



# Co-Design and Co-Verification using a Synchronous Language

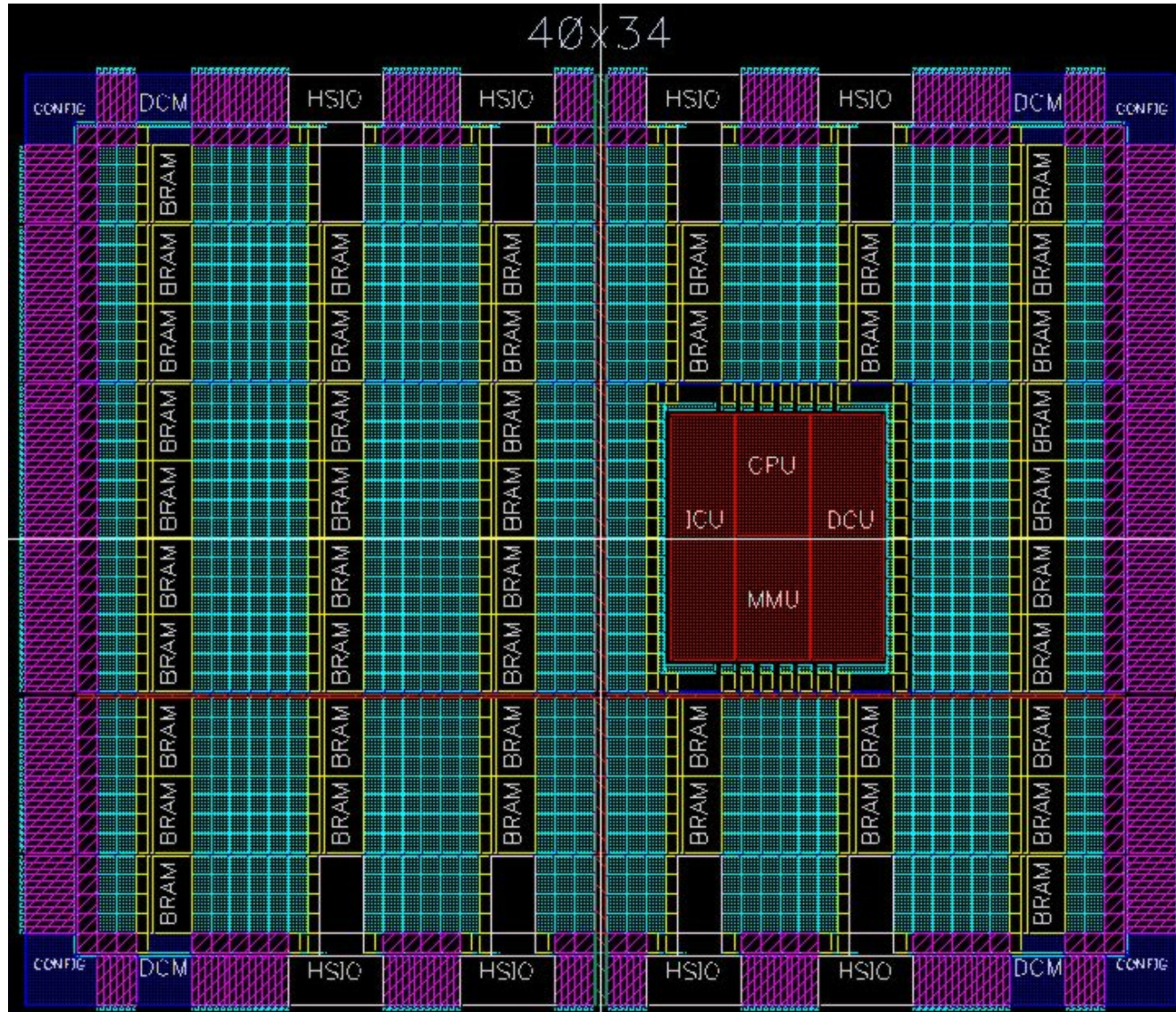
Satnam Singh  
Xilinx Research Labs



Software

Hardware

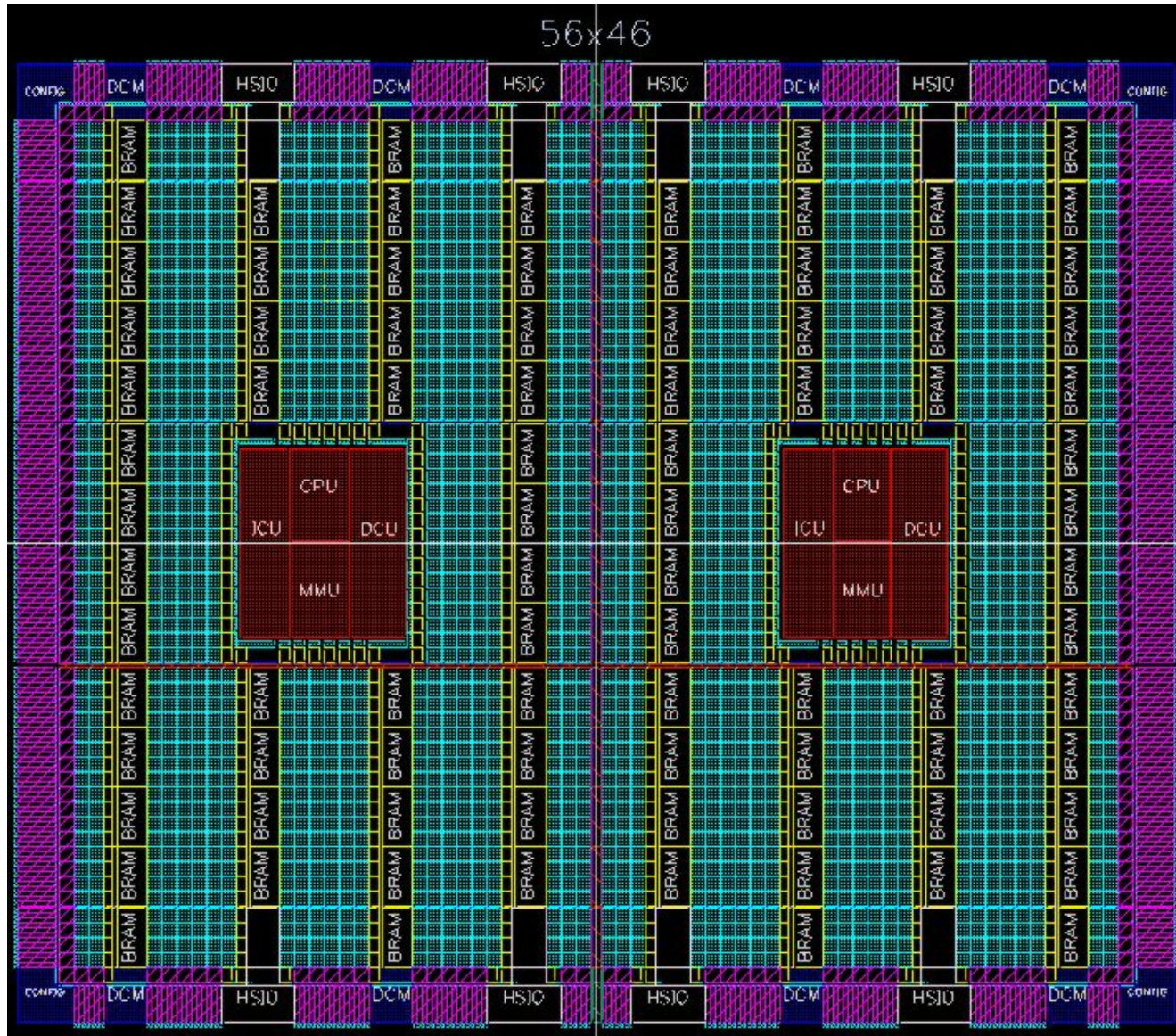




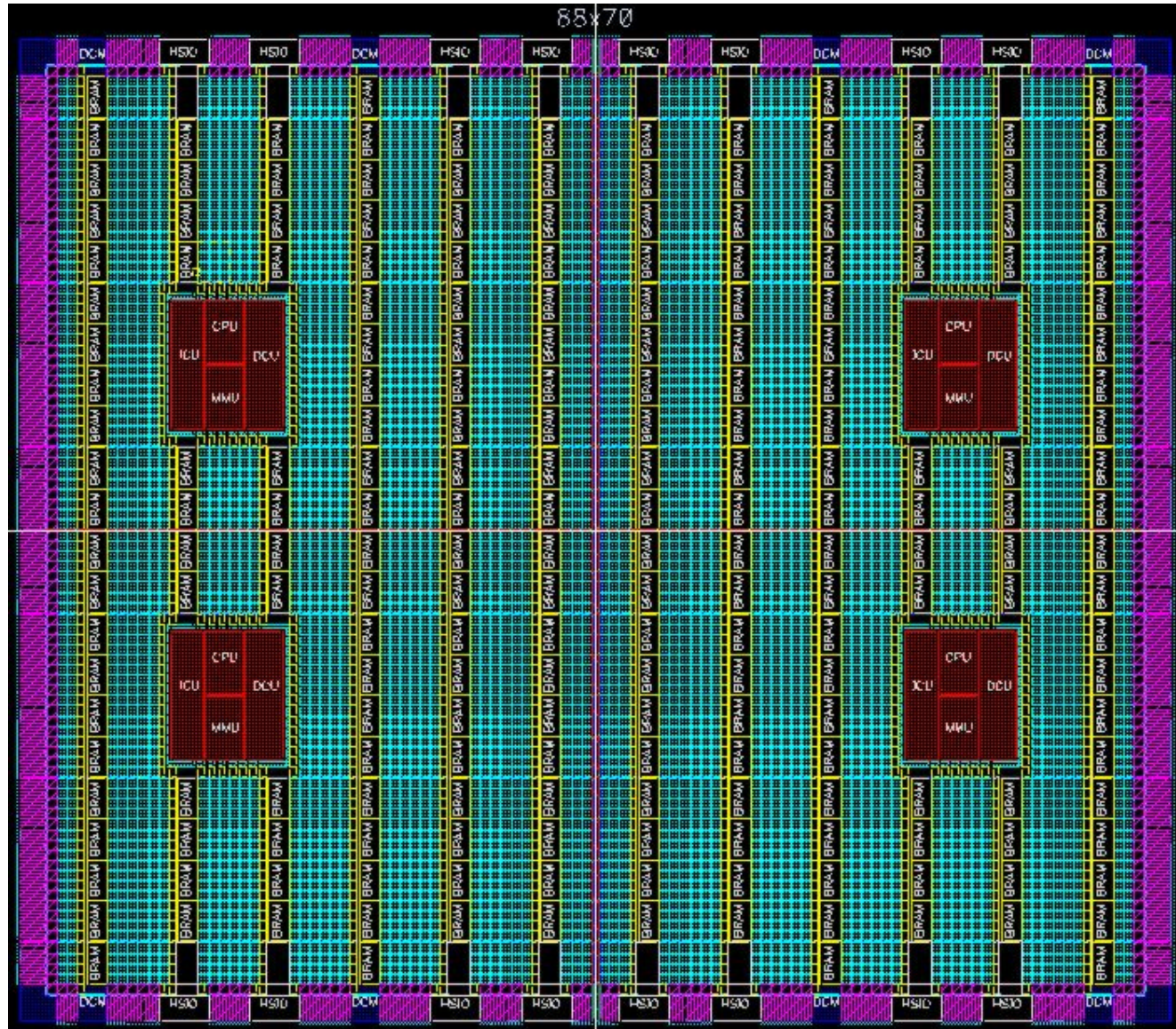
# Virtex-II PRO

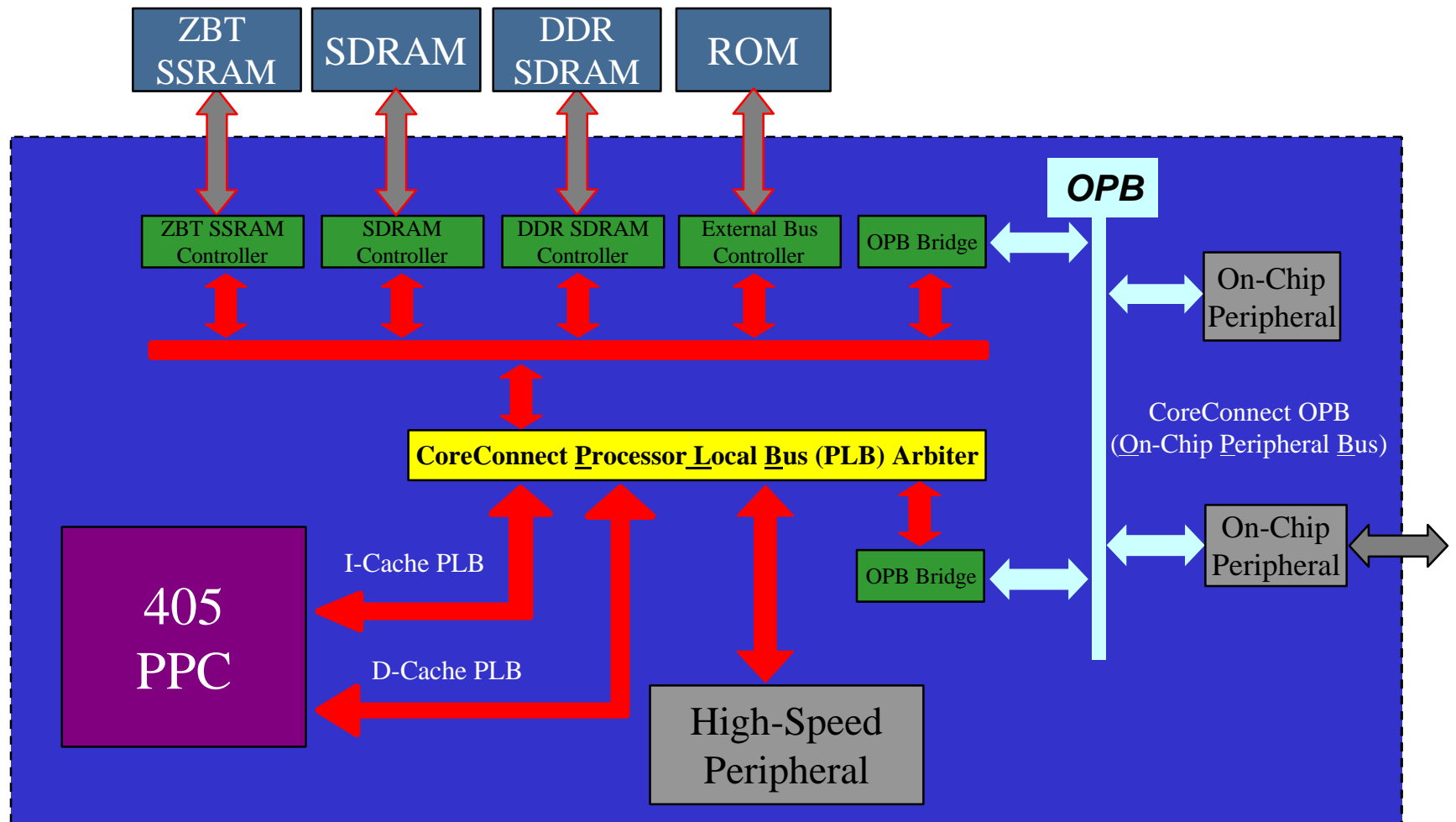
<i>Device</i>	<i>Array Size</i>	<i>Logic Gates</i>	<i>PPCs</i>	<i>GBIOs</i>	<i>BRAMs</i>
<b>2VP2</b>	16 x 22	38K	0	4	12
<b>2VP4</b>	40 x 22	81K	1	4	28
<b>2VP7</b>	40 x 34	133K	1	8	44
<b>2VP20</b>	56 x 46	251K	2	8	88
<b>2VP50</b>	88 x 70	638K	4	16	216













# Formal Techniques Project

- Domain specific languages for hardware design and verification (Lava) and cryptology (Cryptol).
- Formal methods for CAD (routing) and dynamic reconfiguration (reconfiguration controllers).
- **Formal notations/representations for HW/SW co-design and verification (e.g. Esterel)**
- Property checking (PSL/Sugar)
- IP-reuse (more powerful type systems la de Alfaro and Henzinger)

