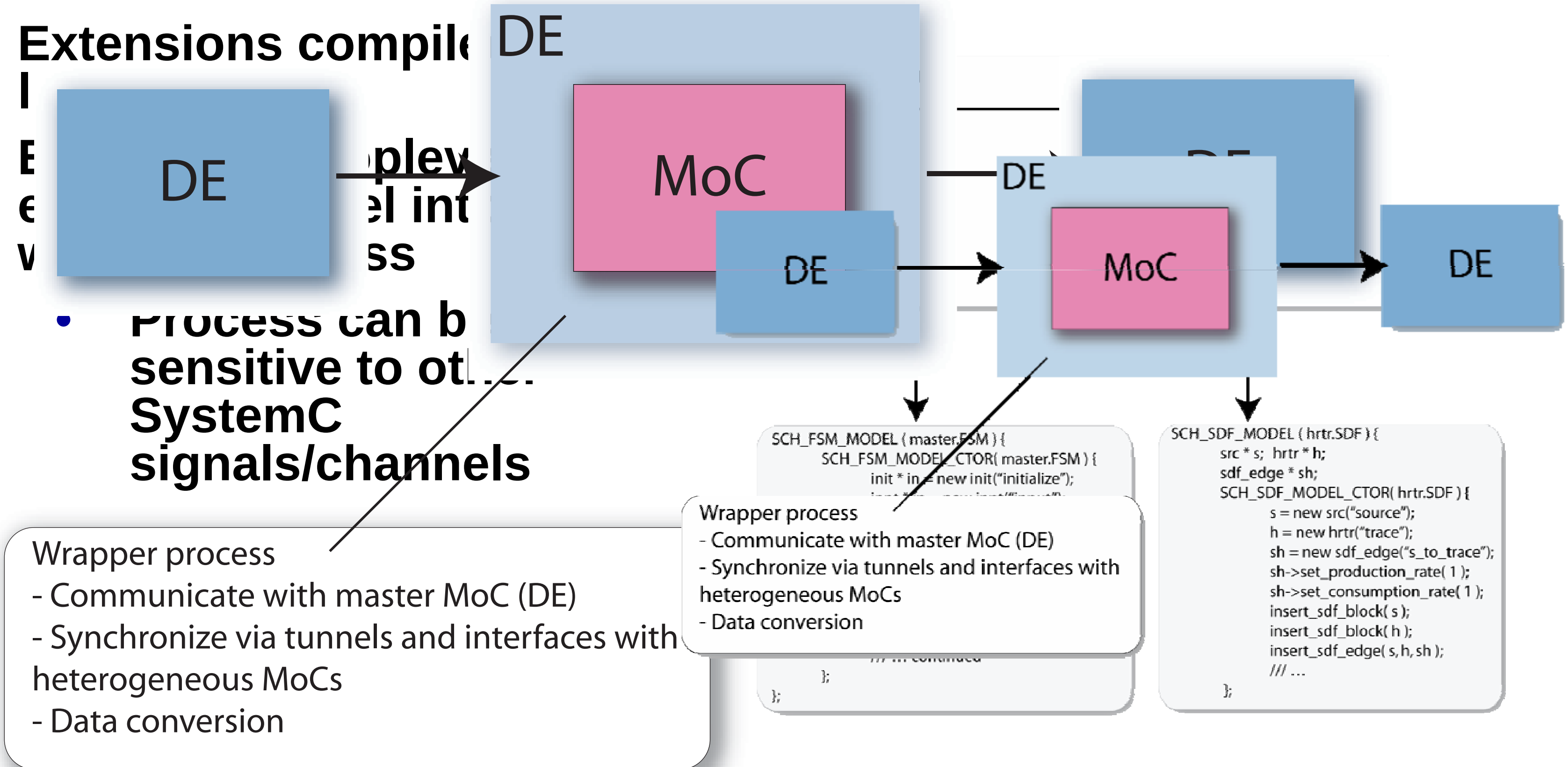
SystemC-H

```
SCH_SDF_MODEL ( hrtr.SDF ) {
    src * s;  hrtr * h;      sdf_edge * sh;
    SCH_SDF_MODEL_CTOR( hrtr.SDF ) {
        s = new src("source");
        h = new hrtr("trace");
        sh = new sdf_edge("s_to_trace");
        sh->set_production_rate( 1 );
        sh->set_consumption_rate( 1 );
        insert_sdf_block( s );
        insert_sdf_block( h );
        insert_sdf_edge( s, h, sh );
        /// …
    };
```


Hiren D. Patel
Virginia Tech.

