USC Viterbi
School of Engineering

A SystemVerilogCSP Front-End to an Asynchronous ASIC Flow

By:

Arash Saifhashemi, Mehrdad Najibi, Yujun Cao, Chen Qian, Gang Wu, Aman Mehra, Roberto Vera, Boyi Huang, and Peter A. Beerel

USC Asynchronous CAD/VLSI Group

http://async.usc.edu/

ICCAD (MSCAS) 2012

USC

Agenda

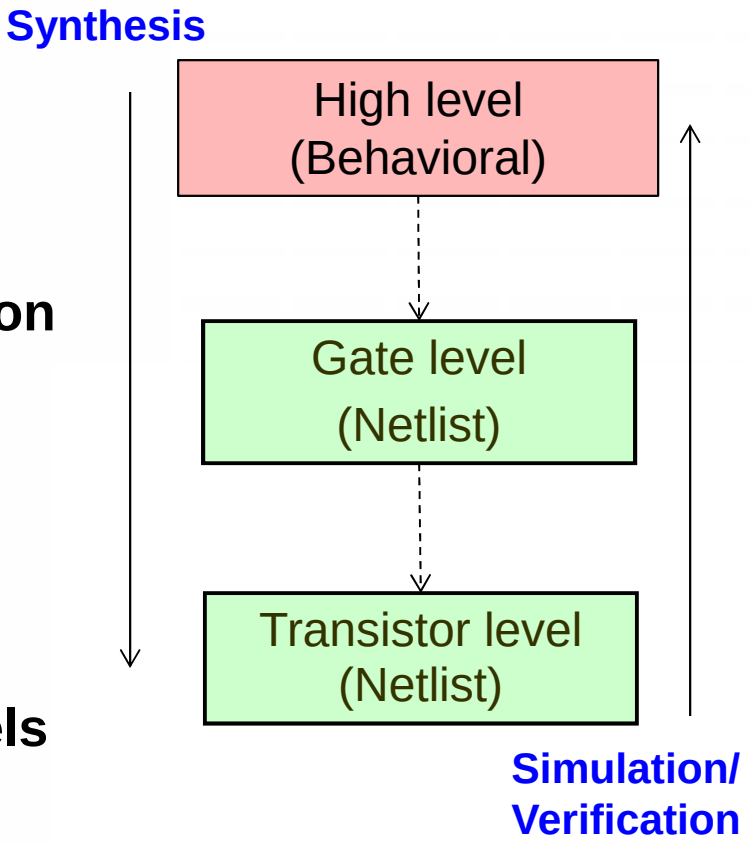
USC Viterbi School of Engineering

- Motivation
- SystemVerilogCSP (SVC) for modeling
- Conversion to RTL (SVC2RTL)
  - Proteus front-end
- Mixed mode simulation
  - Multiple abstraction levels
  - Mixed sync/async simulation

Motivation

USC Viterbi School of Engineering

- Desirable features
  - **Concurrency** e.g.: A=B || (C=D ; E=F)
  - **CSP-like** communication actions
  - **Timing** e.g.: A=B after 5ns
  - Support for **various levels of abstraction**
  - Support by **commercial CAD tools**
  - Ease of **adoption** by synchronous designers
  - Mixed **async-sync** circuits
  - Interfacing at **multiple abstraction levels**
  - Use as **Proteus front-end**
  - **Publicly** available (open source)



Previous Work

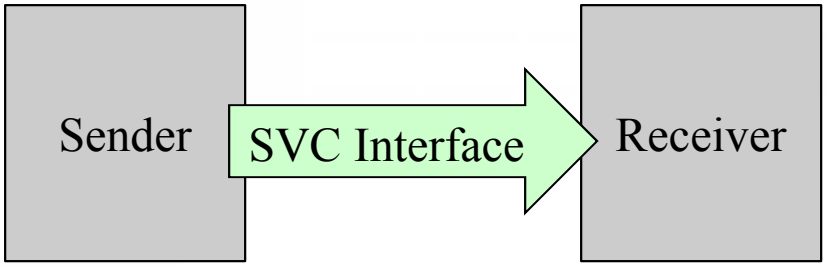
USC Viterbi School of Engineering

- **New Language inspired by CSP**
  - Have limited CAD tool support - LARD [Edwards et al], Tangram [Berkel et al.], CHP [Martin], Haste [Peeters et al.]
- **Software languages**
  - No inherent support for timing, limited CAD tool support - JCSP [Welch et al.]
- **VHDL**
  - Fine grained concurrency is cumbersome [Frankild et al, Renaudin et al, Myers et al]
- **VerilogCSP**
  - Verilog Programming Language Interface: slow; cannot handle multi-channel modules [Saifhashemi et al]
  - Verilog macros are cumbersome and do not support extensions
- **SystemC** [Koch-Hofer, Shanker, …]
- **SystemVerilog** [Tiempo]

SystemVerilogCsp (SVC)

USC Viterbi School of Engineering

- SystemVerilog interface abstracts **channel wires** as well as **communication protocol**
- **Send/Receive**
  - Blocking tasks



Abstract communication

```
module Sender (interface R);
  parameter WIDTH = 8;
  logic [WIDTH-1:0] data;
  always
  begin
   //produce data
   R.Send(data);
end
Endmodule
```