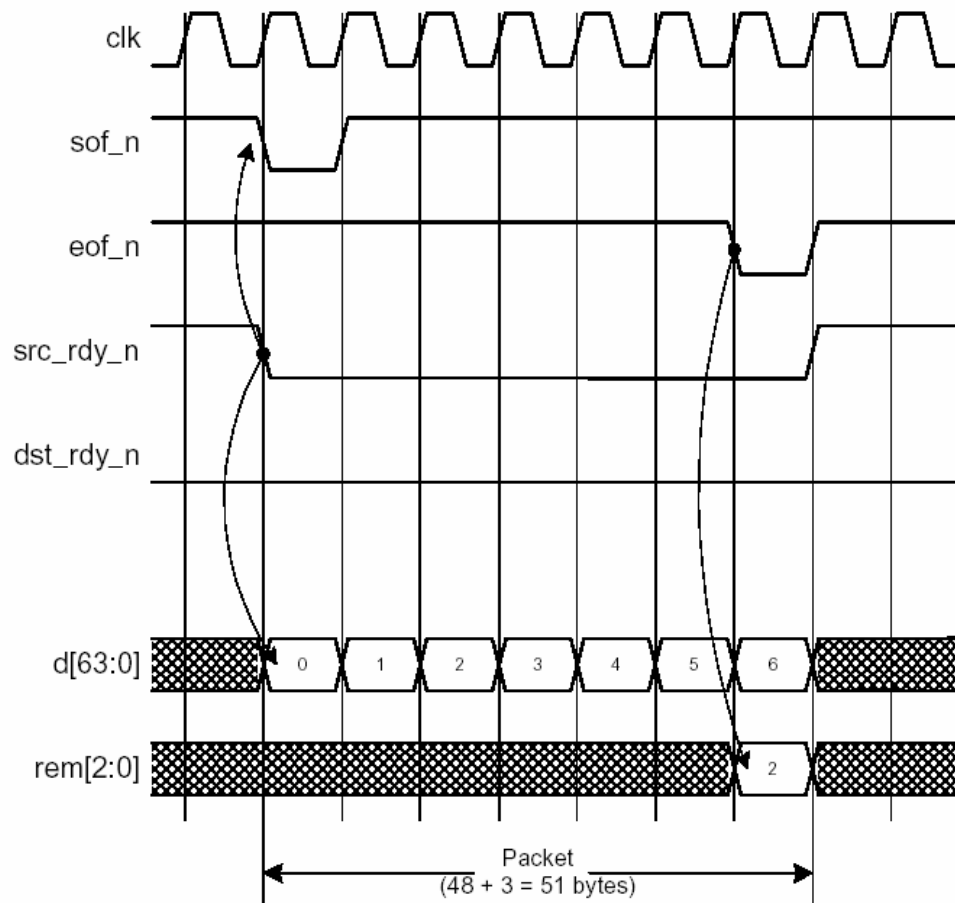


# ~~Problems~~ Challenges

- Customer requirements for migrating software into hardware (“salmon effect”):
  - determinism: multiple SW processes on RTOS vs. genuinely concurrent HW
  - verification
  - isolation
- Customer requirements to trade-off HW/SW partitioning for products at different price points.
- Requirements for verification of SW+HW
- Safe Dynamic Reconfiguration
- Verification of control-based systems



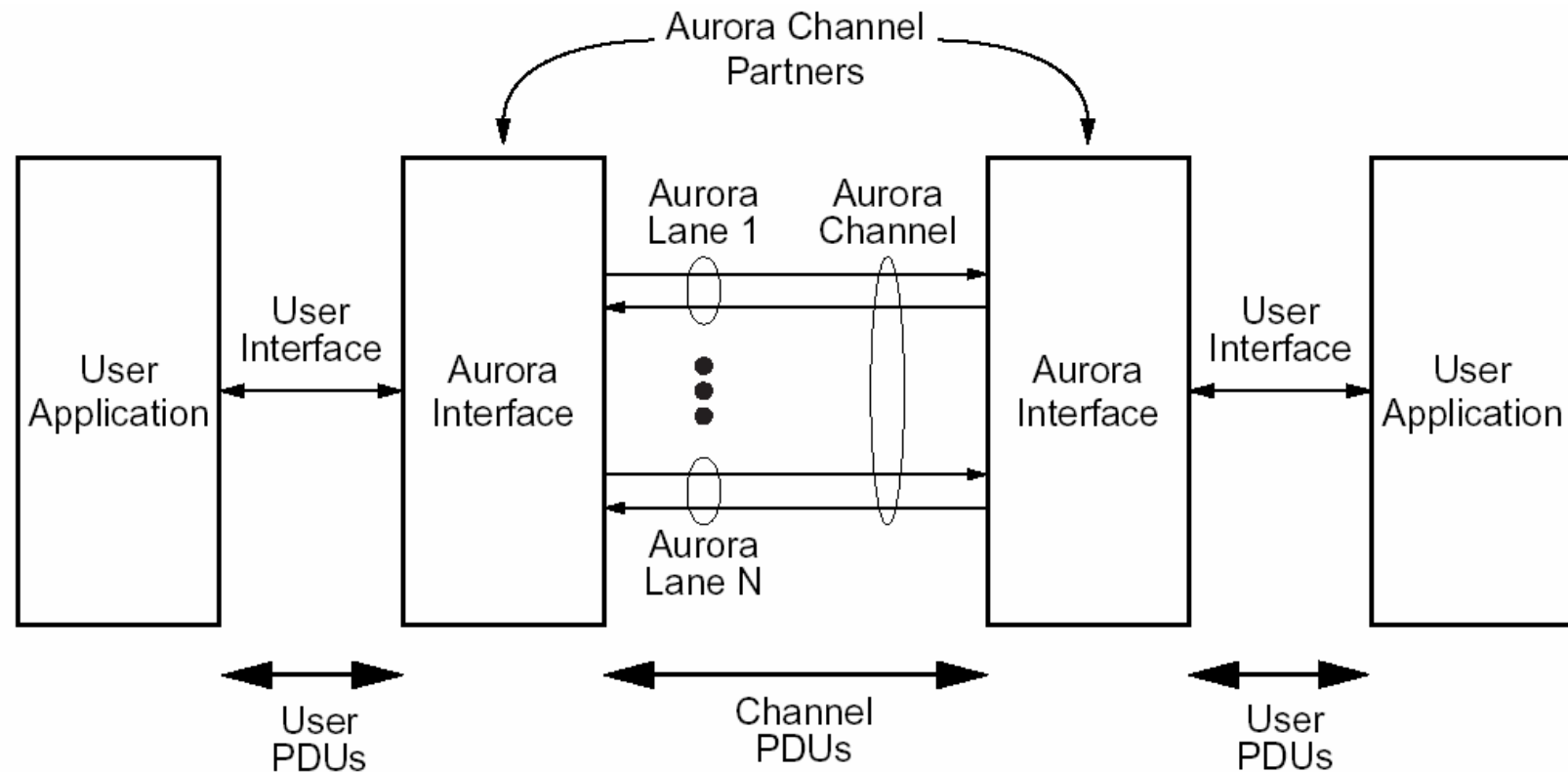
# LocalLink (Point to Point)