Integrating Extensions

- Extensions compile
  l
- E plev
  e el int
  v ss
  - Process can b
    sensitive to ot
    SystemC
    signals/channels
    trigger()



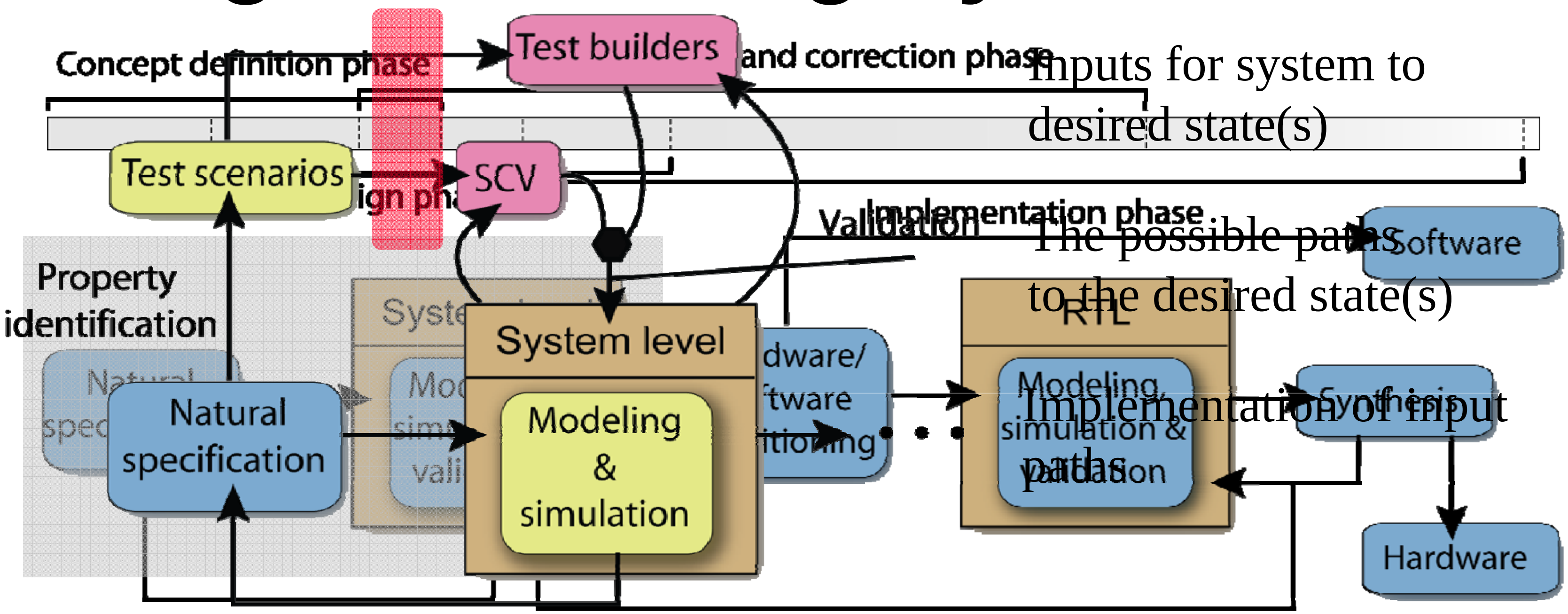
Hiren D. Patel
Virginia Tech.