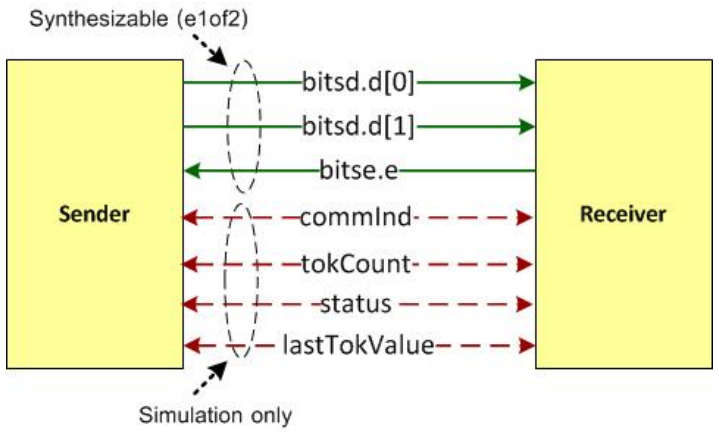
```
module Receiver (interface L);
  parameter WIDTH = 8;
  logic [WIDTH-1:0] data;
  always
  begin
    L.Receive(data);
     //consume data
   end
Endmodule
```

Waveform view

USC Viterbi School of Engineering

```
//Sender (DataGen)
always
  begin
    #Delay;
    R.Send(data);
End
```

```
//Receiver
always
  begin
    L.Receive(data);
    #FL;
    R.Send(data);
    #BL;
  end
```



Receiver pending on Receive

Sender performs Send, Communication happens

No one is Sending or Receiving

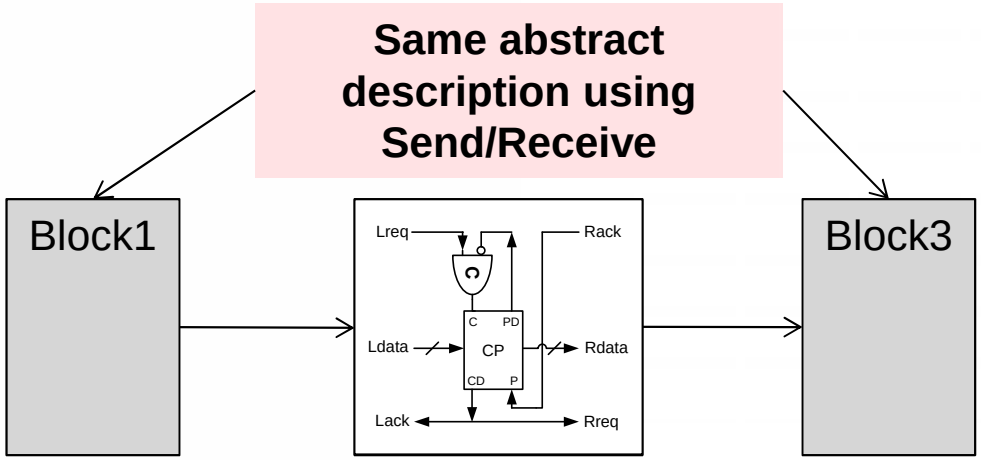
Sender pending on Send

Receiver performs Receive, Communication happens

Supports Mixed-Levels of Abstraction

USC Viterbi School of Engineering

- **Completed** blocks can be simulated with others **still at behavioral level**




```
module mp_fb_csp (interface L, interface R);
    logic data;
    always
    begin
        L.Receive(data);
        R.Send(data);
    end
endmodule
```

Gate-level description of the buffer (After synthesis)

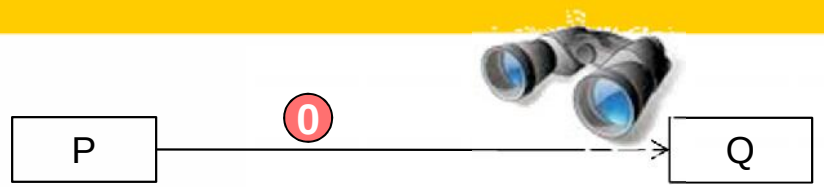
High-level description of the buffer (Before synthesis)

```
module mp_fb_gate (interface L, interface R);
  celement          ce(L.req, pd_bar, c);
  not       inv     (pd_bar, pd);
  cap_pass          cp (c, L.ack, R.ack, pd, L.data, R.data);
endmodule
```

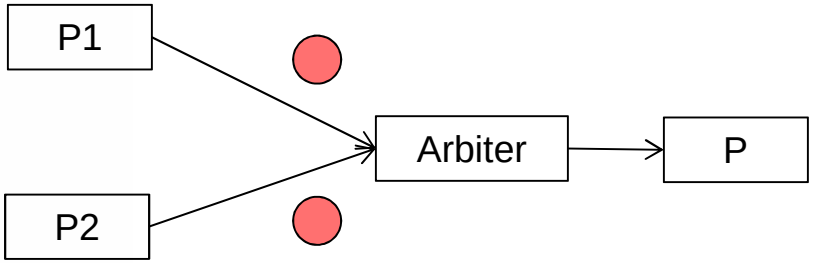
Peek and Probe [CHP, Martin]

USC Viterbi School of Engineering

- Peek
  - Sampling without committing to communication
- Probe
  - Is the channel idle?
  - Used for arbitration




```
task Peek (output logic[WIDTH-1:0] d);
  wait (status == s_pend );
  d = data;
endtask
```



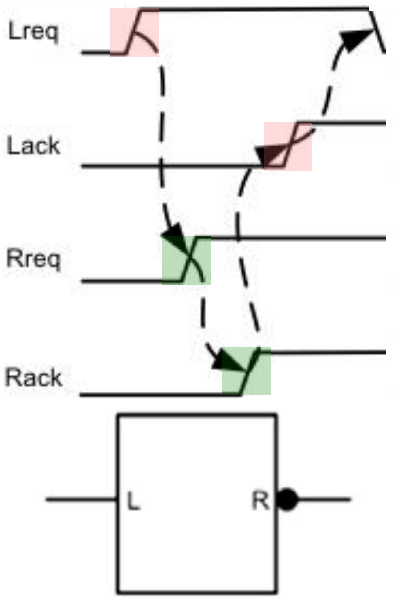

```
wait(ch0.status!=idle && ch1.status!= idle);
winner = Arbitrate (ch0.status, ch1.status);

if(winner == 0)
    ch0.Receive(d);
if(winner ==1)
    ch1.Receive(d);
```