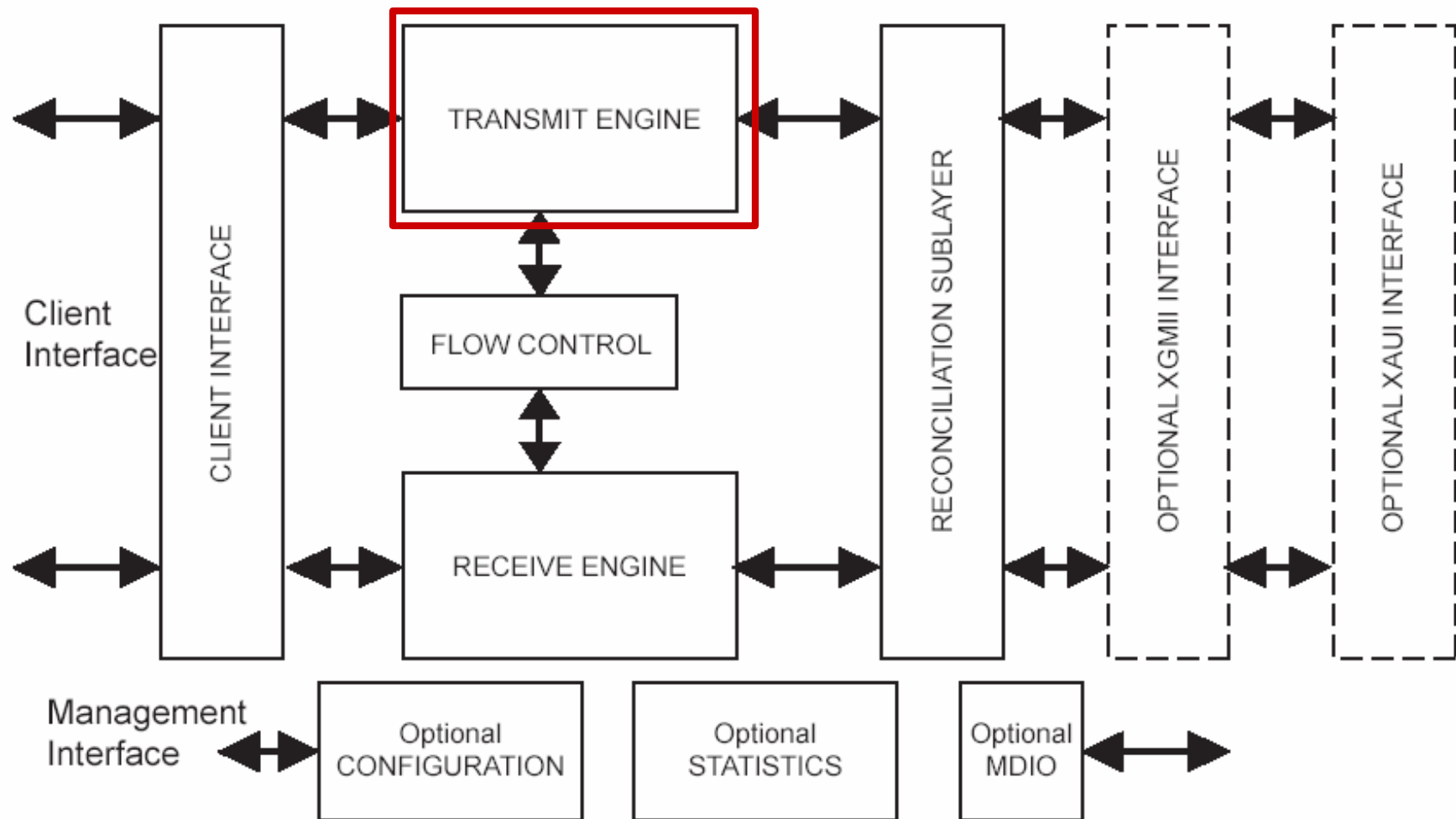
xip109



# Aurora (Link Layer Protocol)

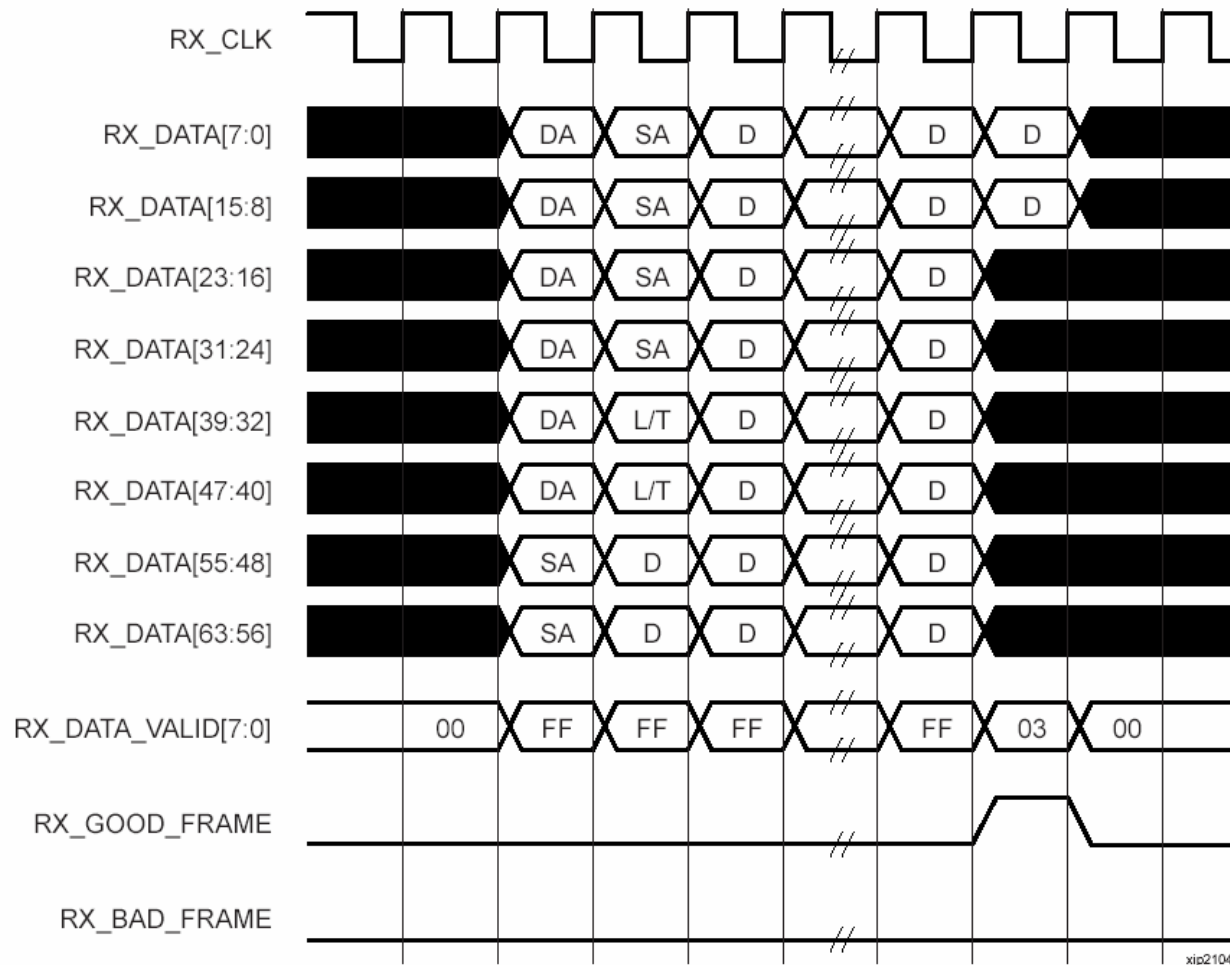


# TX of 10 Gigabit Ethernet MAC

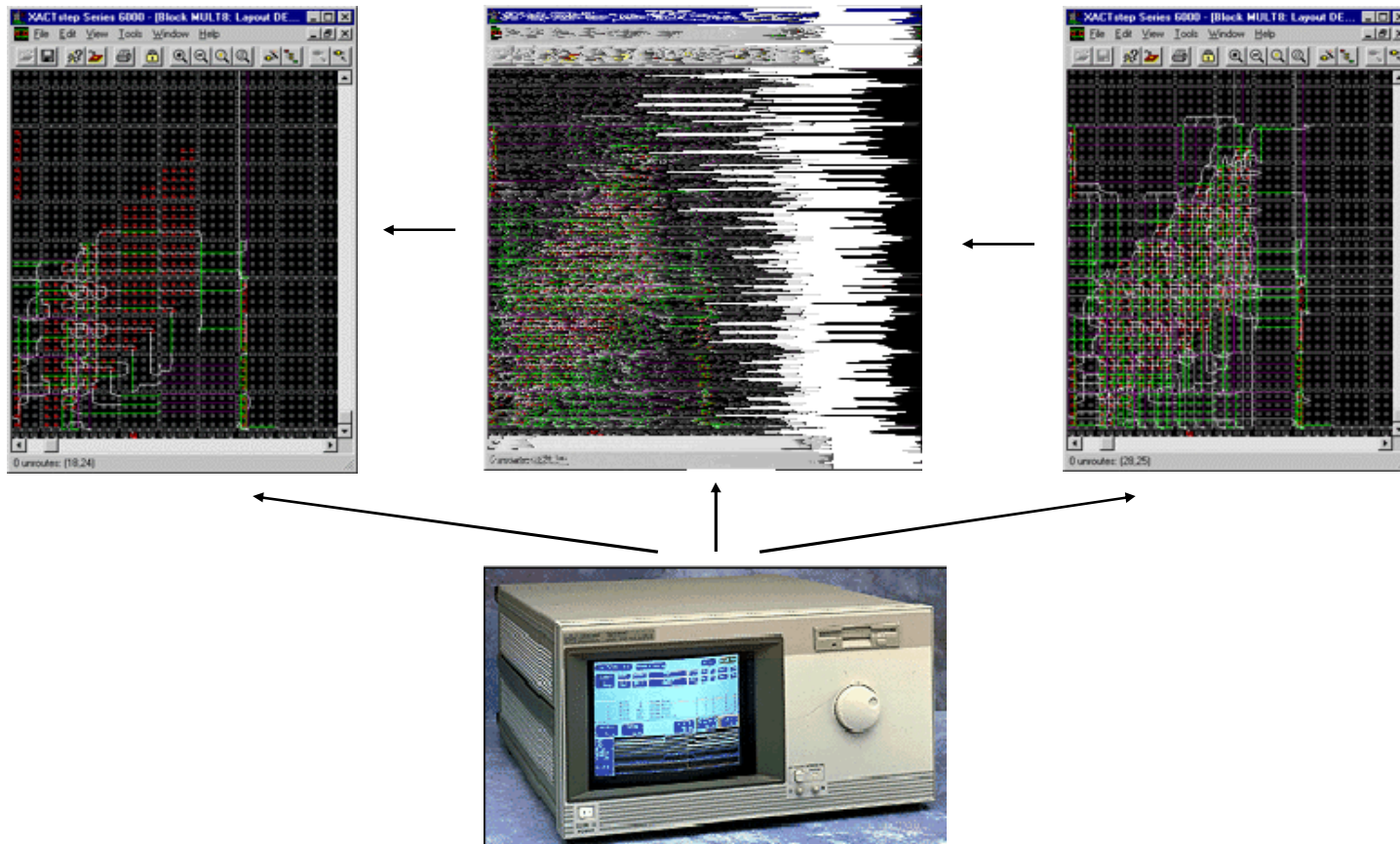




# Verify RX Control Signals

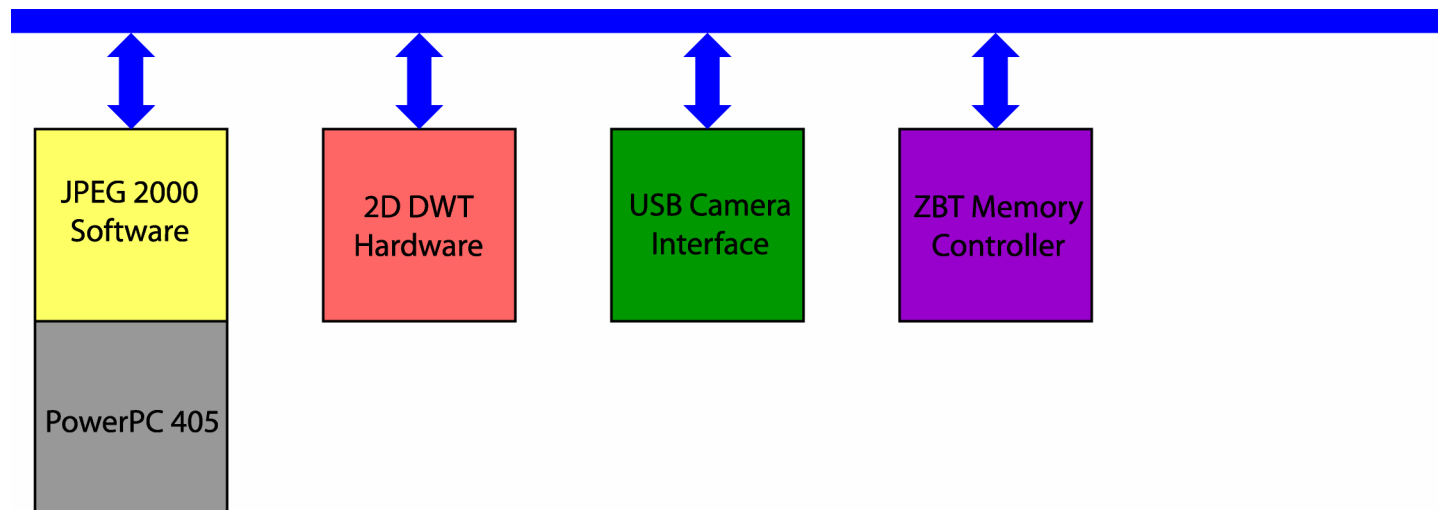


# Safe Dynamic Reconfiguration

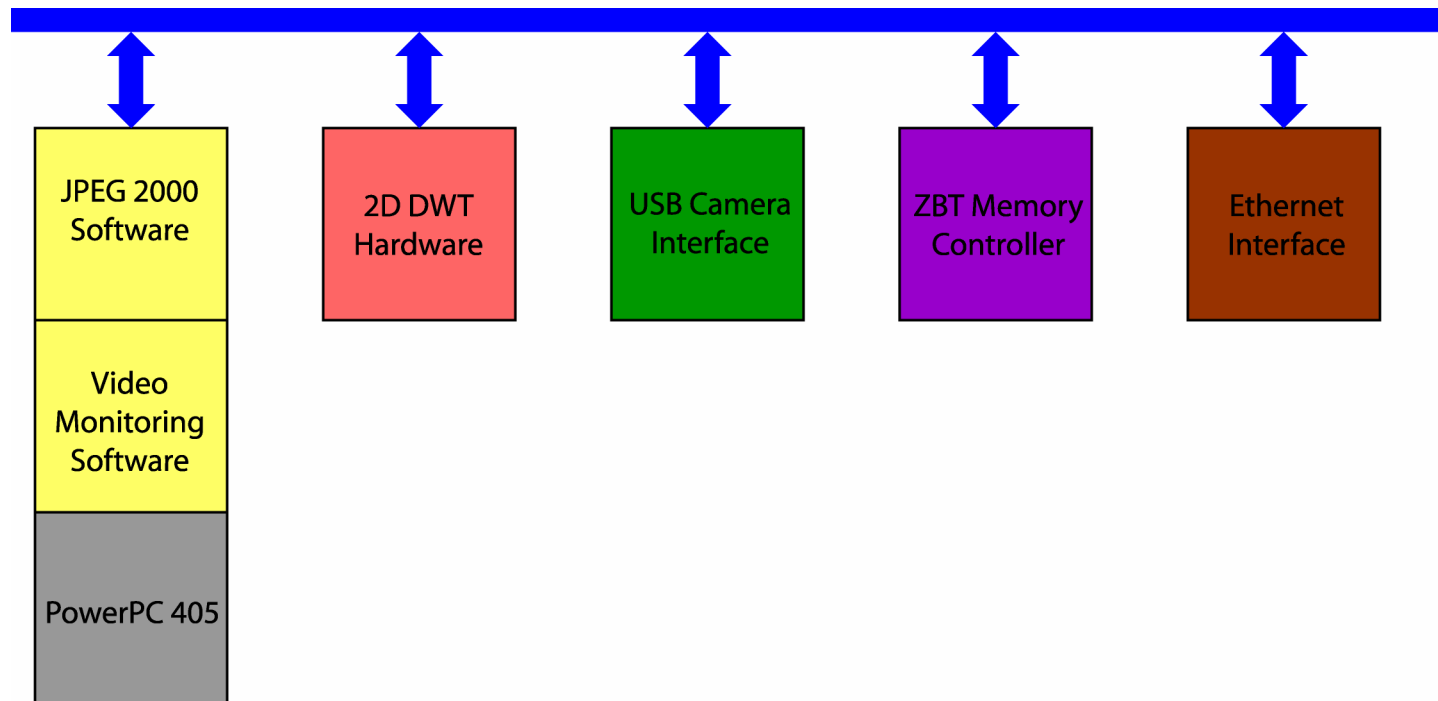




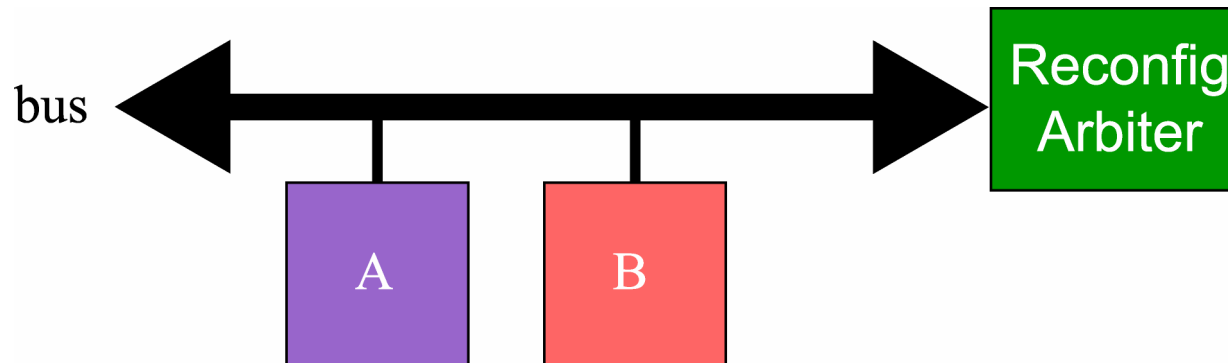
# JPEG2000 Platform



# Video Monitoring Application

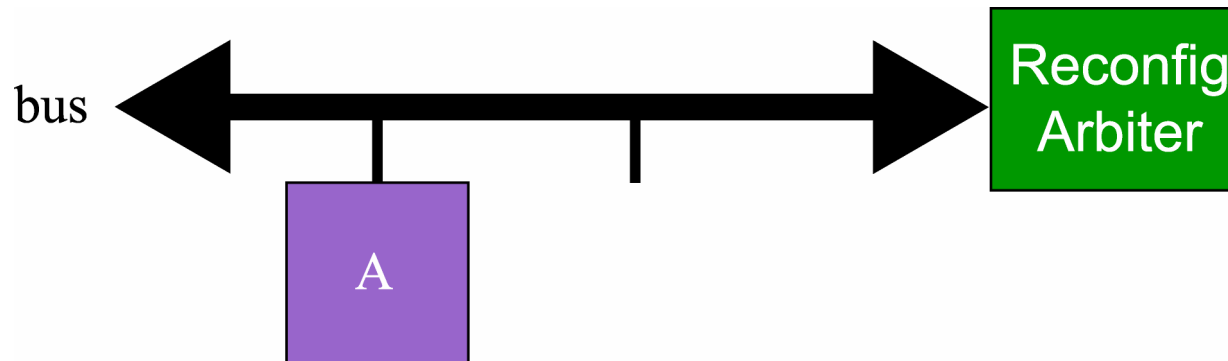


# Bus Based Reconfiguration

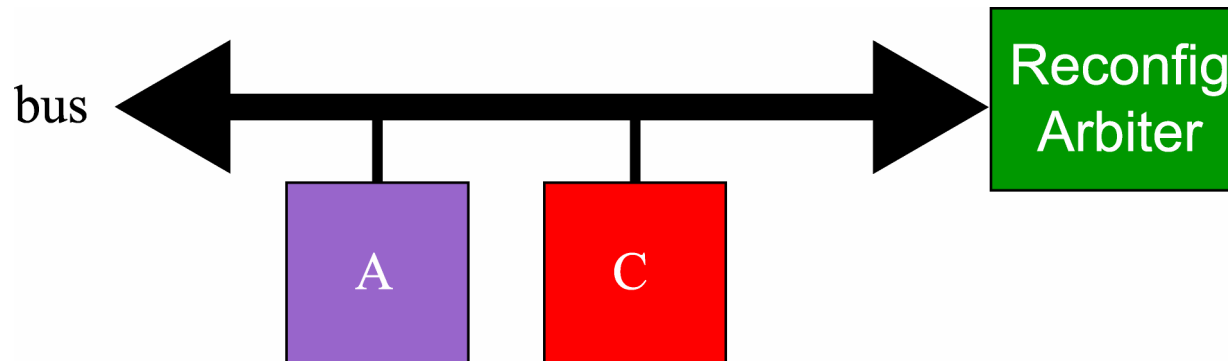




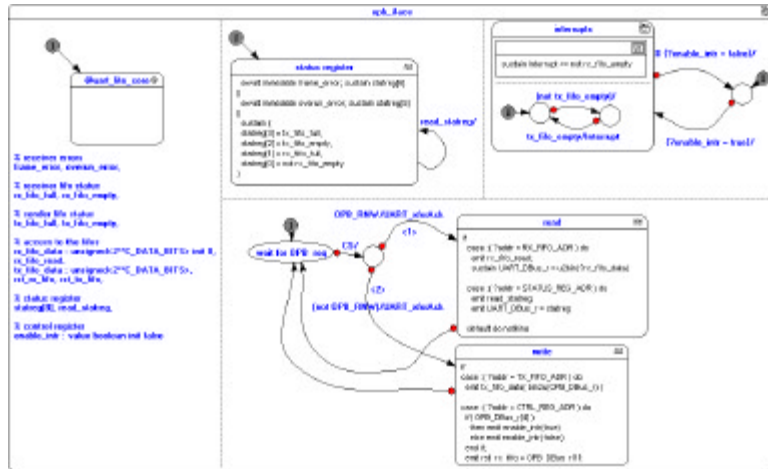
# Bus Based Reconfiguration



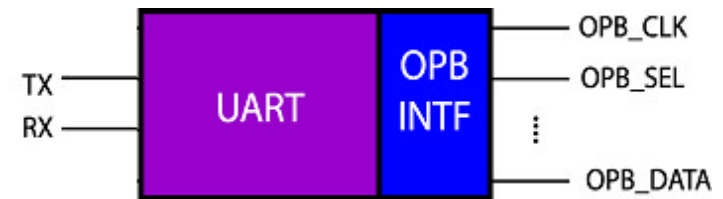
# Bus Based Reconfiguration



# Single Specification for Hardware and Software



HW/SW agnostic specification



VHDL, Verilog -> hardware implementation



```
void uart_device_driver ()  
{  
    .....  
}
```