Simulation Experiments

- Pure SDF models show ~65% speed up
- Pure CSP models show insignificant improvement
- Pure HFSM models showed insignificant improvement
- Heterogeneous behavioral hierarchy models showed up to 40% speed up

Model-driven Validation for Heterogeneous SystemC Designs

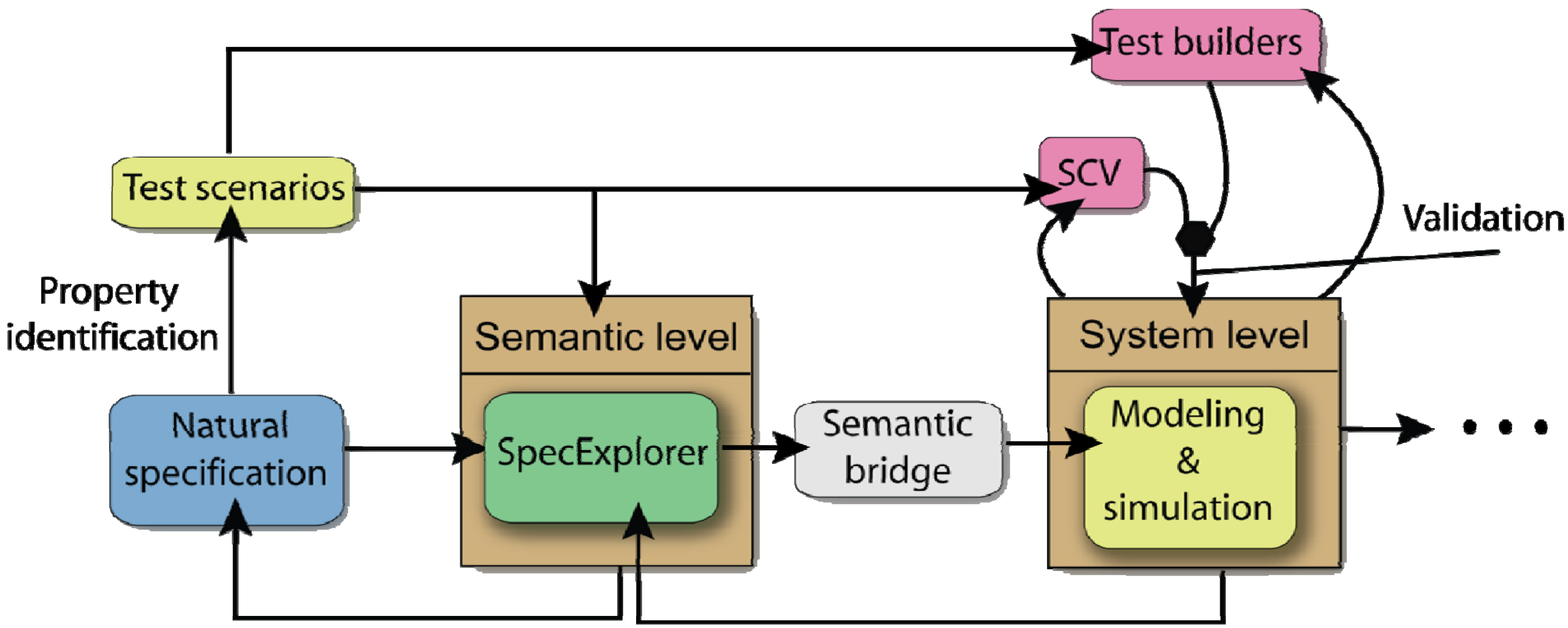
Design flow using SystemC



- Focus only on concept and design phase
- Functional validation
  - Test scenarios
  - Test suite implementation

Hiren D. Patel
Virginia Tech.

New design flow SystemC



- **Semantic level**
  - **AsmL specification of design under investigation**
  - **Simulation**

Hiren D. Patel
Virginia Tech.

Related work

- **Bruschi et al [Politechnicio Di Milano]**
- **Kroening & Sharygina [Carnegie Mellon]**
- **Habibi & Tahar [Concordia University]**
- **Koo & Mishra [University of Florida]**