Split Communication

USC Viterbi School of Engineering

- Handshaking of different channels might be **interleaved**
- Modeling interleaved behavior for **early system evaluation**



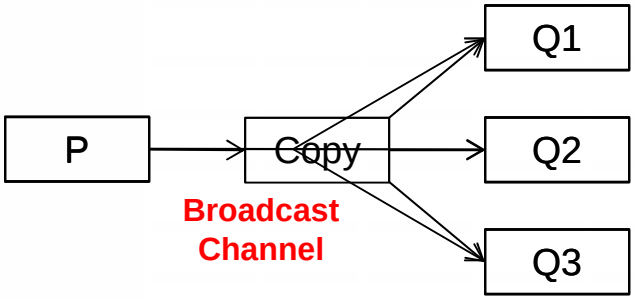

```
module buf (interface L, interface R);
    logic data;
    always
    begin
        L.Receive(data);
        R.Send(data);
    end
endmodule
```

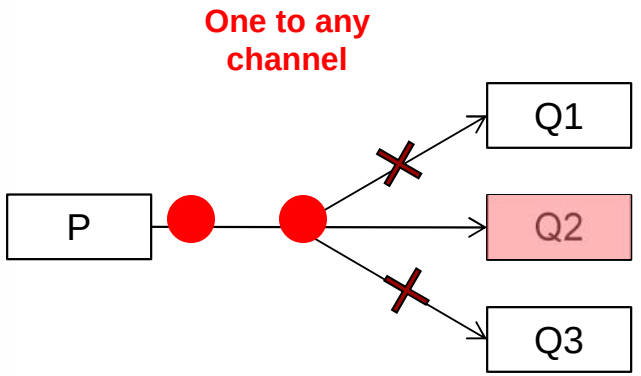
```
module buf_split (interface L, interface R);
    logic data;
    always
    begin
        L.SplitReceive(data, 1);
        R.Send         (data);
        L.SplitReceive(data, 2);
    end
endmodule
```

One-To-Many and One-To-Any Channels

USC Viterbi School of Engineering

- One sender to multiple receivers
  - Option 1: Use a copy block
  - Option 2: Shared channels [JCSP, Welch et. al]
    - Sender and receiver send and receive **as if the channel is a normal one-to-one channel**
    - Top level module specifies the channel is broadcast
- One sender to multiple receiver - JCSP [Welch et. al]
  - Only one of the receiver participates in communication





Agenda

USC Viterbi School of Engineering

- Motivation
- SystemVerilogCSP (SVC) for modeling
- Conversion to RTL (SVC2RTL)
  - Proteus front-end
- Mixed mode simulation
  - Multiple abstraction levels
  - Mixed sync/async simulation

SVC Front-end

USC Viterbi School of Engineering

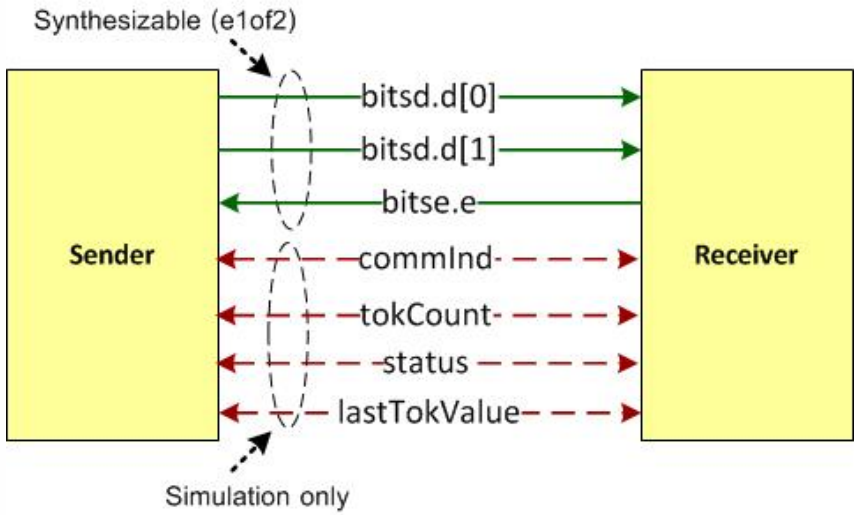
- The industrial version of Proteus uses CAST (based on CHP) as front-end.

```
module CondAccumulator_base;
define ABSTRACT_CondAccumulator ()
(e1of2 -C1,-C2,-C3, e1of2[16] -I1,-I2,+O){
  csp {
      //Initialization block
      s=0;
      //Forever block
      *[  x1=0, x2=0;
        C1?c1, C2?c2, C3?c3;
        [c1 → I1?x1 [] else → skip],
        [c2 → I2?x2 [] else → skip];
        s = s + x1 + x2;
        [c3 → O!s   [] else → skip];
      ]
  }
}
```

The e1ofN_M Interface

USC Viterbi School of Engineering

- CAST and Proteus naming convention for PCHB template [Lines'98]
  - An **M** size array of **e1-of-N** channels
- Only synthesizable signals are visible to the synthesizer



SVC2RTL Synthesizable Template

USC Viterbi School of Engineering

- Synthesizable SVC
  - Close to RTL
- Limitaitons
  - Only one Send/Receive per channel per iteration
  - Delays and fork/join blocks ignored