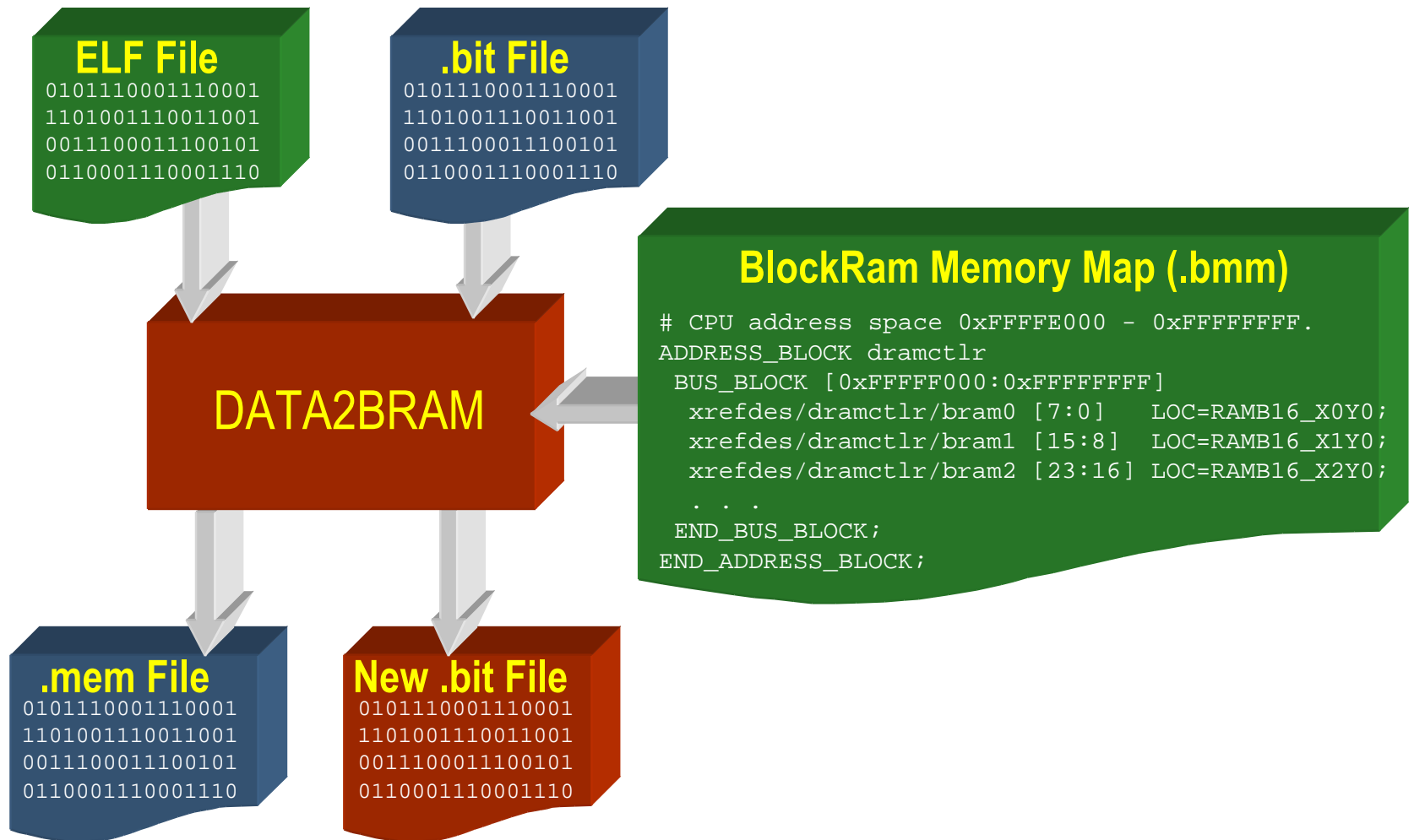
uart.c

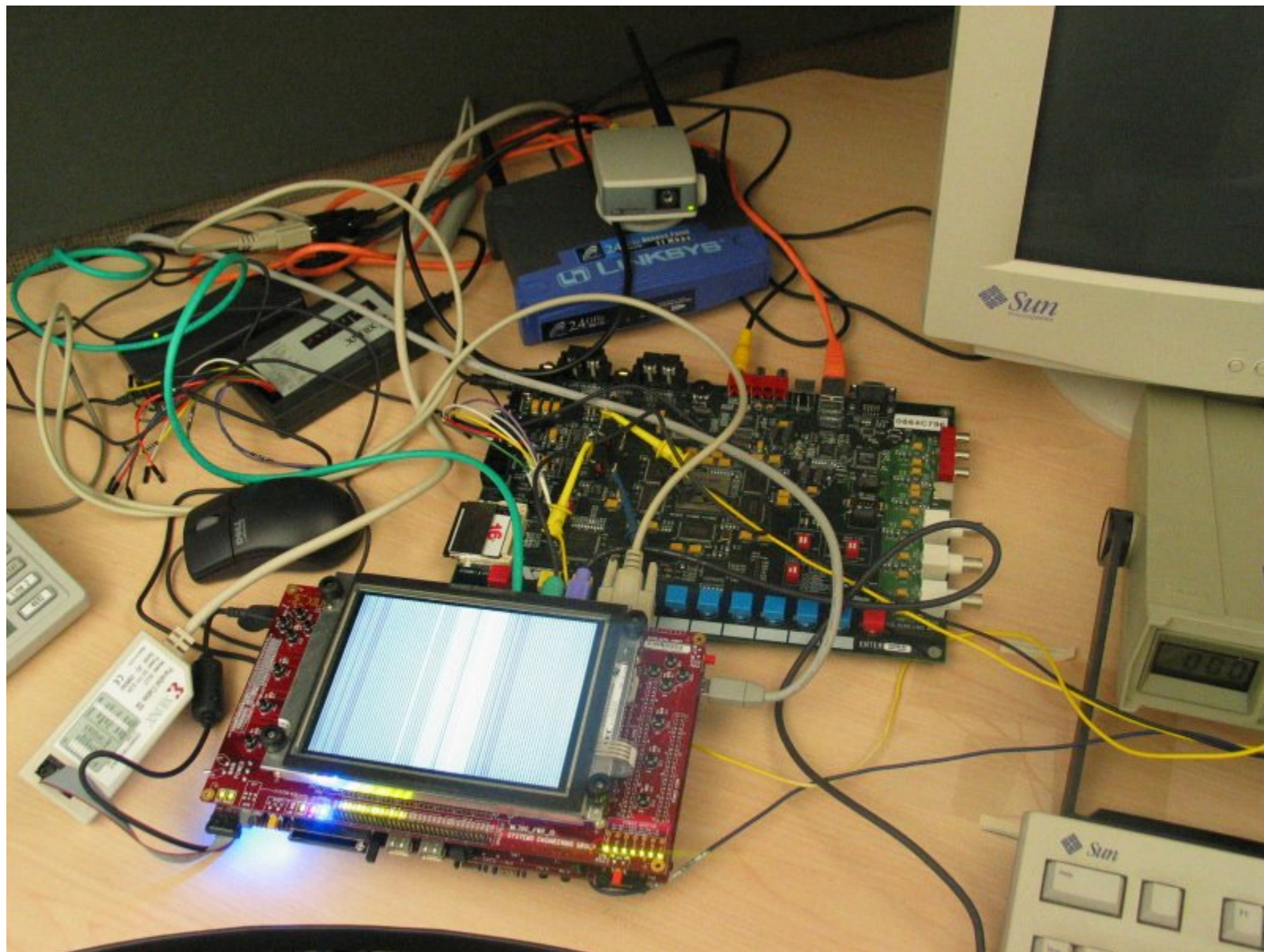
C -> software implementation





# Configuration

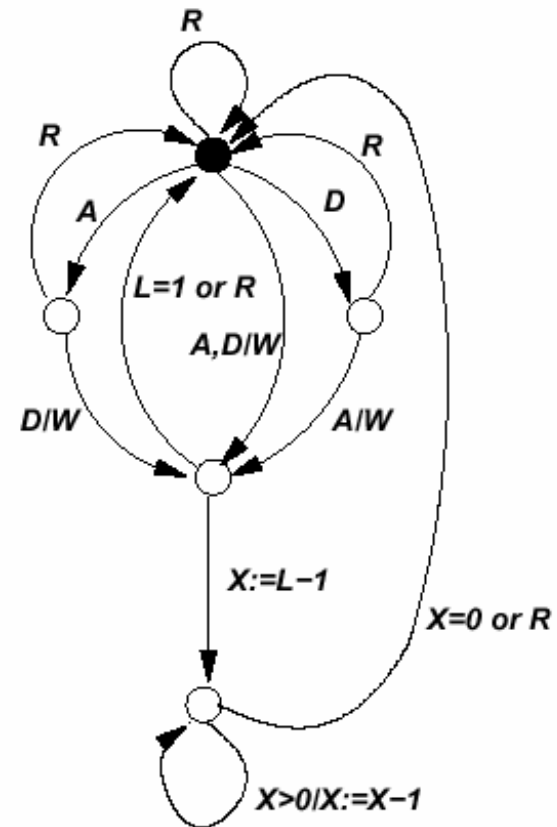






# FSM Specification

*Write to memory as soon as Addr and Data have arrived. Wait for memory Latency before iterating. Restart behavior each Replay.*

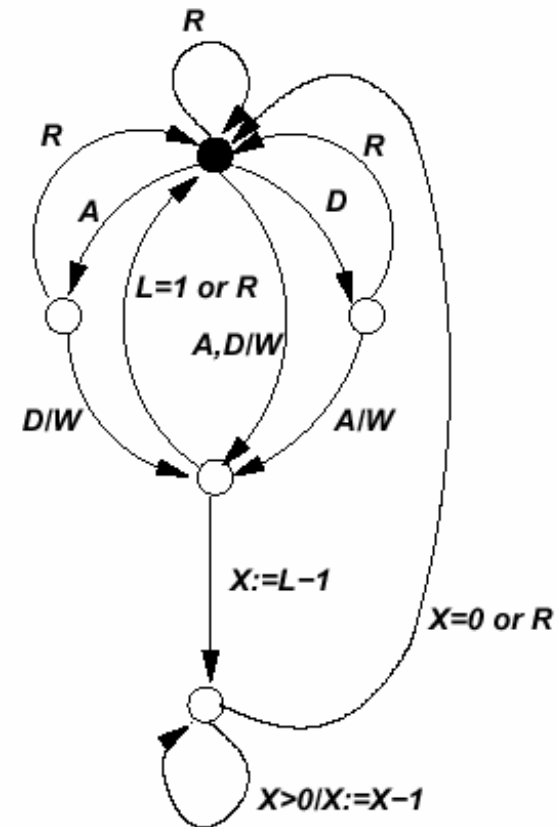


# Esterel Specification

*Write to memory as soon as Addr and Data have arrived. Wait for memory Latency before iterating. Restart behavior each Replay.*

```

loop
  abort
  { await Addr || await Data };
  emit Write(funcW(?Addr,?Data));
  await Latency tick
  when Replay
end loop
    
```



# FIFO Extract

```
{ // equation style
  sustain{
    ReadEmpty = Read and pre(FifoIsEmpty)
    FifoIsEmpty = if (?Entries = 0) }
||
  // imperative style
  every DeltaEntries do
    emit next ?Entries <= ?Entries
      + ?DeltaEntries;
  end every
}
```



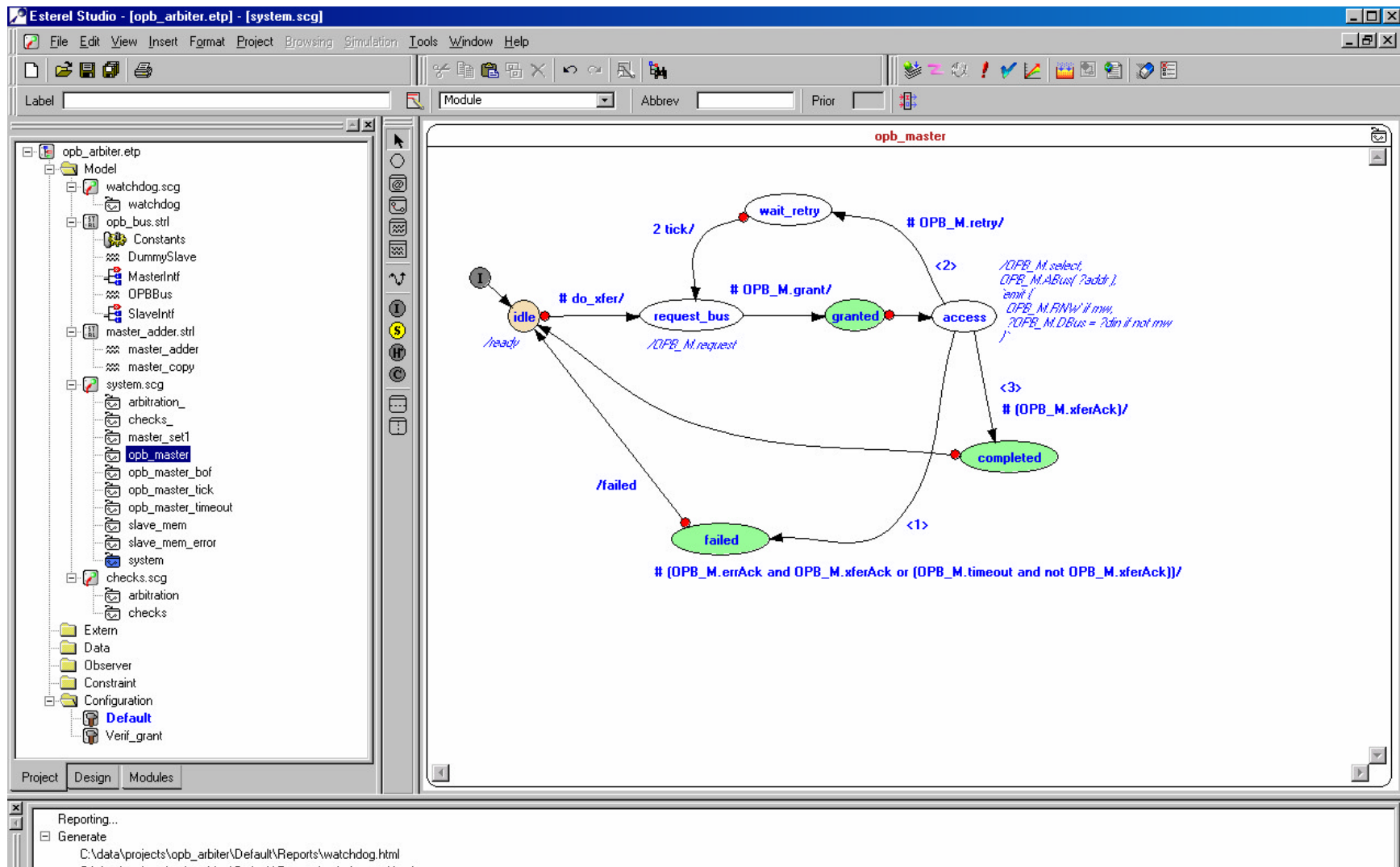
# Concurrent Loops

```
for i < 2 dopar
  sustain {
    FifoEmpty[i] = if (pre(?Entries) = i)
    FifoFull[i]  = if (pre(?Entries) =
                      Size - i)
  }
end for
```

# Orthogonality

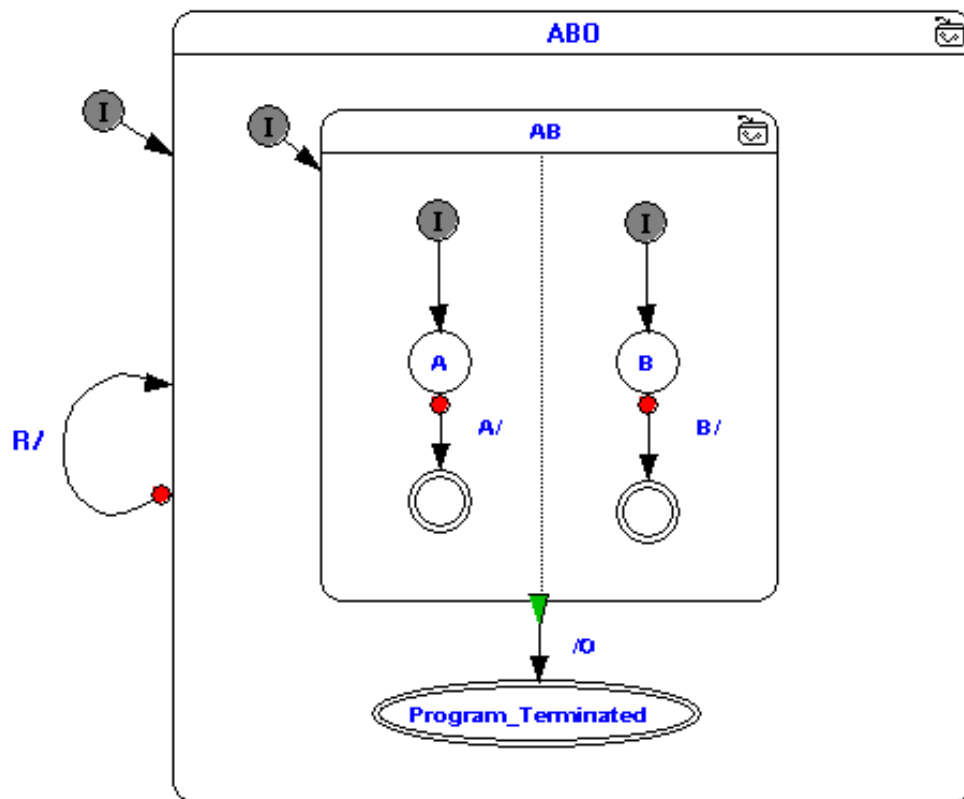
- Orthogonal language constructs for:
  - Sequencing
  - Concurrency
  - Waiting
  - Pre-emption
- Freely mixable at any level.
- “Things are only written once.” Gérard Berry.

# Esterel Studio



# Creating design

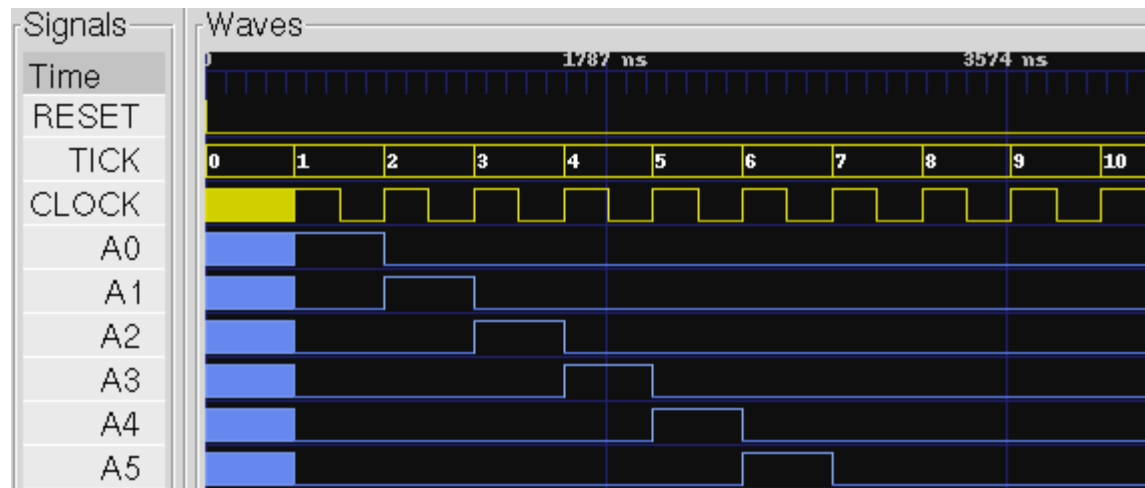
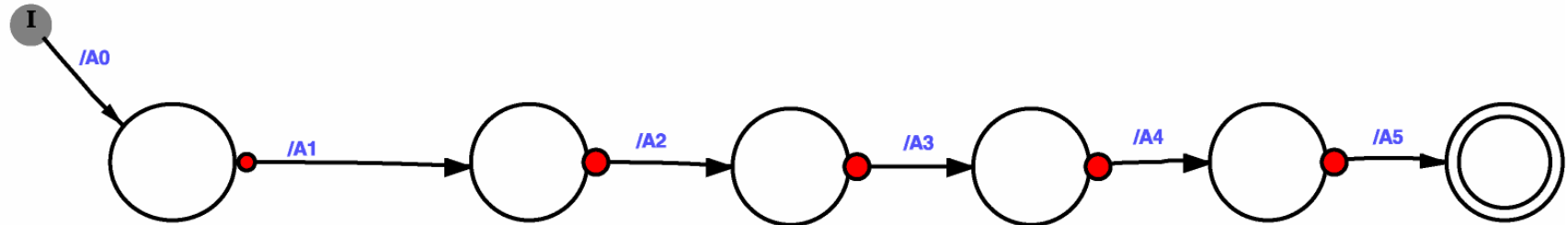
Via Safe State Machines