Microsoft's SpecExplorer

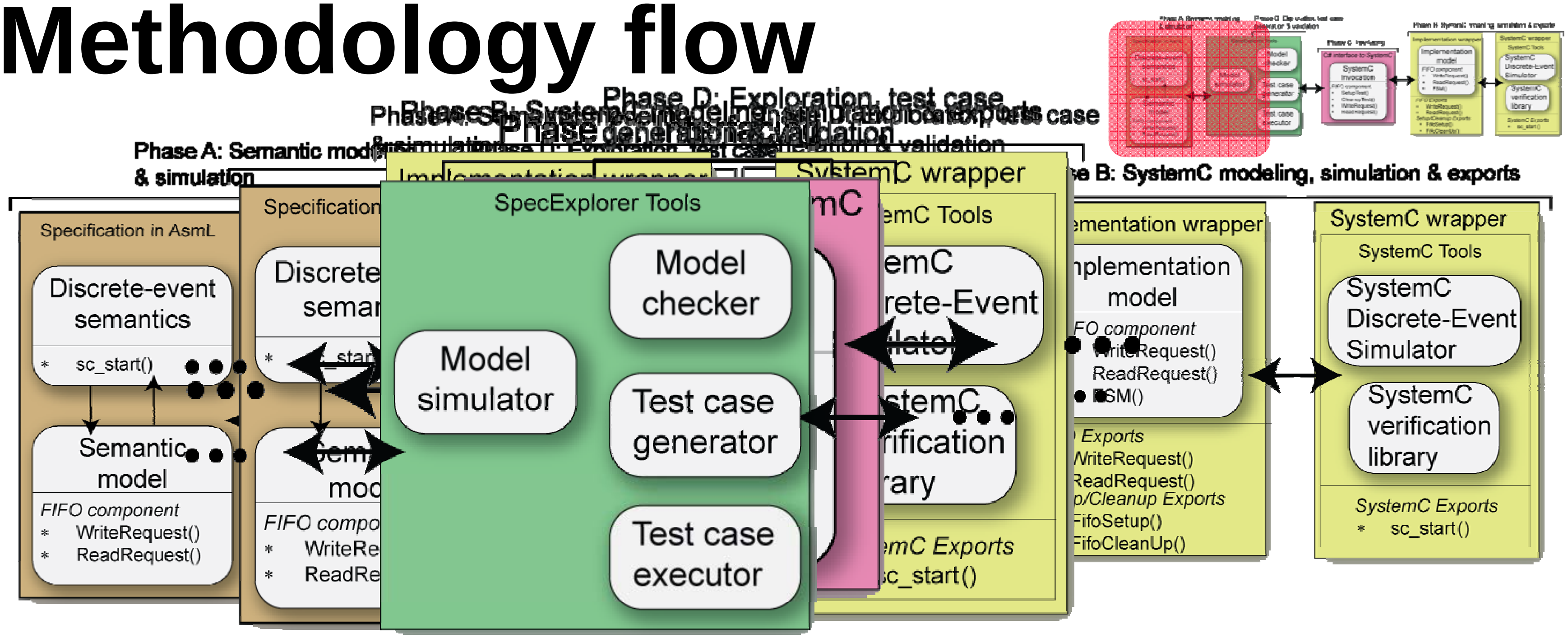
- Specification, exploration and test suite generation
- Specifications in AsmL
  - Based on ASM formalism
    - Sequential and parallel constructs
  - Serve proof obligations
- Exploration
  - States: grouping and filters
  - Actions: observable and controllable
- Test suite generation
  - Generate test paths from exploration

Model-driven validation for SystemC using SpecExplorer

- We need to provide:
  - Formal description of intended design similar to designing in SystemC
    - Precise formal semantics for the Discrete-event MoC
  - Method for SpecExplorer to drive implementation

Hiren D. Patel
Virginia Tech.