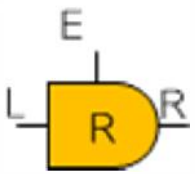
```
module CondAccumulator (
                e1of2_1.In   C1,C2,C3,
                e1of2_16.In  I1,I2,
                e1of2_16.Out O);
 logic [I1.W-1:0] x1;
 logic [I2.W-1:0] x2;
 logic[O.W-1:0] sum;
 logic c1, c2, c3;
 always begin
    sum = 0;        //Reset value
    forever begin
     x1=0; x2=0;  //Default values
     fork
      C1.Receive(c1);
      C2.Receive(c2);
      C3.Receive(c3);
     join
     if (c1) I1.Receive(x1);
     if (c2) I2.Receive(x1);
     sum= sum + x1 + x2;
     if (c3) O.Send(sum);
    end
 end
endmodule
```

Conditional Communication (RECEIVE/SEND Cells)

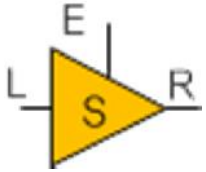
USC Viterbi School of Engineering

- Implement conditionality
  - RECEIVE:
    - E = 1: behaves like a buffer
    - E = 0: doesn't receive from left, but sends a dummy token to the right
  - SEND
    - E = 1: behaves like a buffer
    - E = 0: receives from left, doesn't send to right

```
always begin
  forever begin
    E.Receive(e);
    if (e==1) L.Receive(d);
    else      d=0;
    R.Send(d);
  end
end
```



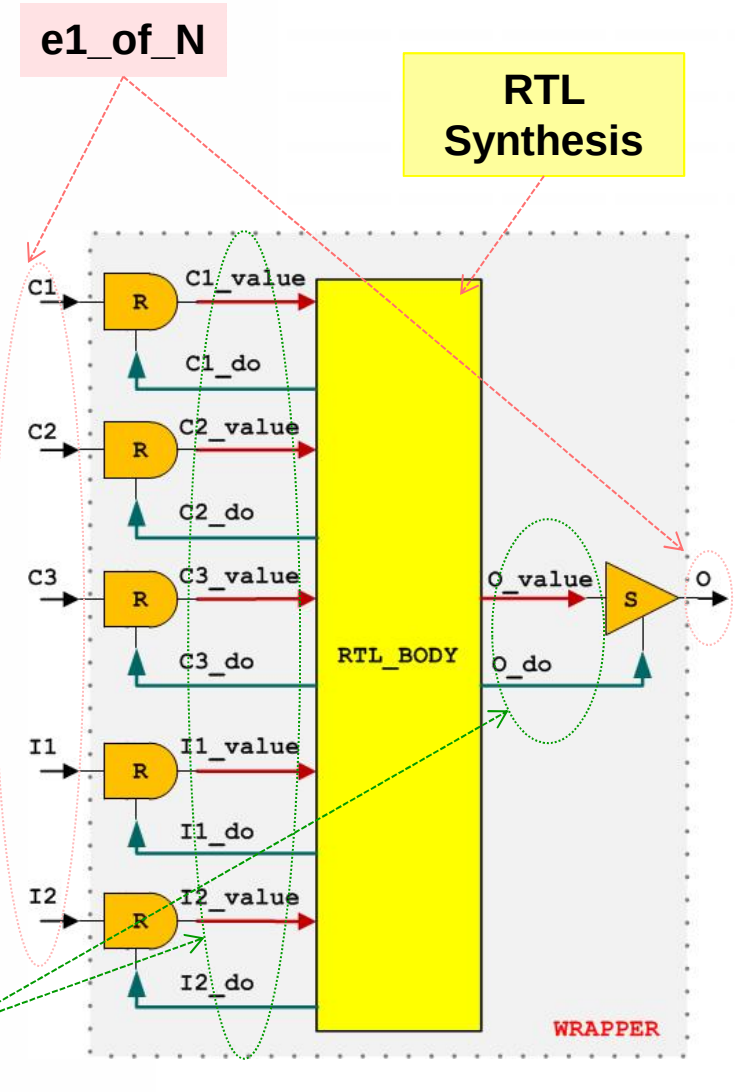

```
always begin
  forever begin
    E.Receive(e);
    L.Receive(d);
    if (e==1)
      R.Send(d);
  end
end
```



The Wrapper Module

USC Viterbi School of Engineering

- SVC2RTL - Creates Top-Level Wrapper
  - Instantiate SEND and RECEIVE cells
    - Single-rail on one side
    - 1-of-N on other side
  - Create and Instantiate RTL Body
    - Implements the logic and enables inputs for SEND/RECEIVE
    - Single-rail on both sides
    - Each iteration is mapped to one clock cycle
- Synthesized using RTL synthesizer
  - The “Image” netlist