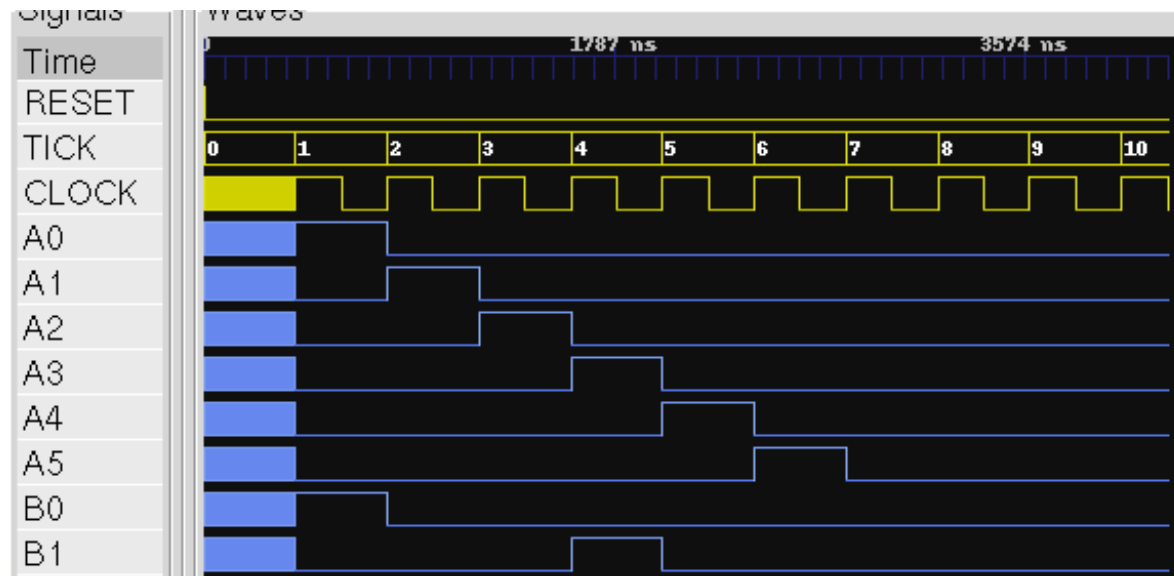
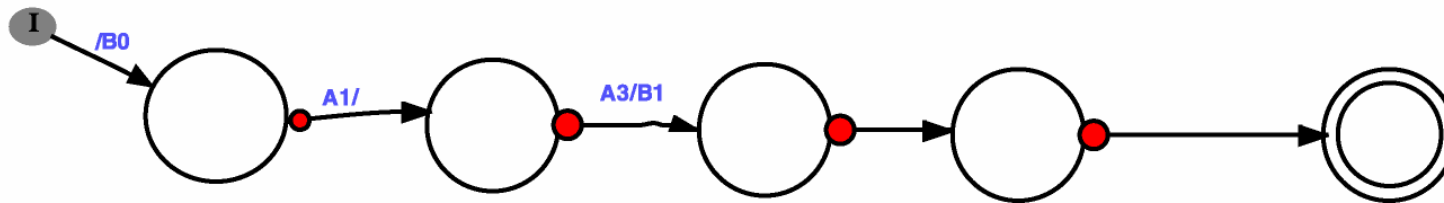
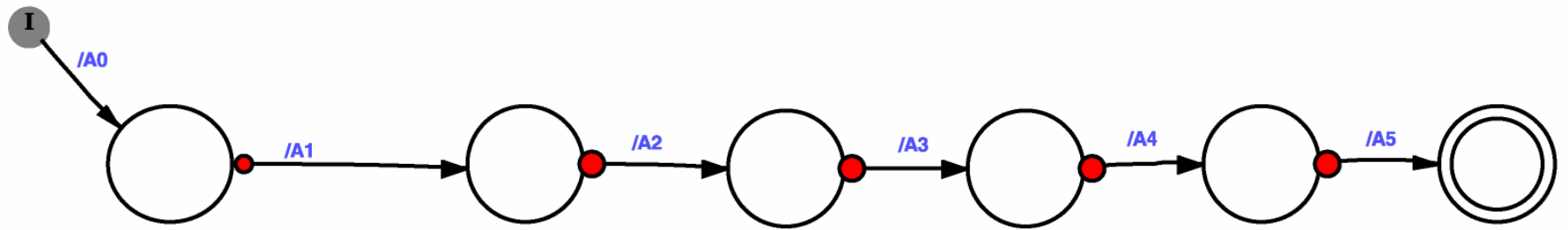


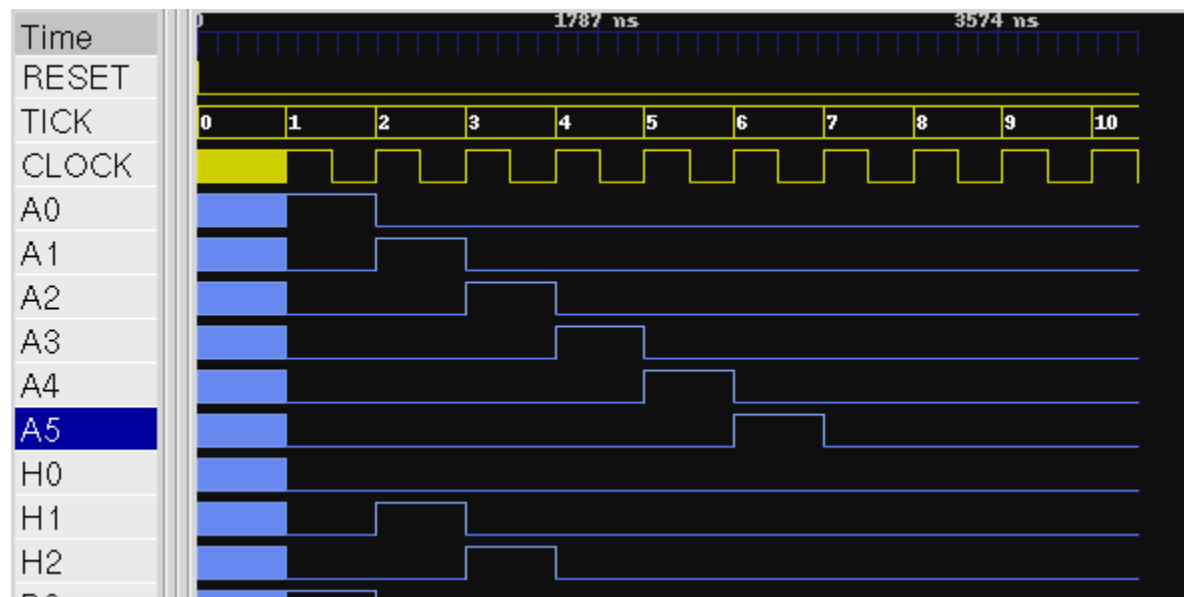
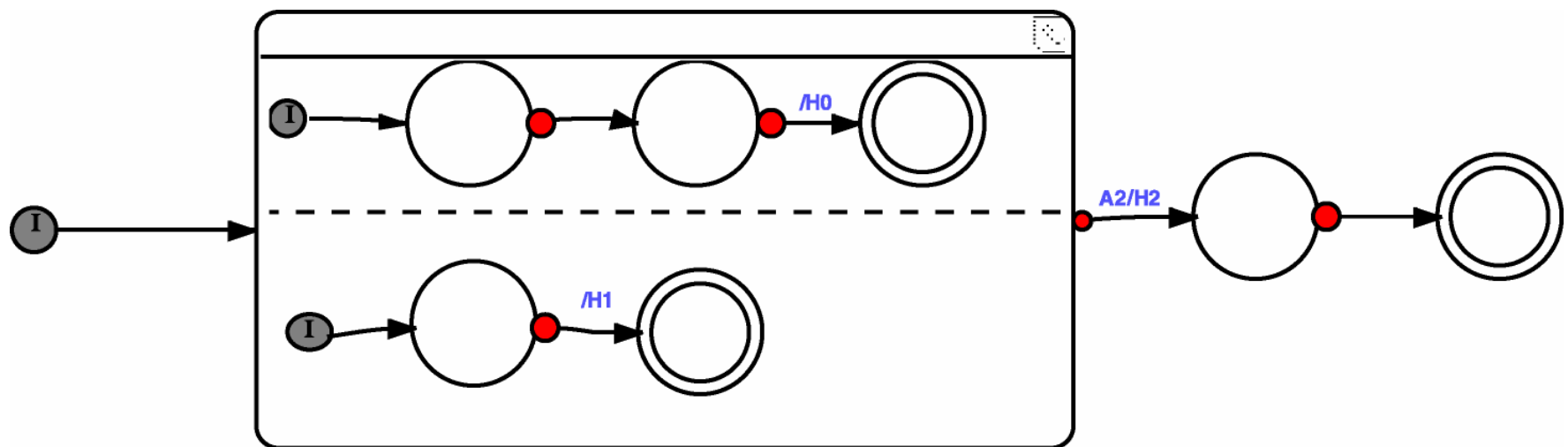
Via Esterel code

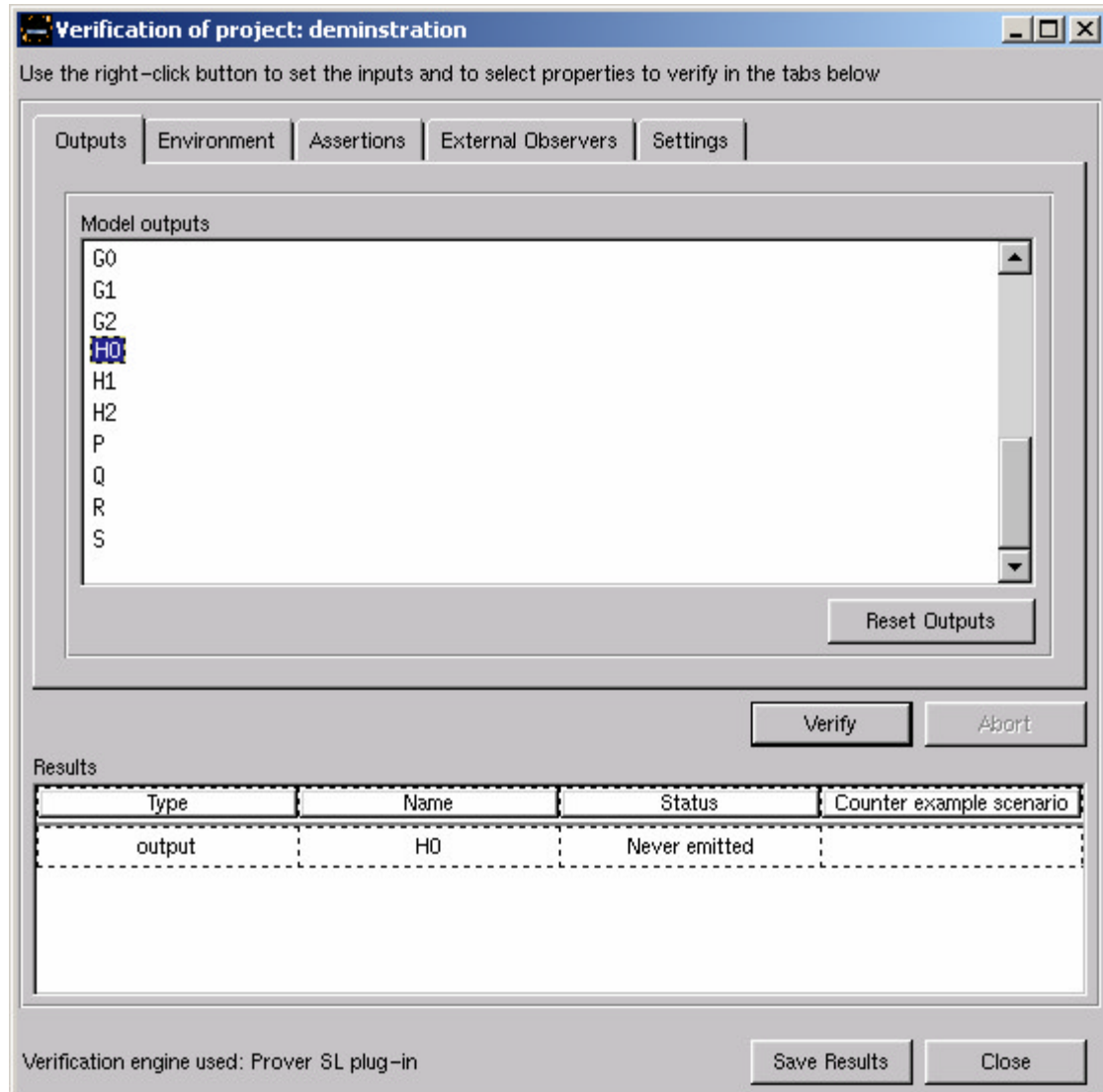
```
loop
  [ await A || await B ];
  emit O
each R
```





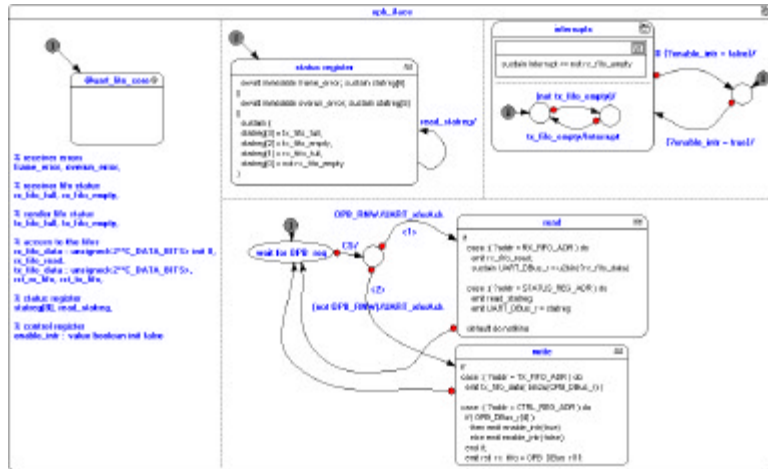




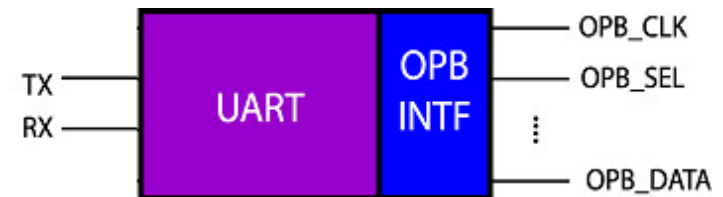




# Code generation



Esterel design



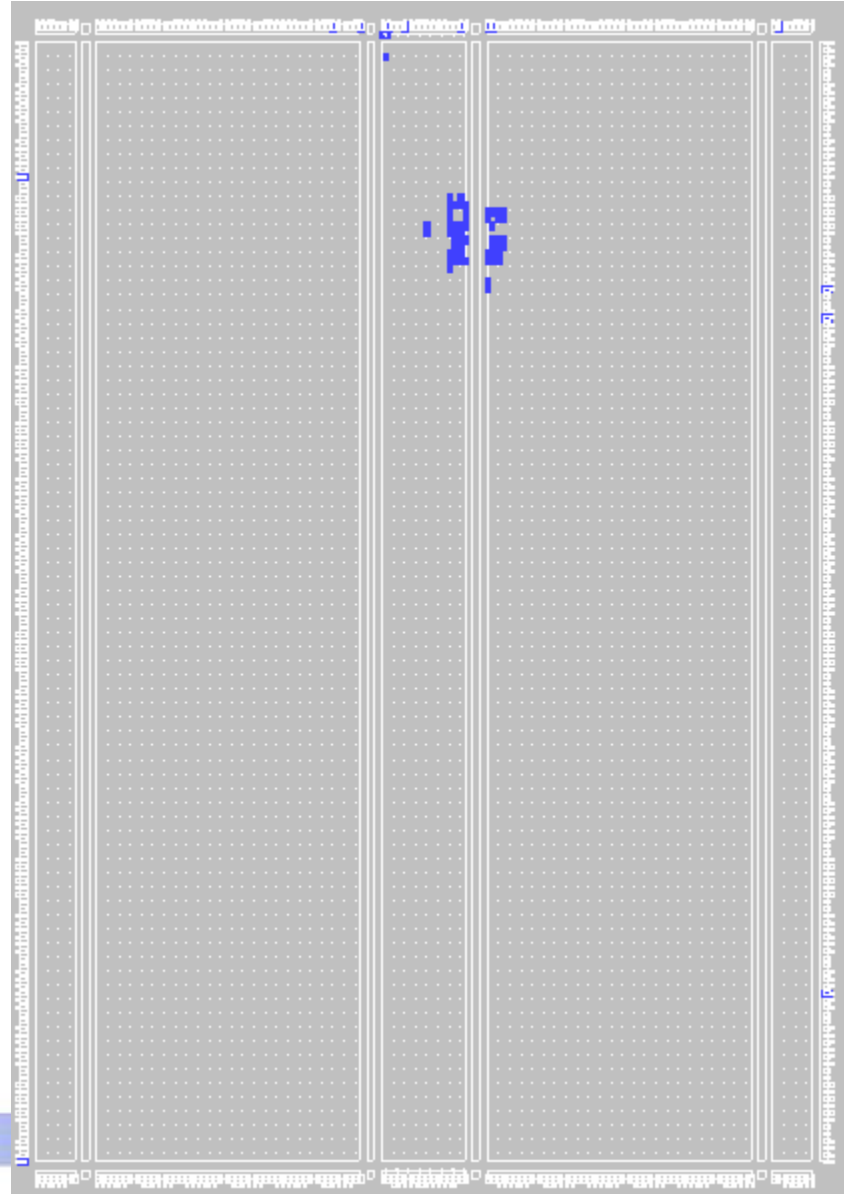
VHDL, Verilog -> hardware implementation

```
void uart_device_driver ()  
{  
    .....  
}
```