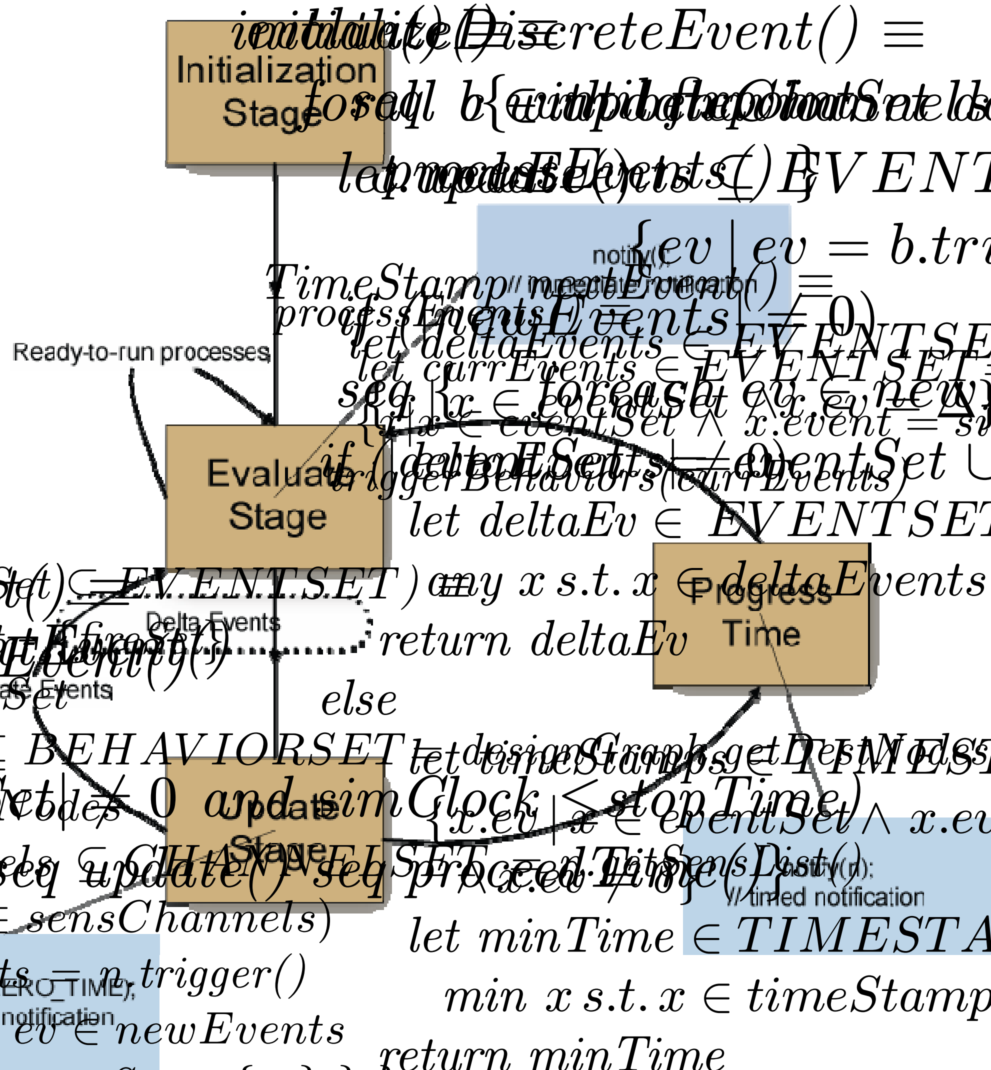
Methodology flow



- Discrete-event simulation semantics provided
- Semantic model
  - Design under investigation
  - Bindings from SystemC to C#
- Simulate using SpecExplorer's model simulator
  - Controllable and observable actions

Hiren D. Patel
Virginia Tech.

Discrete-event Semantics



$initialize() \equiv$ / $initializeDiscreteEvent() \equiv$

$forall\ b \in initialFunctionSet\ do$ / $seq\ \{\ update() ...$

$let\ modelEvents \subseteq EVENTSET = \{ev \mid ev = b.trigger()\}$

$seq\ \{\ foreach\ ev \in newEvents\ \ eventSet := eventSet \cup \{ev\}\ \}$

$TimeStamp\ nextEvent() \equiv$

$processEvents() \equiv$

$if\ (|deltaEvents| \neq 0)$

$let\ deltaEvents \in EVENTSET = \{x \mid x \in eventSet \wedge x.ev = \Delta\}$

$let\ currEvents \in EVENTSET = \{x \mid x \in eventSet \wedge x.event = simClock\}$