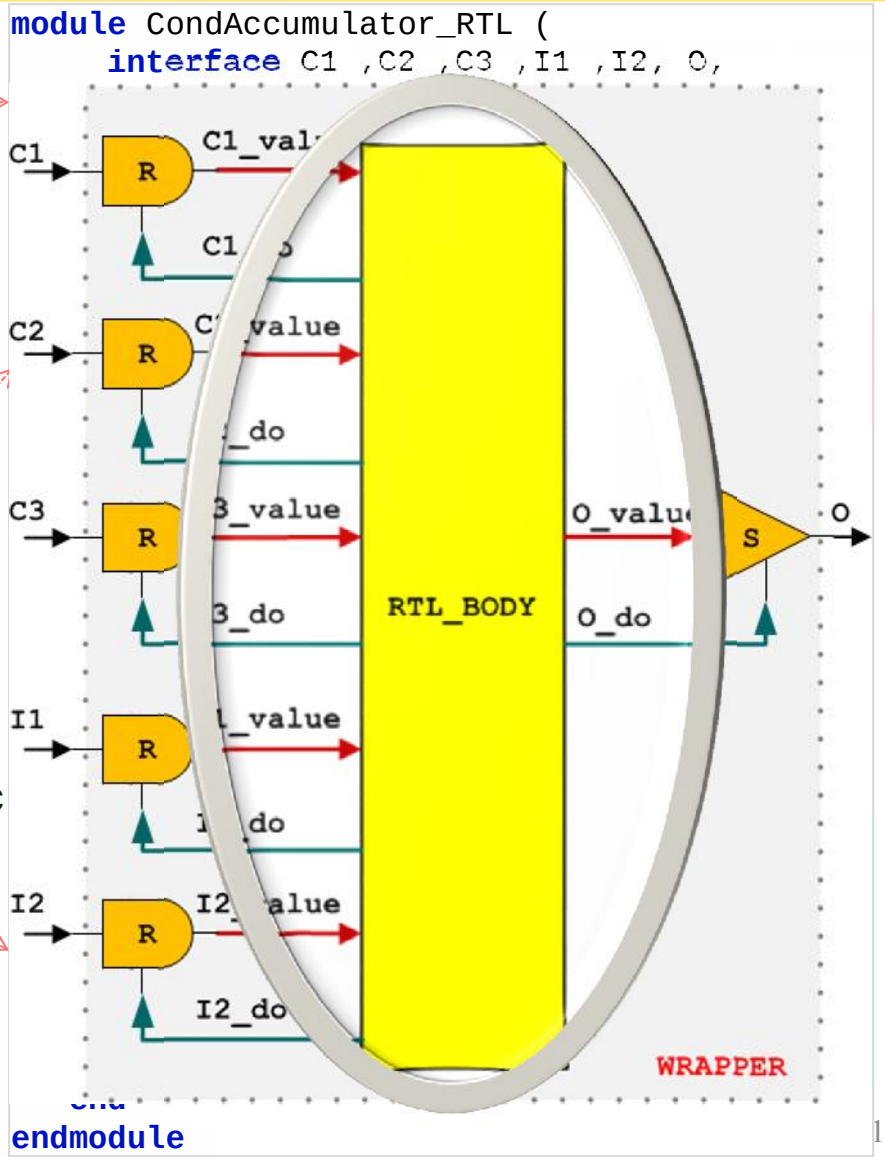
The RTL_Body with Synthesizable Tasks

USC Viterbi School of Engineering

Receive: samples **value**, and asserts **do**
Send: asserts both **value** and **do**
InitDo: assigns 0 to do
InitValue: assigns 0 to value

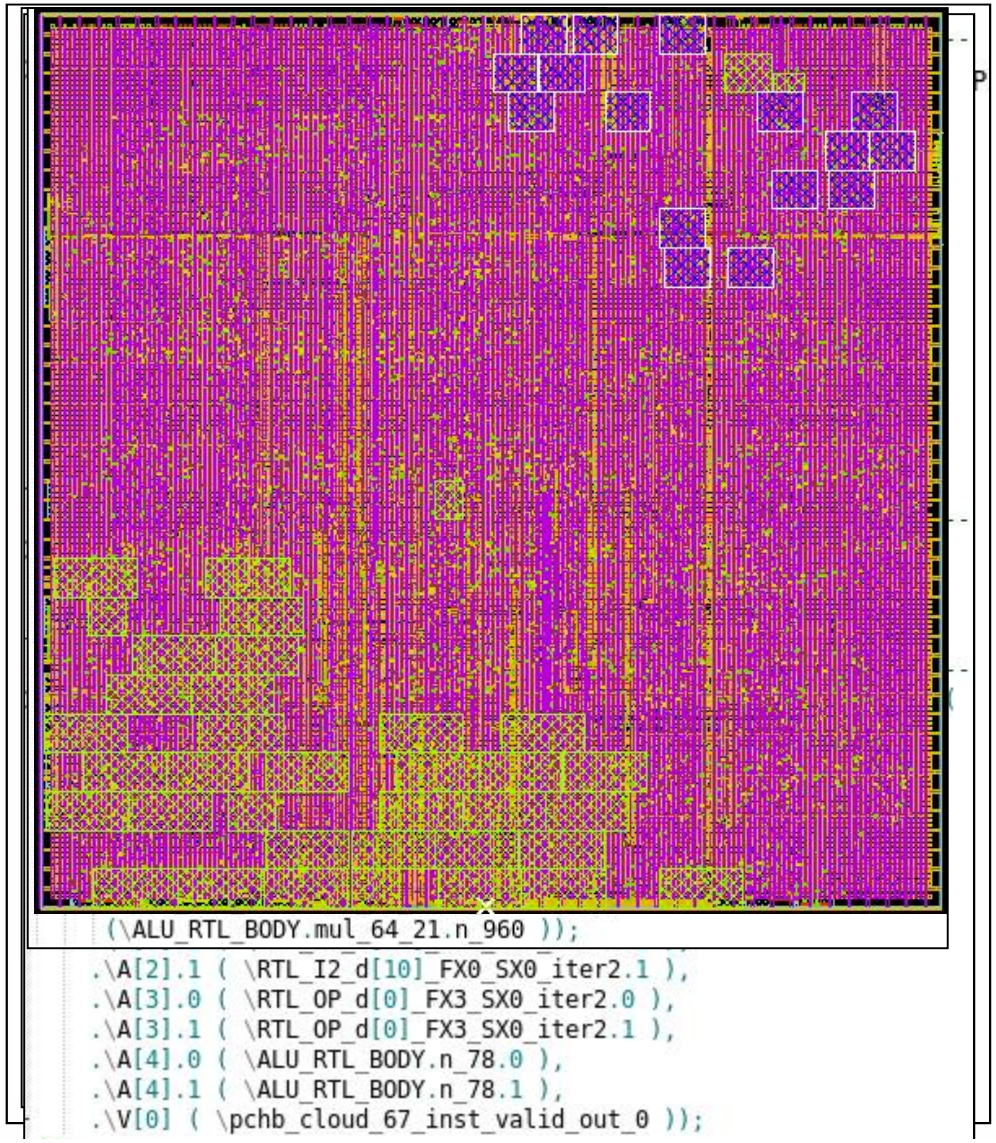
```
module CondAccumulator (
                e1of2_1.In   C1,C2,C3,
                e1of2_16.In  I1,I2,
                e1of2_16.Out O);
 logic [I1.W-1:0] x1;
 logic [I2.W-1:0] x2;
logic[O.W-1:0] sum;
 logic c1, c2, c3;
 always begin
    sum = 0;          //Reset value
    forever begin
     x1=0; x2=0;     //Default values
     fork
     C1.Receive(c1);
     C2.Receive(c2);
     C3.Receive(c3);
     join
     if (c1) I1.Receive(x1);
     if (c2) I2.Receive(x1);
     sum= sum + x1 + x2;
     if (c3) O.Send(sum);
    end
 end
endmodule
```