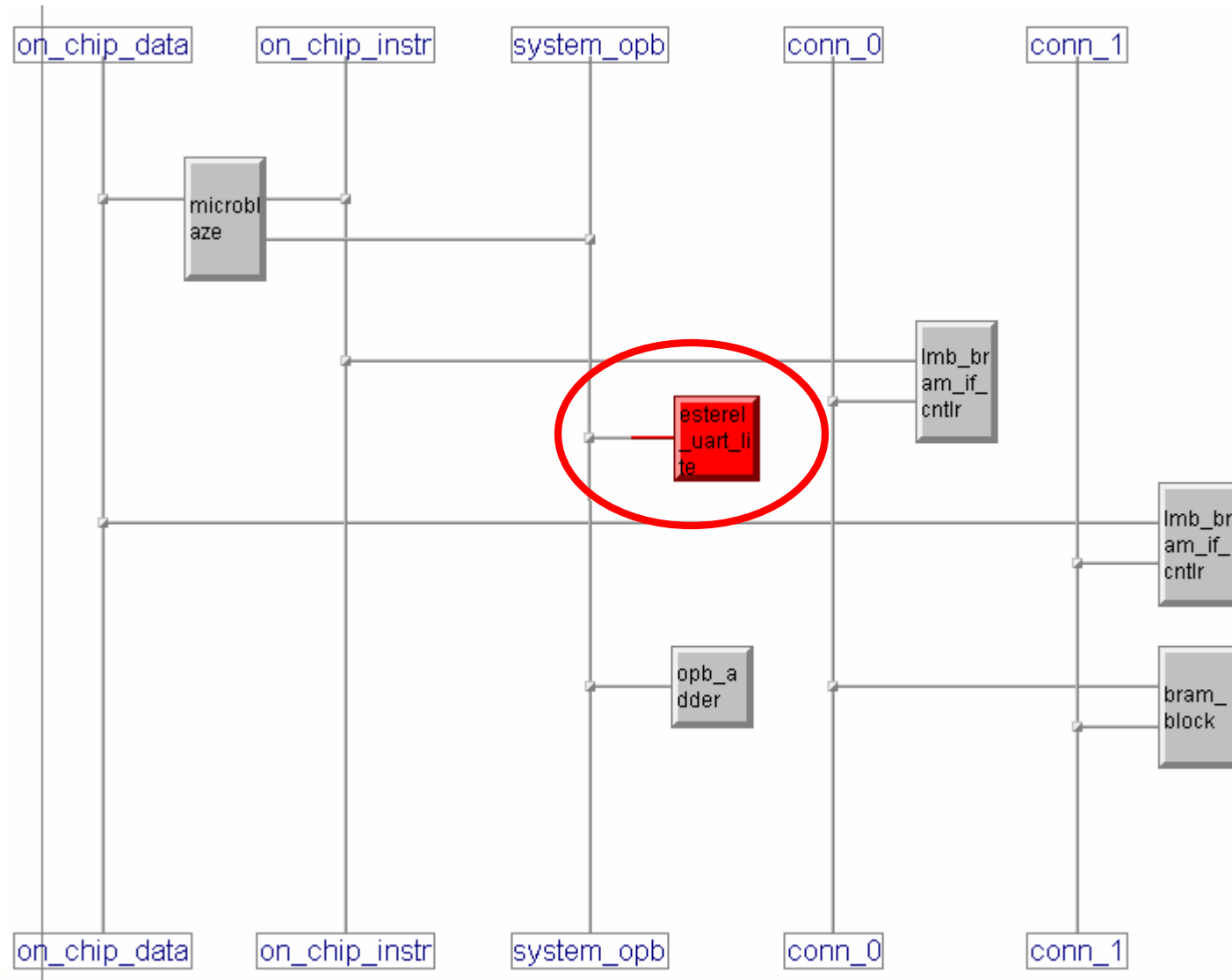
uart.c

C -> software implementation

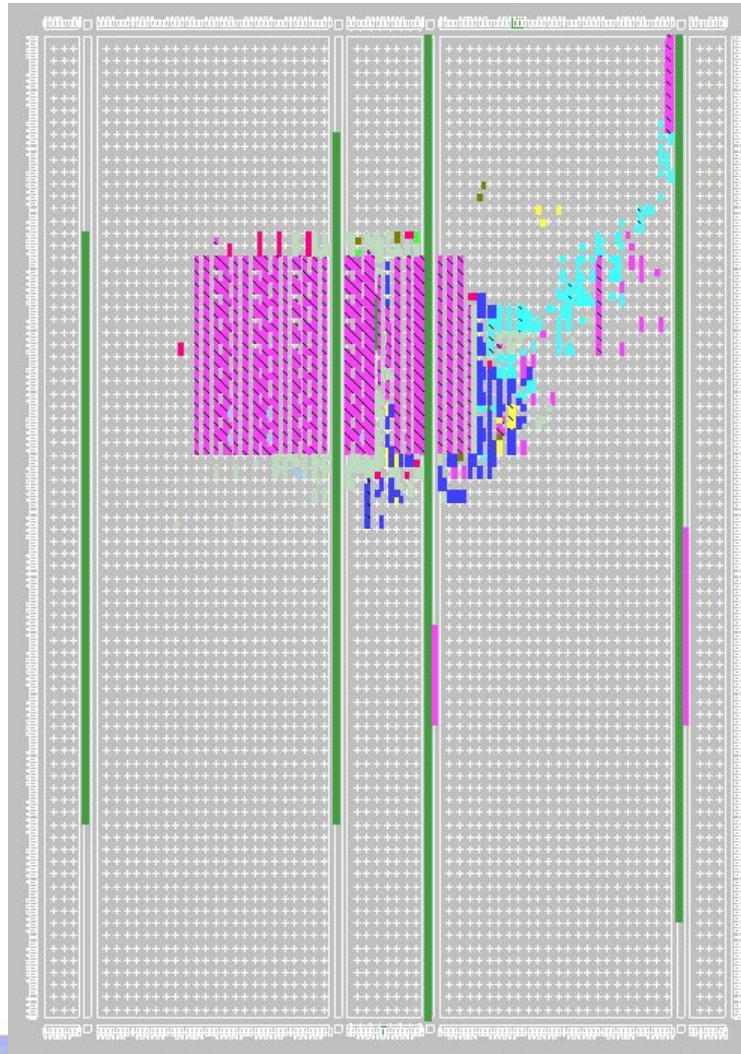
# Hardware UART XC2V1000



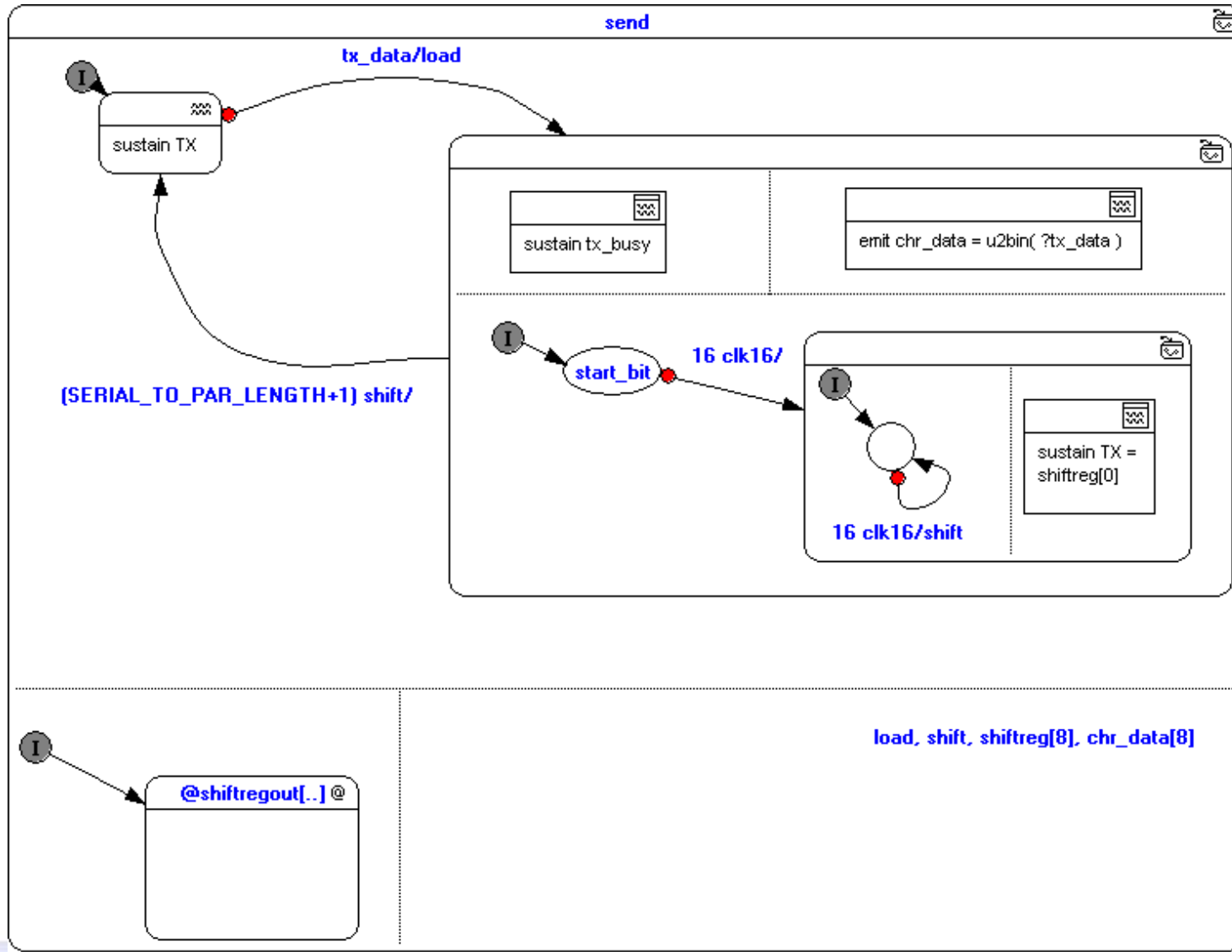
# Direct use in SoC



# Soft UART MicroBlaze XC2V1000

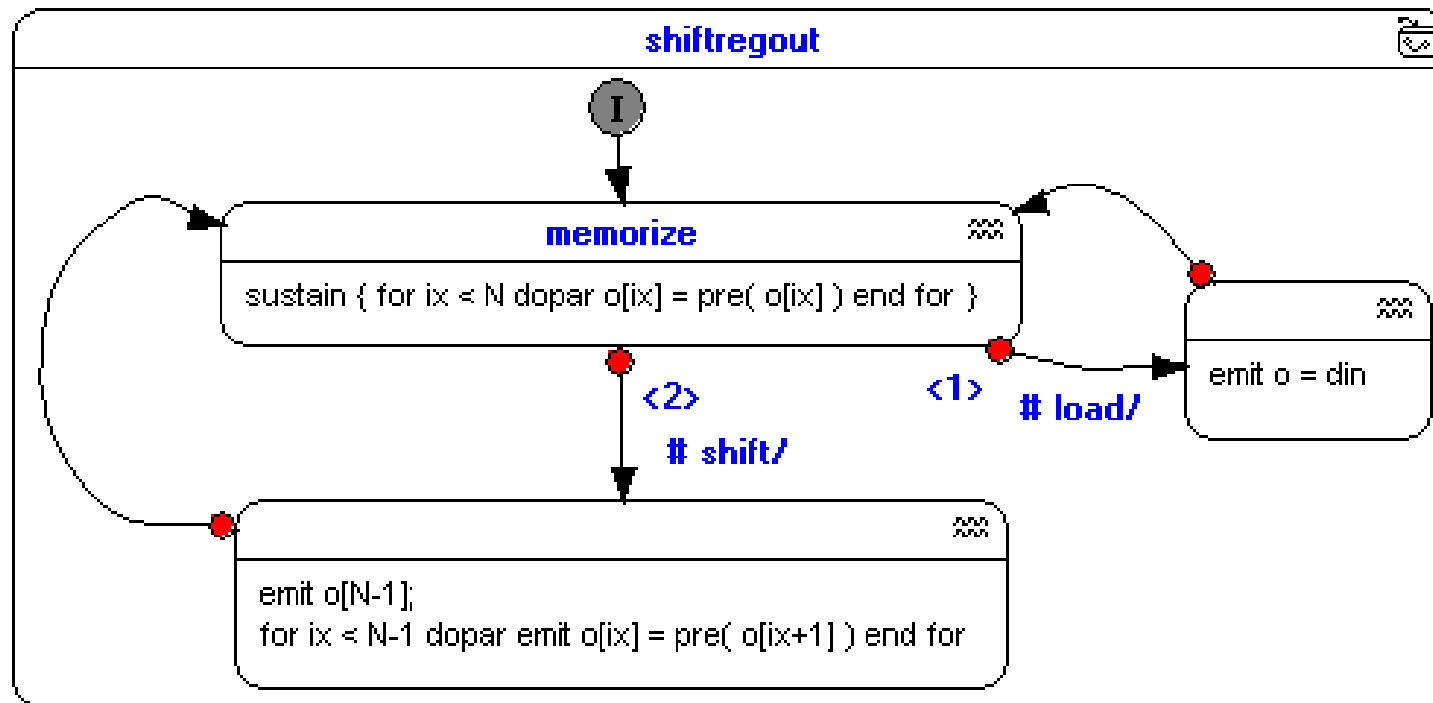


# sender

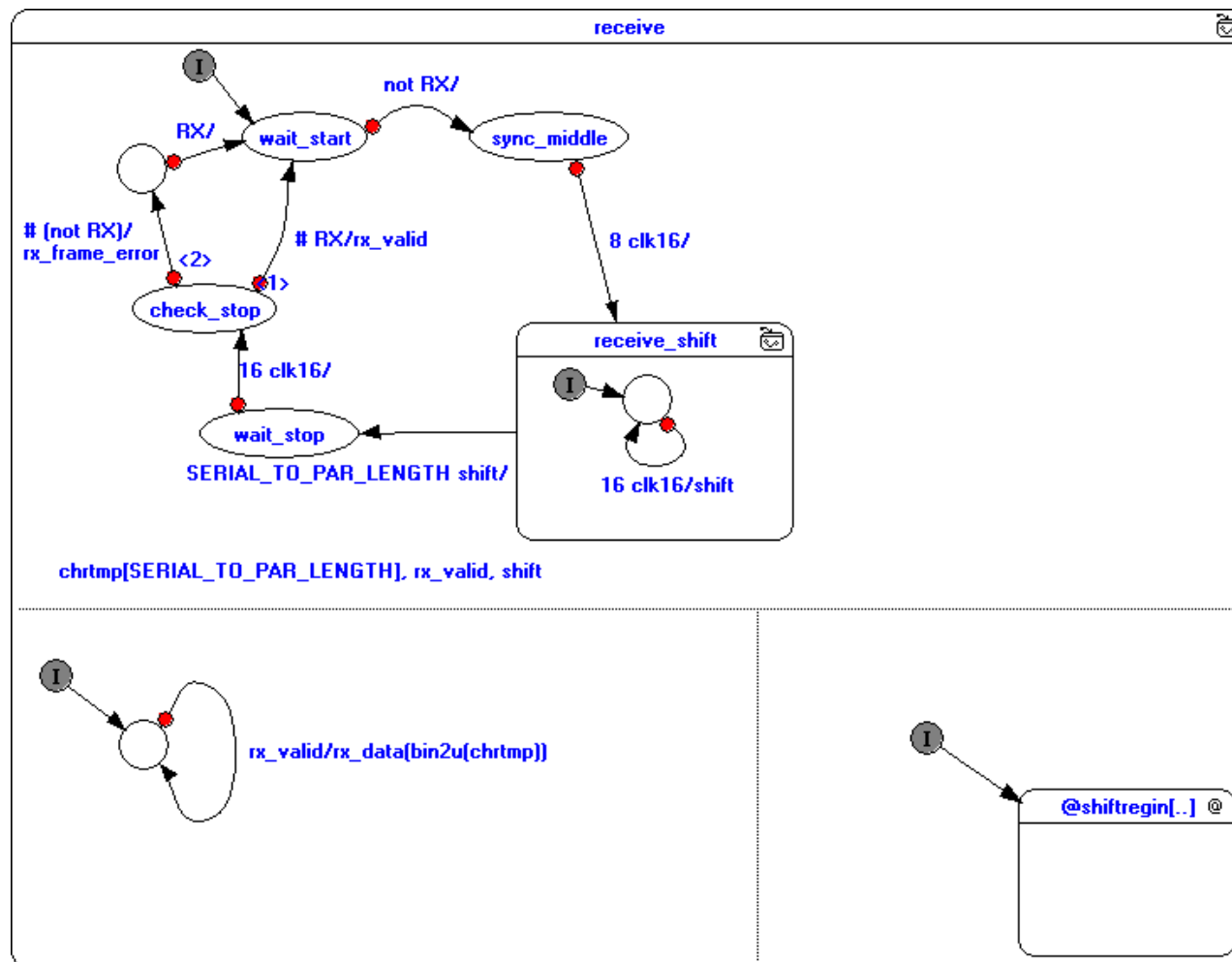




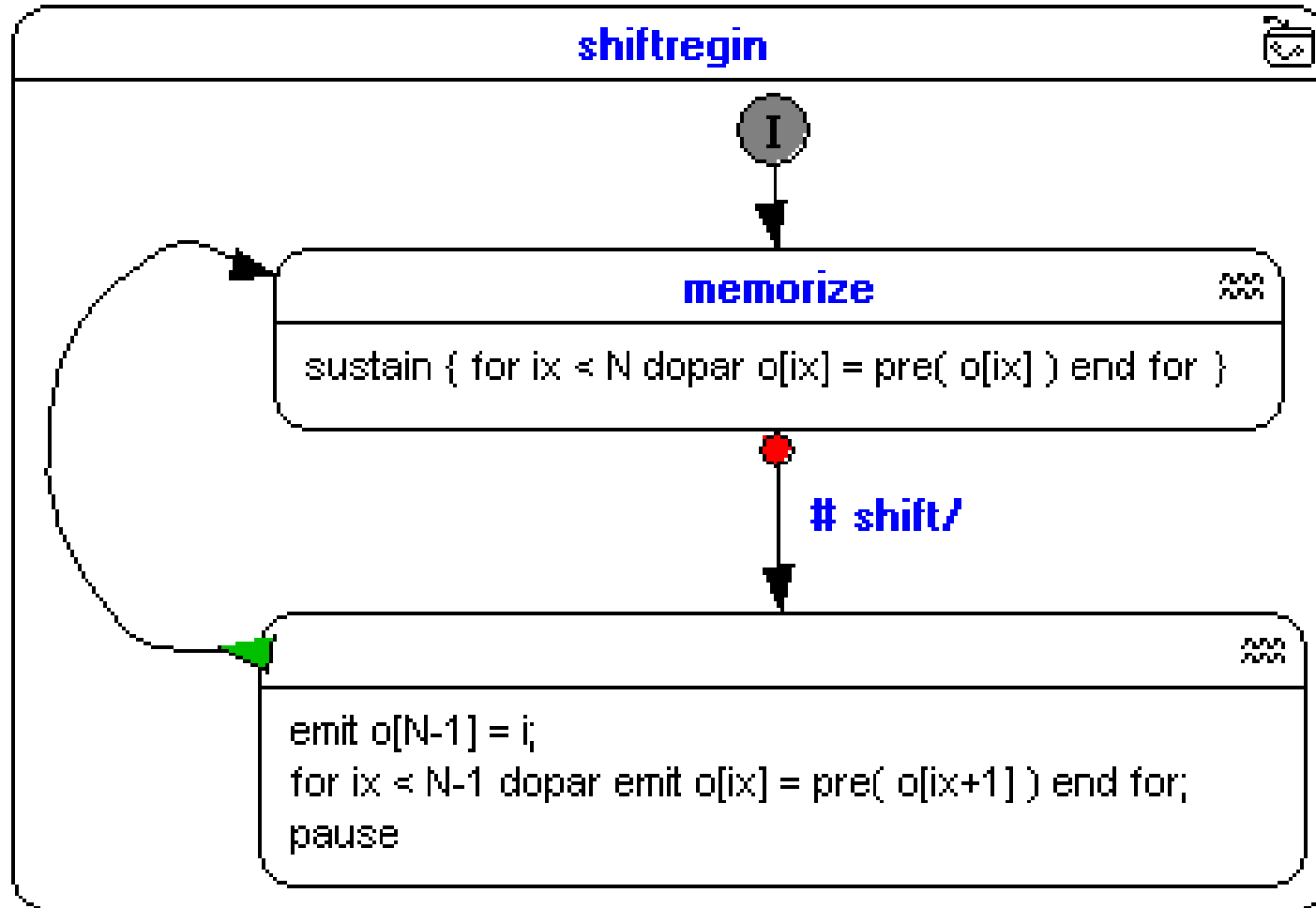
# parallel to serial shift



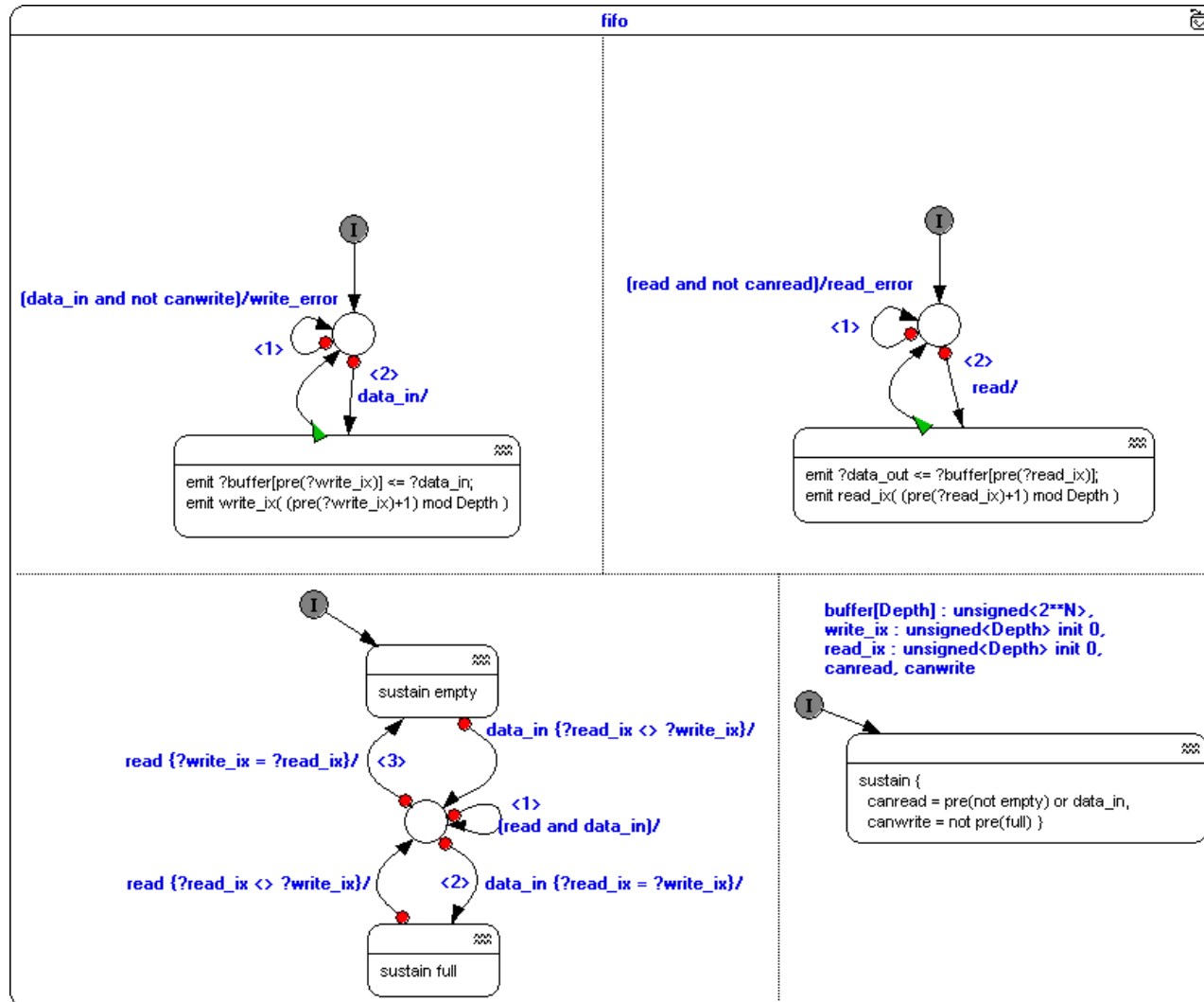