$triggerBehaviors(currEvents)$

$let\ deltaEv \in EVENTSET = any\ x\ s.t.\ x \in deltaEvents$

$return\ deltaEv$

$else$

$let\ timeStamps \in TIMESTAMP = \{x.ev \mid x \in eventSet \wedge x.ev > 0\}$

$let\ minTime \in TIMESTAMP = min\ x\ s.t.\ x \in timeStamps$

$return\ minTime$

$proceedTime() \equiv$ / $triggerDiscreteEvent() \equiv$

$triggerBehaviors(fireSet \subseteq EVENTSET) \equiv$

$eventSet := eventSet - fireSet$

$initializeDiscreteEvent()$

$seq\ \{\ forall\ f \in fireSet$

$let\ triggerNodes \subseteq BEHAVIORSET = designGraph.getDestNodes(f.channel)$

$while\ (|eventSet| \neq 0\ and\ simClock < stopTime)$

$forall\ n \in triggerNodes$

$evaluate()\ seq\ update()\ seq\ proceedTime()$

$let\ sensChannels \subseteq CHANNELSET = n.getSensList()$

$if\ (f.channel \in sensChannels)$

$let\ newEvents = n.trigger()$

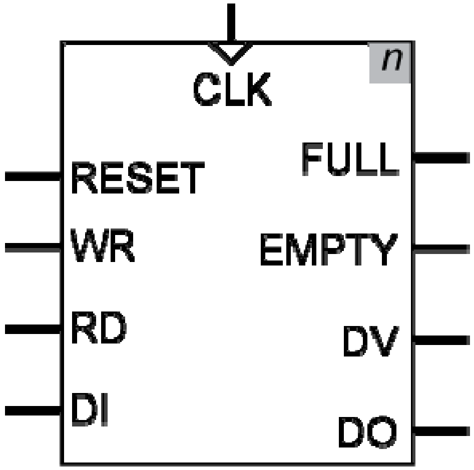
$seq\ \{\ foreach\ ev \in newEvents$

$eventSet := eventSet \cup \{ev\}\ \}\ \}$

Hiren D. Patel
Virginia Tech.

Modeling & simulation of simple FIFO example

- For a fixed size n
- WR is valid
  - DI in pushed
- RD is valid
  - DO has popped value
  - DV is valid
- FULL is high
  - Size n reached
- EMPTY is high
  - Size 0 is reached
- RESET empties FIFO



CLK: Clock input	FULL: Full FIFO
WR: Write request	EMPTY: Empty FIFO
RD: Read request	DV: Data valid
DI: Data input	DO: Data output

Example from: H. D. Patel and S. K. Shukla, "*Model-driven validation of SystemC designs*", in proceedings of Design Automation Conference, pp 29-34, 2007

Phase A: Semantic model

- FIFO FSM
  - INIT, REQUESTS, WAIT
- REQUESTS
  - Update FULL/EMPTY
  - Write
    - push(DI)
    - Reset WR
  - Read
    - pop and store DO
    - DV is valid
    - Reset RD

```
FSM()                                          AsmL
★require (reqmode = PROCESS )
★reqmode := READWRITE
 if (mode = INIT)
   mode := REQUESTS
 if (mode = REQUESTS)
  step
    Full() // Update FULL status
    Empty() // Update EMPTY status
  step
   if ( WR.read() = true and FULL.read() = false )
    push(DI.read()) // Throws excep. overflow
    WR.write( false )
   else
    if ( RD.read() = true and EMPTY.read() = false )
     DO := pop() // Throws excep. overflow
     DV.write( true )
     RD.write( false )
   mode := WAIT
 if ( mode = WAIT)
   DV.write( false )
   mode := REQUESTS
```

Phase B: Implementation model

- **Translation is intuitive**
- **Possible errors**
  - **Overflow**
  - **Underflow**
- **Validation in SystemC**