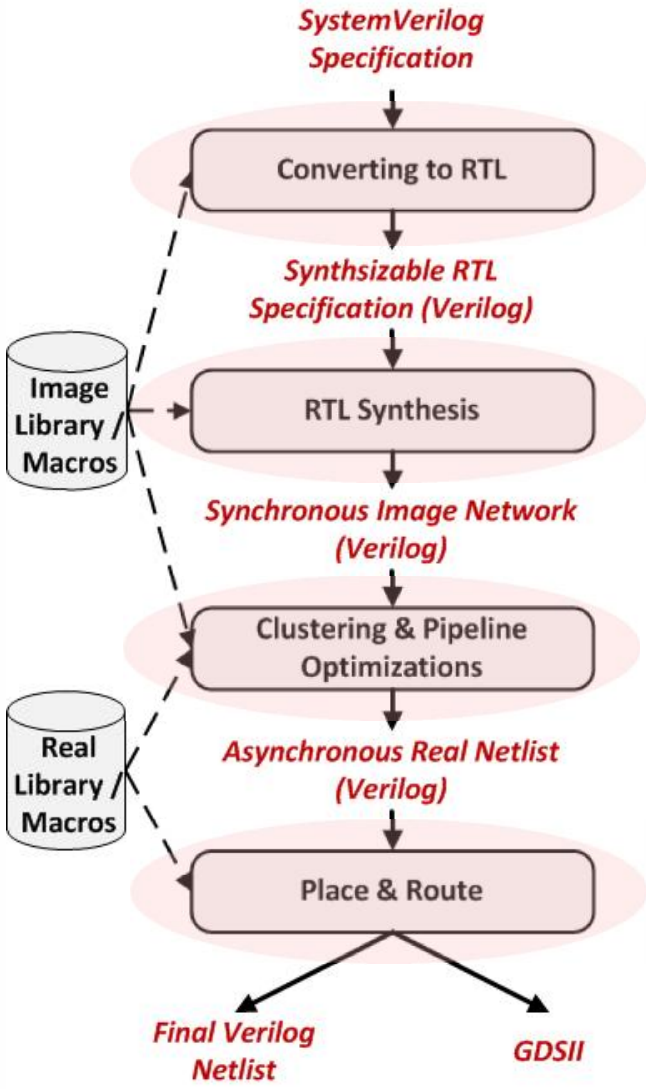
Output and next state logic



Proteus-a Flow

USC Viterbi School of Engineering





Agenda

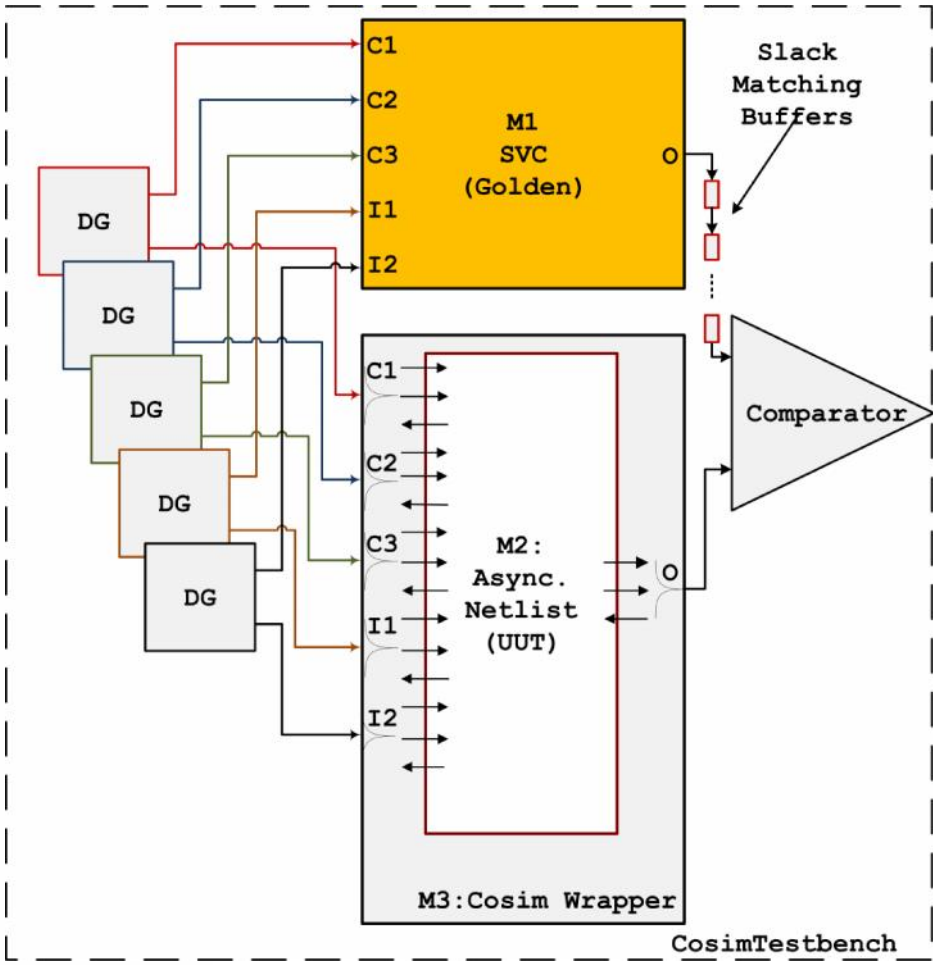
USC Viterbi School of Engineering

- Motivation
- SystemVerilogCSP (SVC) for modeling
- Conversion to RTL (SVC2RTL)
  - Proteus front-end
- Mixed mode simulation
  - Multiple abstraction levels
  - Mixed sync/async simulation