# receive



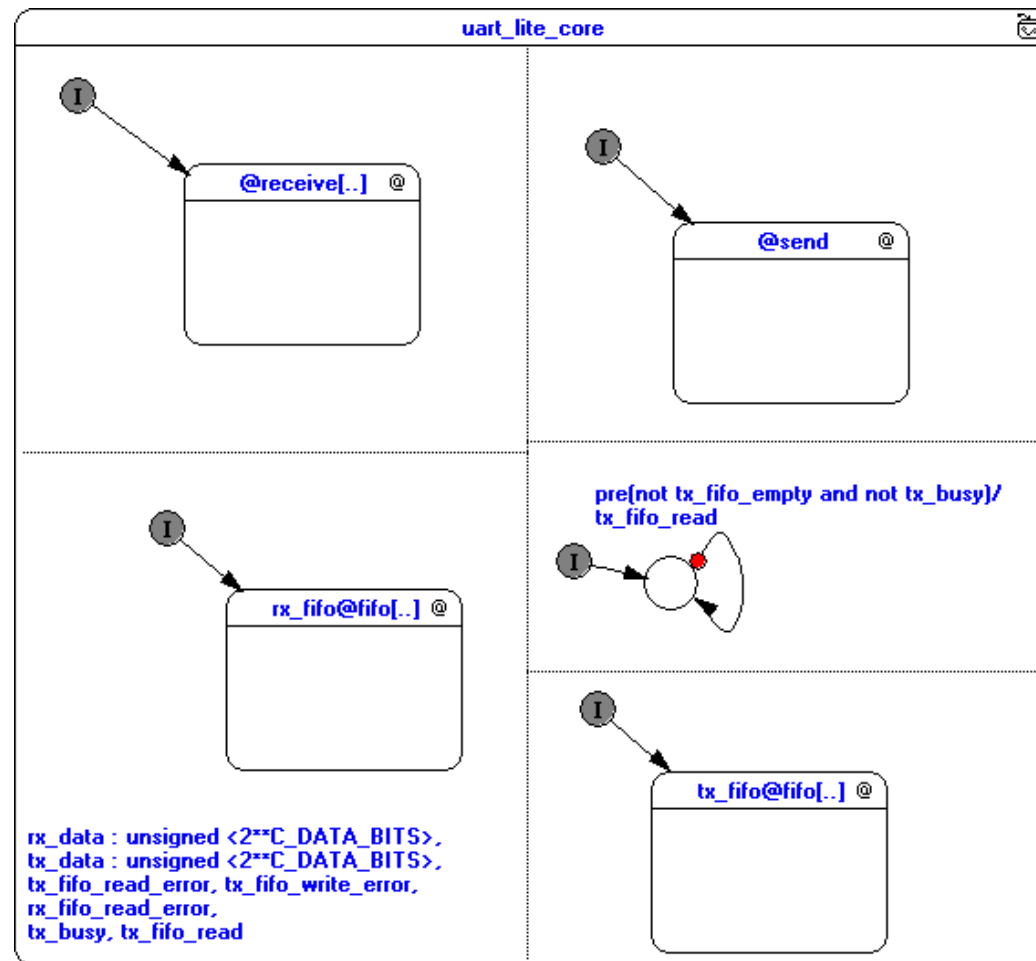
# serial to parallel



# FIFO

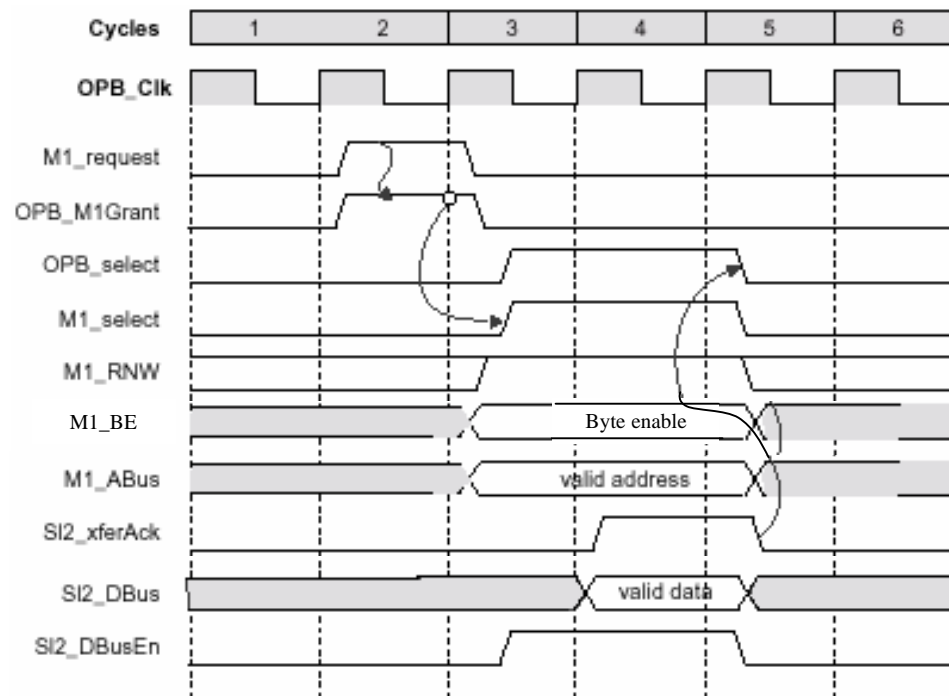


# UART without bus interface

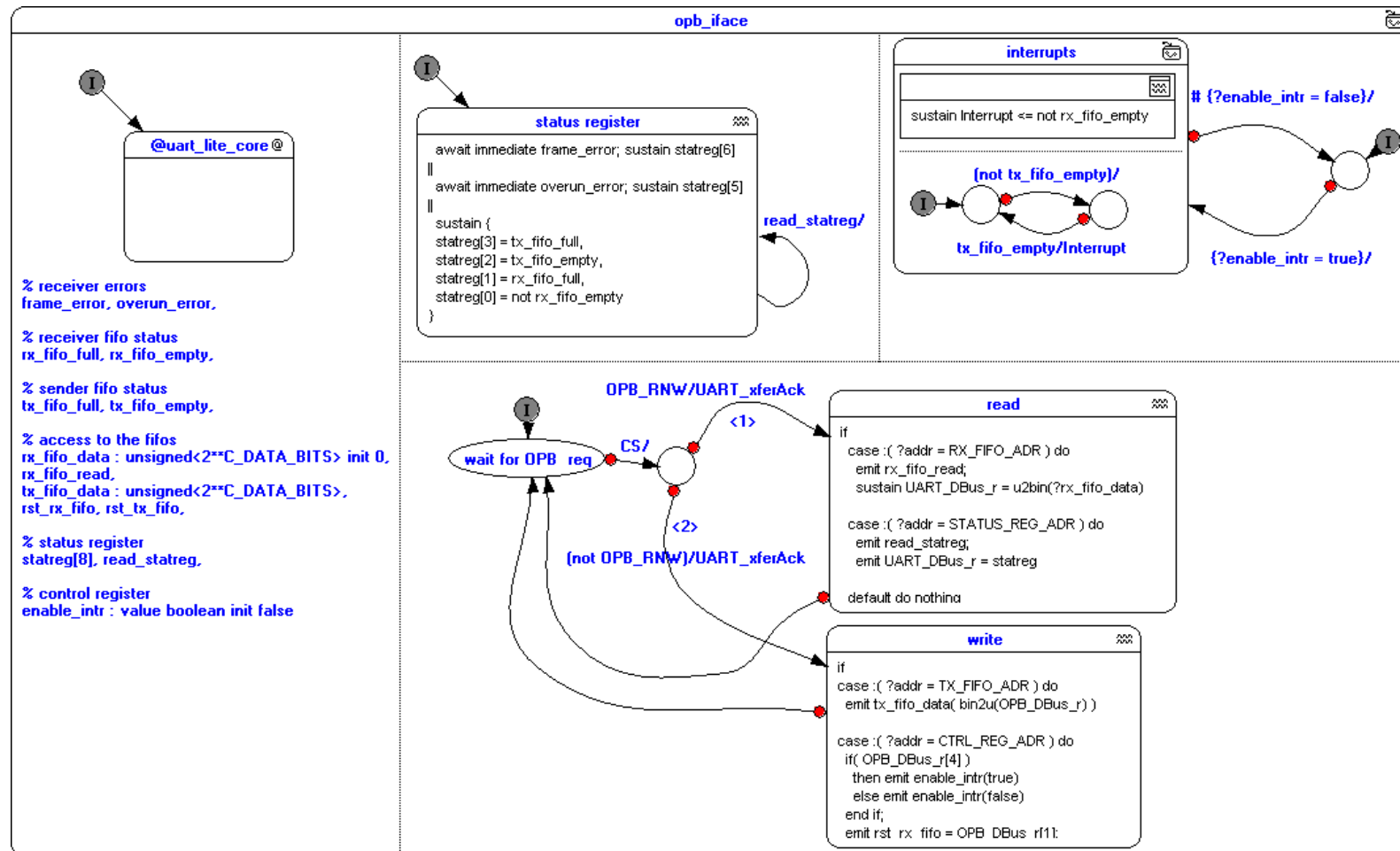




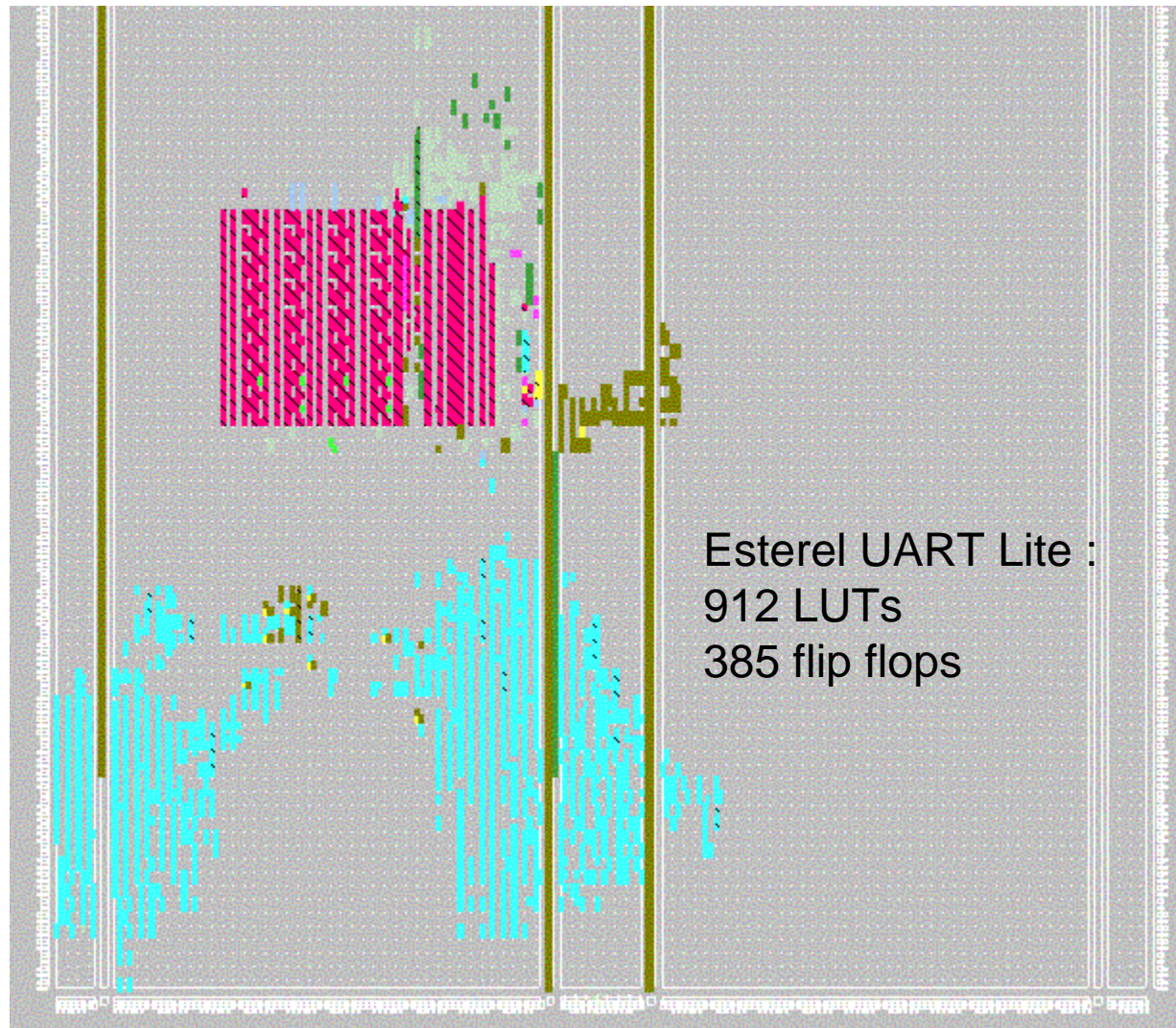
# OPB Protocol



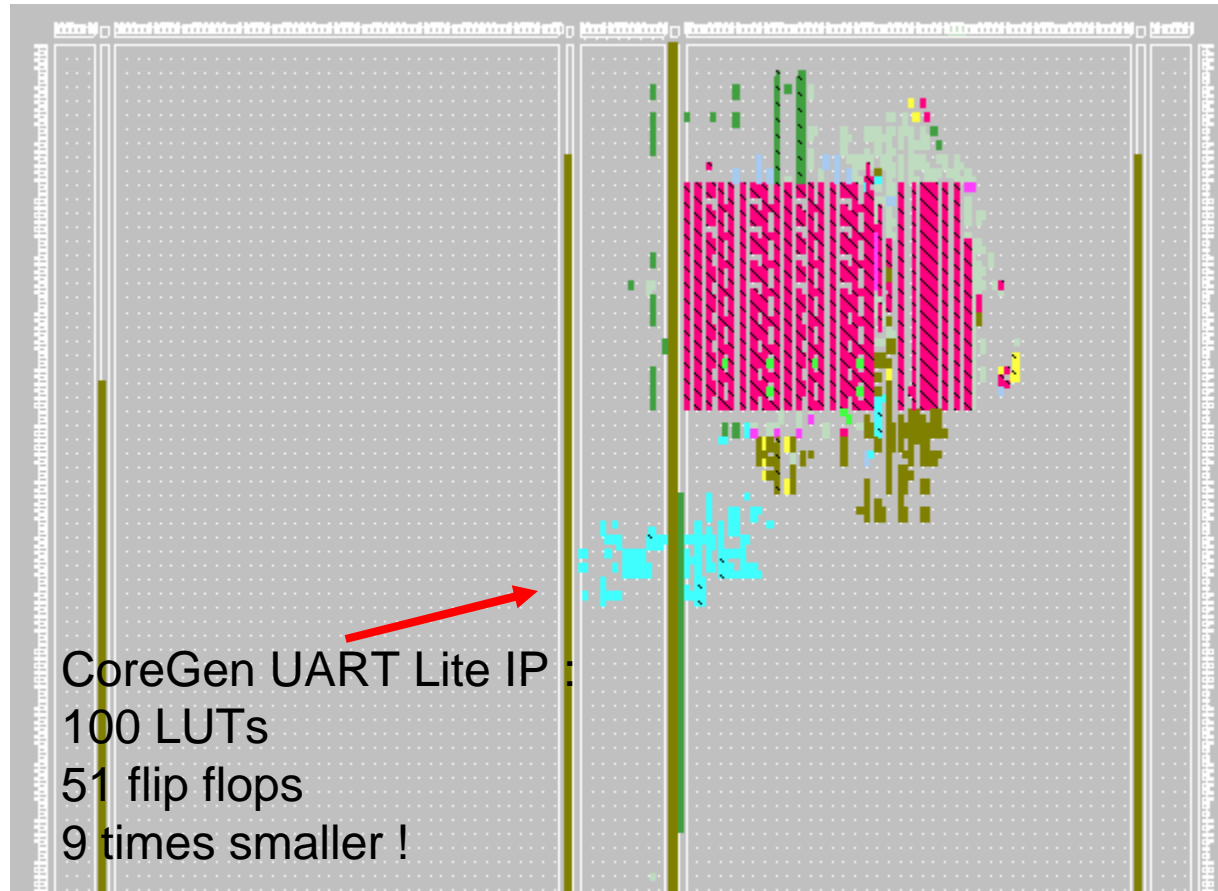
# UART with OPB Interface



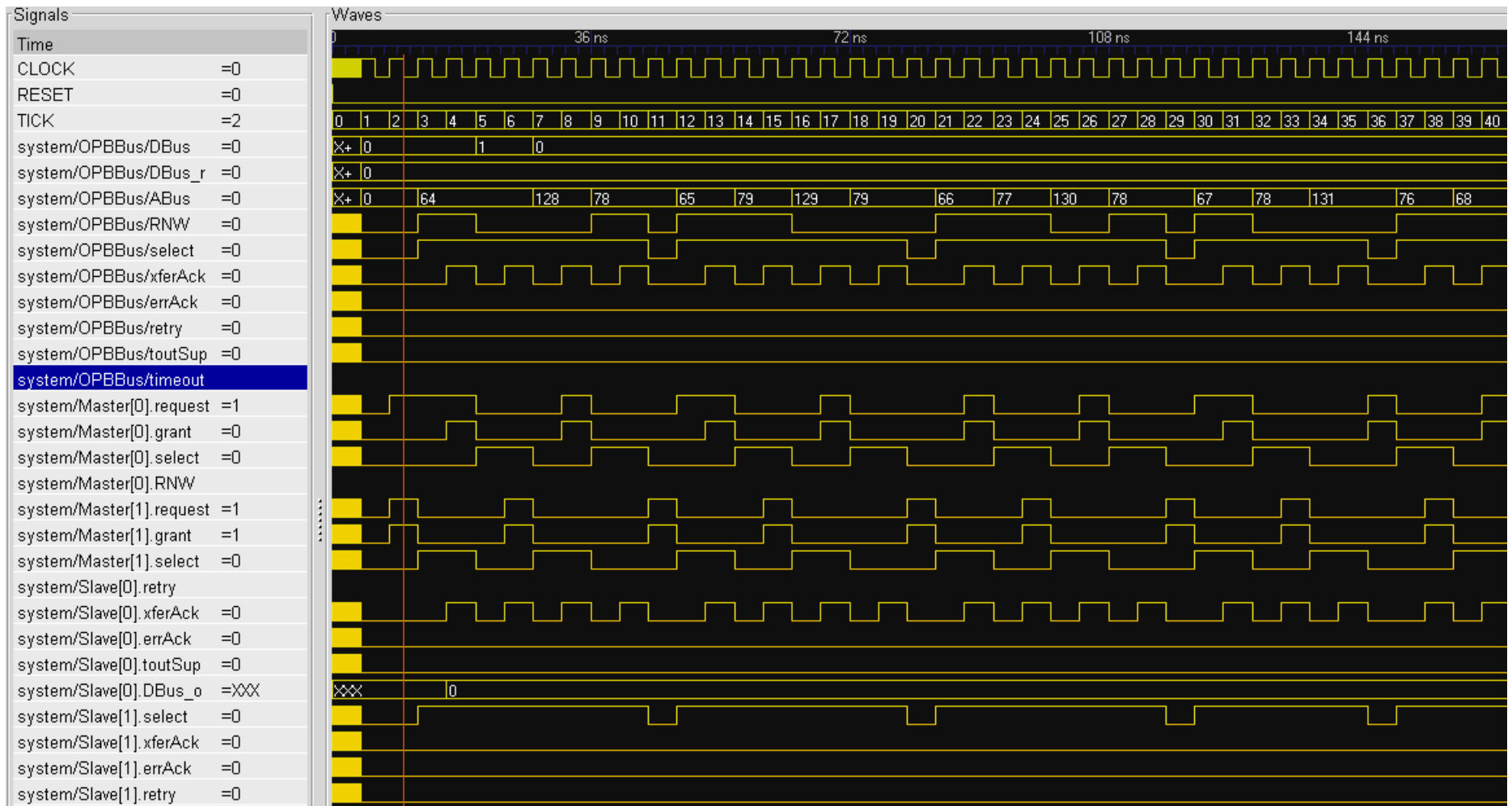