SystemC

Possible overflow

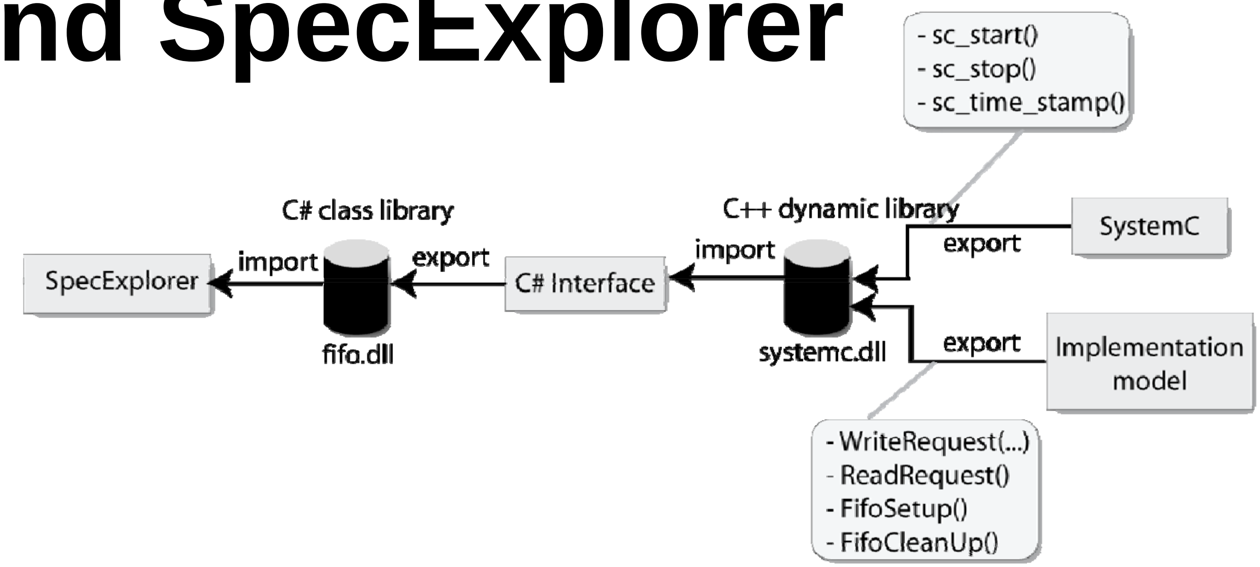
Possible underflow

```
void FSM() {
  if ( mode == INIT )
   mode = REQUESTS;
  else if ( mode == REQUESTS ) {
   Full(); Empty();
   if ( (WR == true ) /*&& (FULL == false)*/ ) {
        q.push(DI); // Throw excep. overflow
        WR = false;
        write_req.write( false );
   }
   else if ( (RD==true) /*&& (EMPTY==false)*/ ) {
        DO = q.front(); q.pop(); // Excep. underflow
        DV = true;
        RD = false;
        read_req.write( false );
   }
   mode = WAIT;
  }
  else if ( mode == WAIT ) {
   DV = false;
   mode = REQUESTS;
  } }
```

Phase B: Exporting SystemC and design actions

- SystemC wrapper
  - Must be done once only
  - sc_start() to drive simulation
  - Additional members may be exported for debugging purposes
- Implementation model wrapper
  - Stimulus that drives model
    - FIFO example
      - WriteRequest
      - ReadRequest
  - Setup and cleanup