Validation by Co-simulation

USC Viterbi School of Engineering

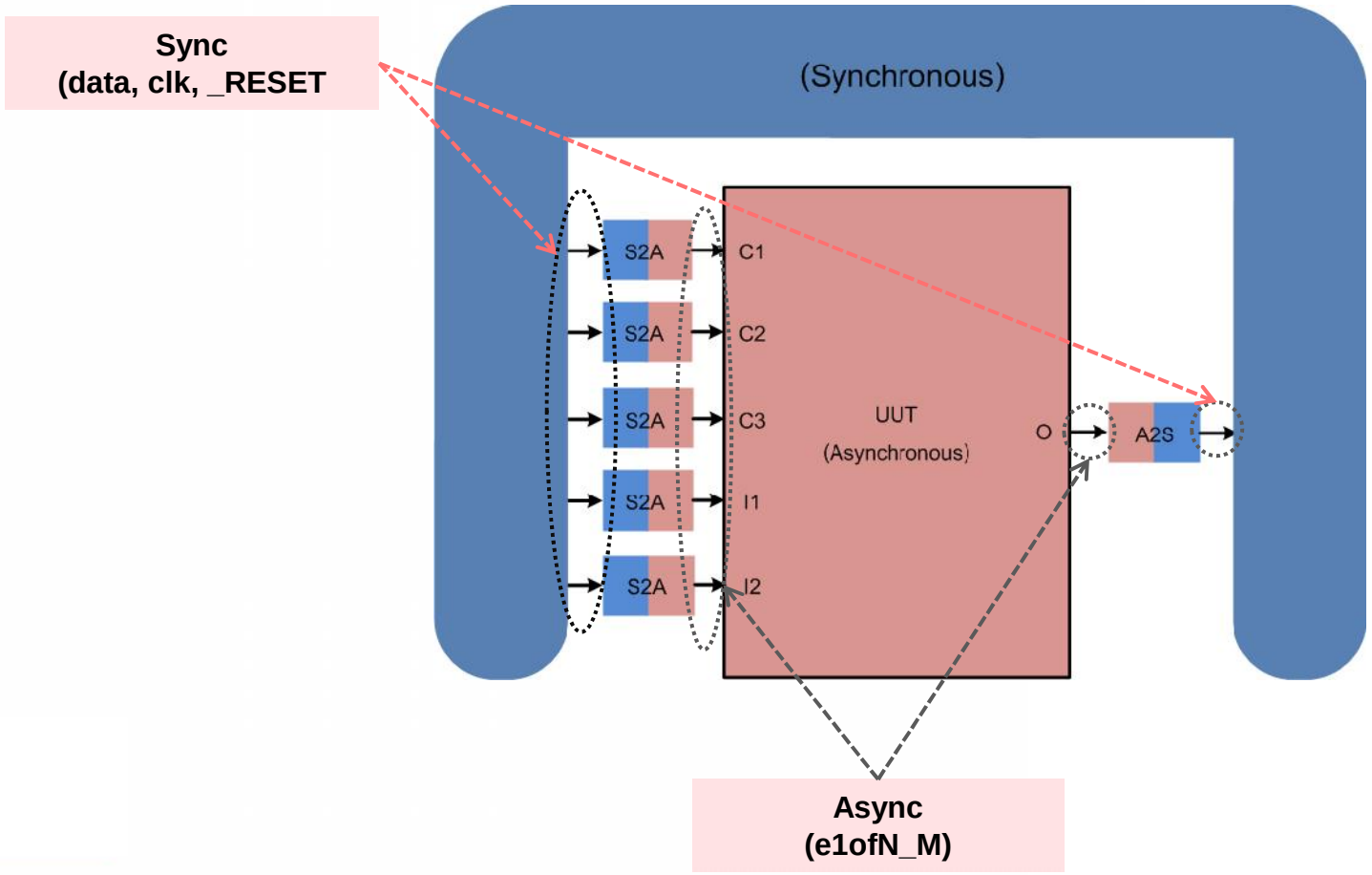
- Testbench includes:
  - The SVC module (Golden)
  - The output of Proteus (UUT)
  - Data generator at inputs
  - Comparators at outputs



Mixed Sync/Async Design

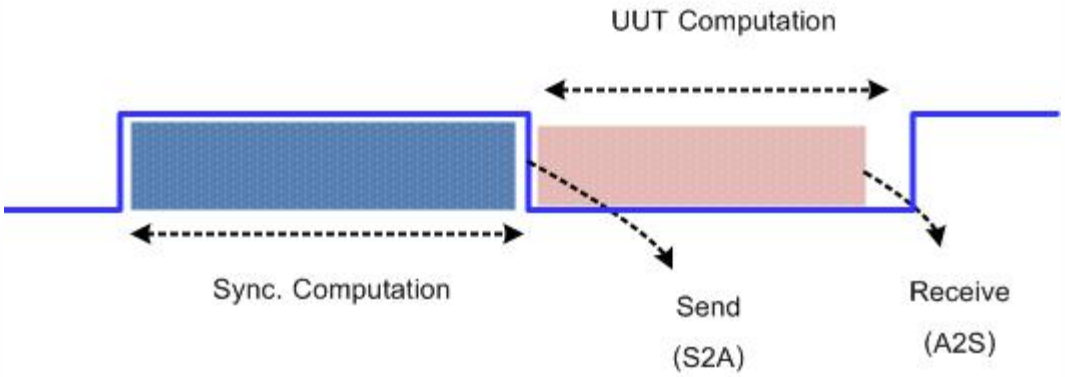
USC Viterbi School of Engineering

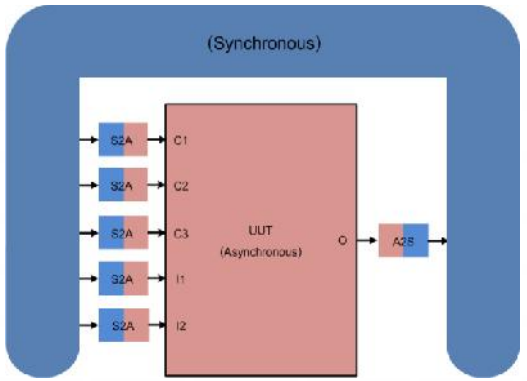
- Asynchronous island with A2S and S2A



S2A and A2S

USC Viterbi School of Engineering






```
module s2a #(parameter SAMPLE_DELAY=10ns)
(
  interface sync    //has data, CLK, and _RESET
  interface async,  //e1ofN_M interface
);
always begin
    //Wait for reset pulse
    wait (sync._RESET == 0);
    wait (sync._RESET == 1);
    forever begin : sample
       @(negedge sync.CLK)
       #SAMPLE_DELAY;
       async.Send(sync.data);
    end
end
```

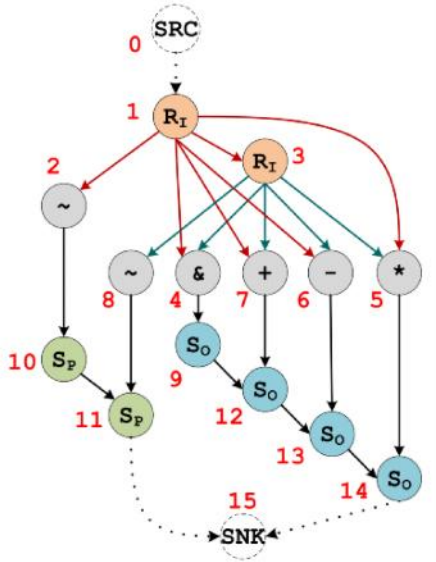
```
module a2s #(parameter M=1)
(
  interface async,  //e1ofN_M interface
  interface sync    //has data, CLK, and _RESET
);
logic [M-1:0] validValue='0;
always begin : receive_and_assert
       //Wait for reset pulse
       wait (sync._RESET == 0);
       wait (sync._RESET == 1);
       forever
       begin
          @(negedge sync.CLK)
          async.Receive(validValue);
       end
end
assign sync.data = validValue;
endmodule
```

Future Extension

USC Viterbi School of Engineering

- Addressing single Receive/Send limitation:
  - Create a CDFG
  - Map to the classic minimum latency scheduling under resource constraints

```
//Serial ALU
forever begin
  I.Receive(x);
  I.Receive(y);
  xNot = ~x;
  yNot = ~y;
  oAnd = x & y;
  oAdd = x + y;
  oSub = x - y;
  oMul = x * y;
  O.Send(oAnd);
  O.Send(oAdd);
  O.Send(oSub);
  O.Send(oMul);
  P.Send(xNot);
  P.Send(yNot);
end
```



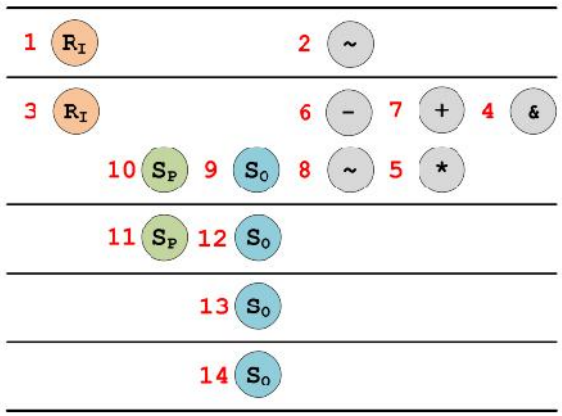
$$\text{minimize } \mathbf{c}^T\mathbf{t} \text{ such that:}$$

$$\sum_{l=1}^{L} x_{il} = 1, \quad \forall i = 0, \ldots, n$$

$$\sum_{l=1}^{L} l.x_{il} \geq \sum_{l=1}^{L} l.x_{jl}, \quad i, j = 0, \ldots, n, \quad (v_j, v_i) \in E$$

$$\sum_{i:C(v_i)=k} x_{il} \leq 1, \quad \begin{cases} k & = 1, \ldots, C_{max} \\ l & = 1, \ldots, L \end{cases}$$

$$x_{il} \in \{0, 1\}; \quad i = 0, \ldots, n; \quad l = 1, \ldots, L$$



Summary and Conclusions

USC Viterbi School of Engineering

- Asynchronous circuits can/should leverage sync. tools.
- SVC is suitable for channel based async. Circuits.
  - Both for modeling and synthesis
  - Mixed mode simulation
- Standard and open source
  - SystemVerilogCSP for modeling and simulation:

http://async.usc.edu/index.php/research/current/9-systemverilogcsp

Questions: saifhash@usc.edu

  - Proteus-a (Academic license): pabeeral@usc.edu