Phase C: Interfacing SystemC, C# and SpecExplorer



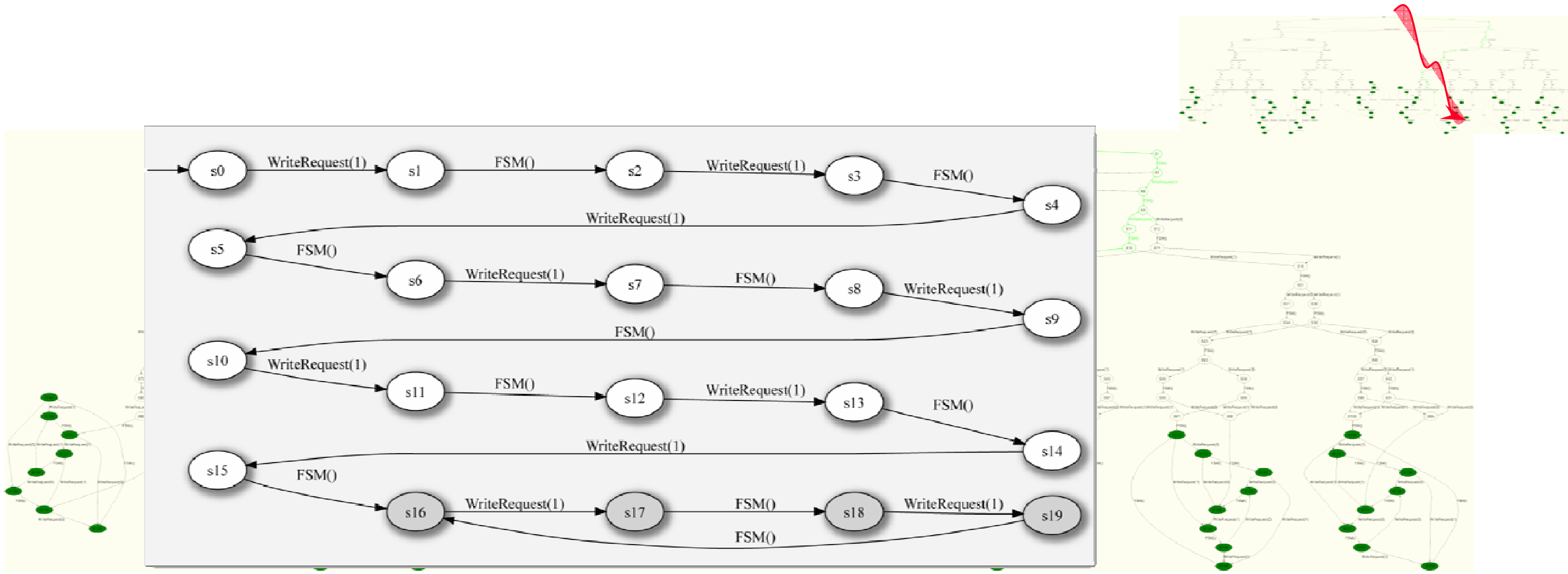
- **Import actions from SystemC and implementation wrappers**
- **Abstract C# class**
  - **Invokes imported actions**
- **Build C# class as reference library**
- **SpecExplorer refers to C# library**
  - **Actions bound to abstract class actions**

Hiren D. Patel
Virginia Tech.

Phase D: Exploration and test suite generation

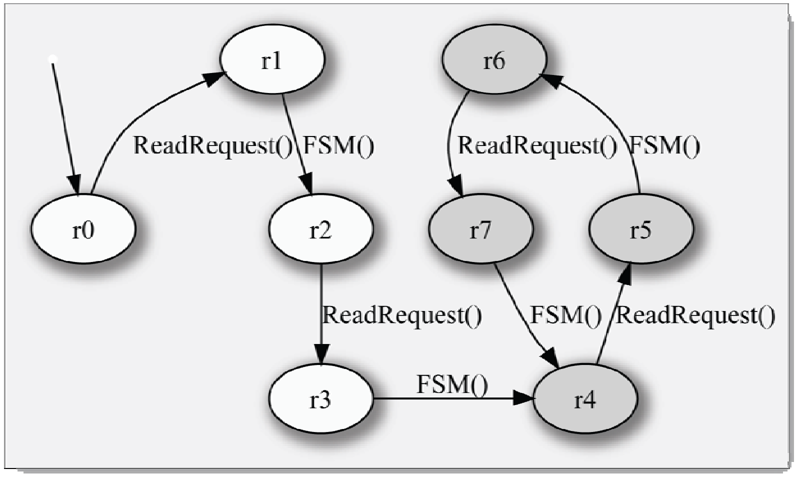
- Exploration techniques
  - State groups
  - State filters
  - Accepting states
  - Controllable, observable and scenario actions
- Test suite generation
  - Bind exploration actions to C# interface actions
- Generate test suite
  - All possible paths to accepting states

Testing for overflow



- **Size n=3**
- **Accepting state allowing for n+1 writes**
  - **Only valid write requests are when in REQUESTS**
- **Throws an exception**

Testing for underflow



- Size n=3
- Accepting state allowing for read request when FIFO is empty
- Throws an exception

Summary and Future work

- MoC-driven design to SystemC
  - Work on synthesis from MoC descriptions
- Model-driven methodology for validation of heterogeneous designs in SystemC
  - Extend framework for automatic translation of MoCs
  - Add other MoCs
  - Examples: network routers, FIR filters, …
- Integrate related researches to realize recipe

Sorry – no more slides!