# Generated circuit



# Comparison with Original CoreGen IP

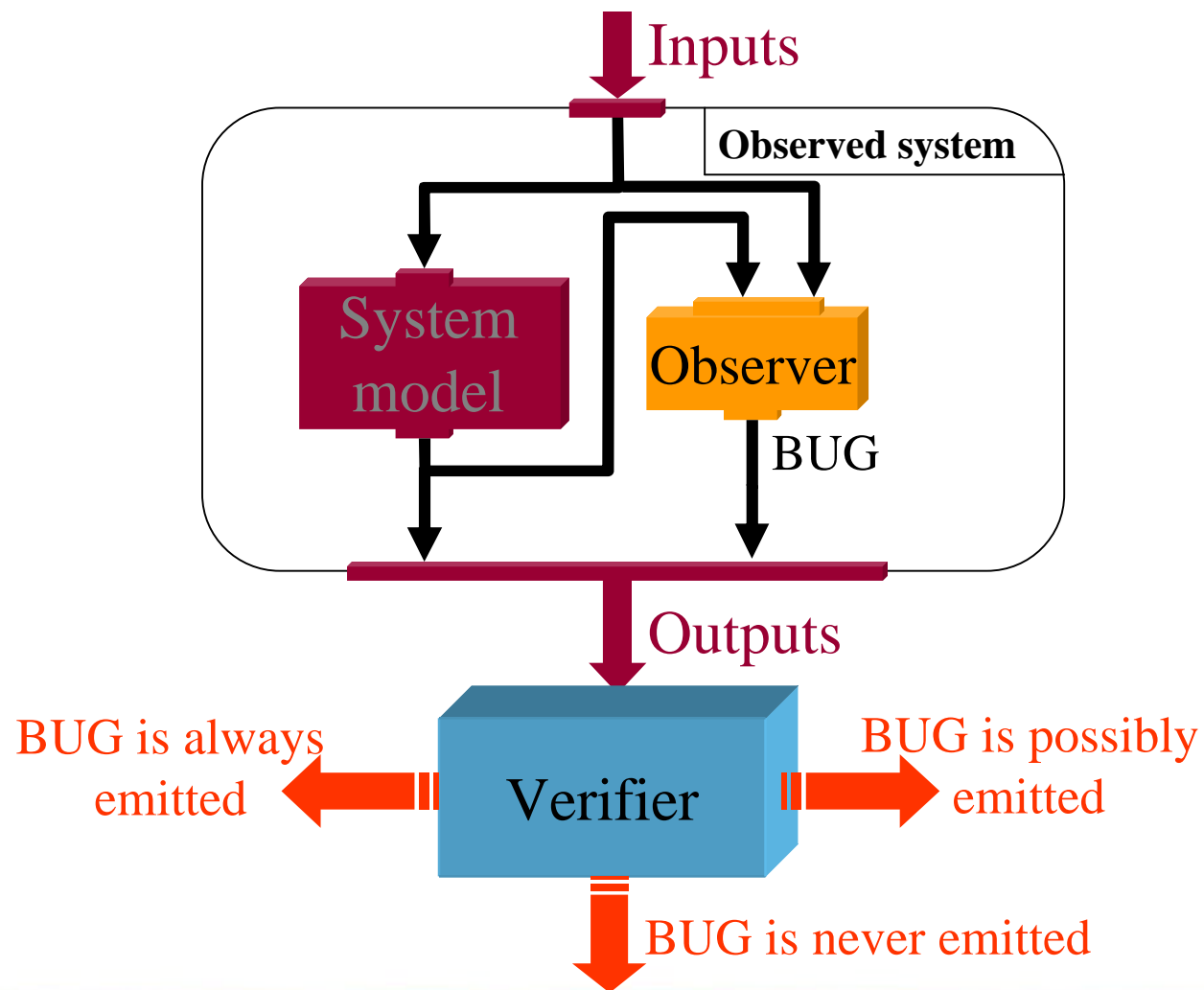


# Verification by simulation



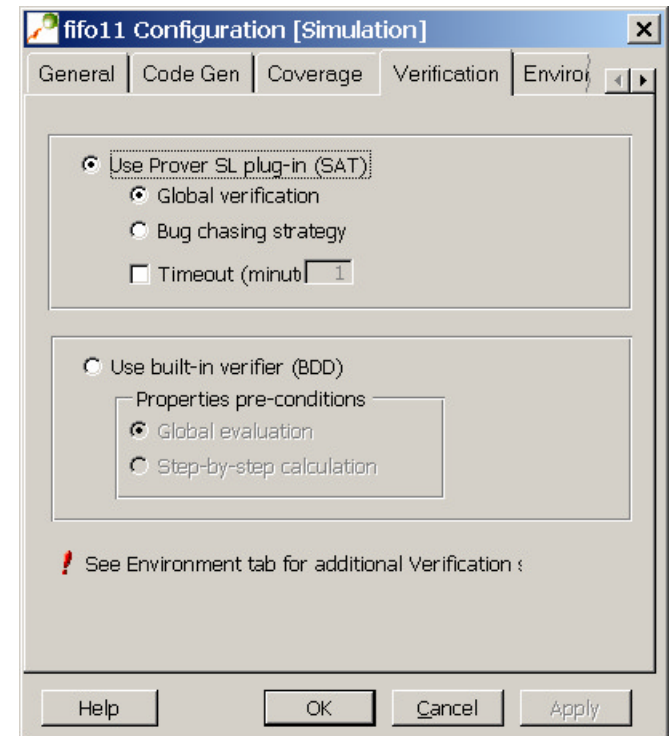


# Verification with Observers



# Verification engines

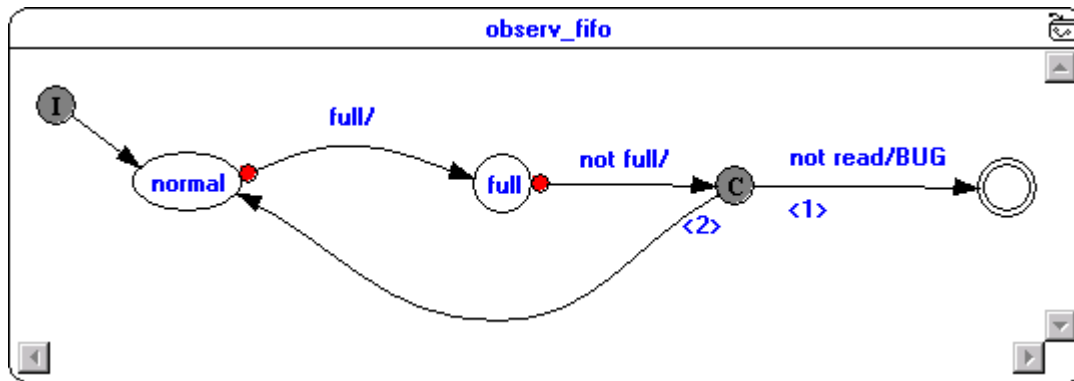
- 2 proof engines available inside Esterel Studio
  - Built-in verifier : TiGer
    - BDD technique
  - Prover Plug-in
    - SAT technique



# Formal verification

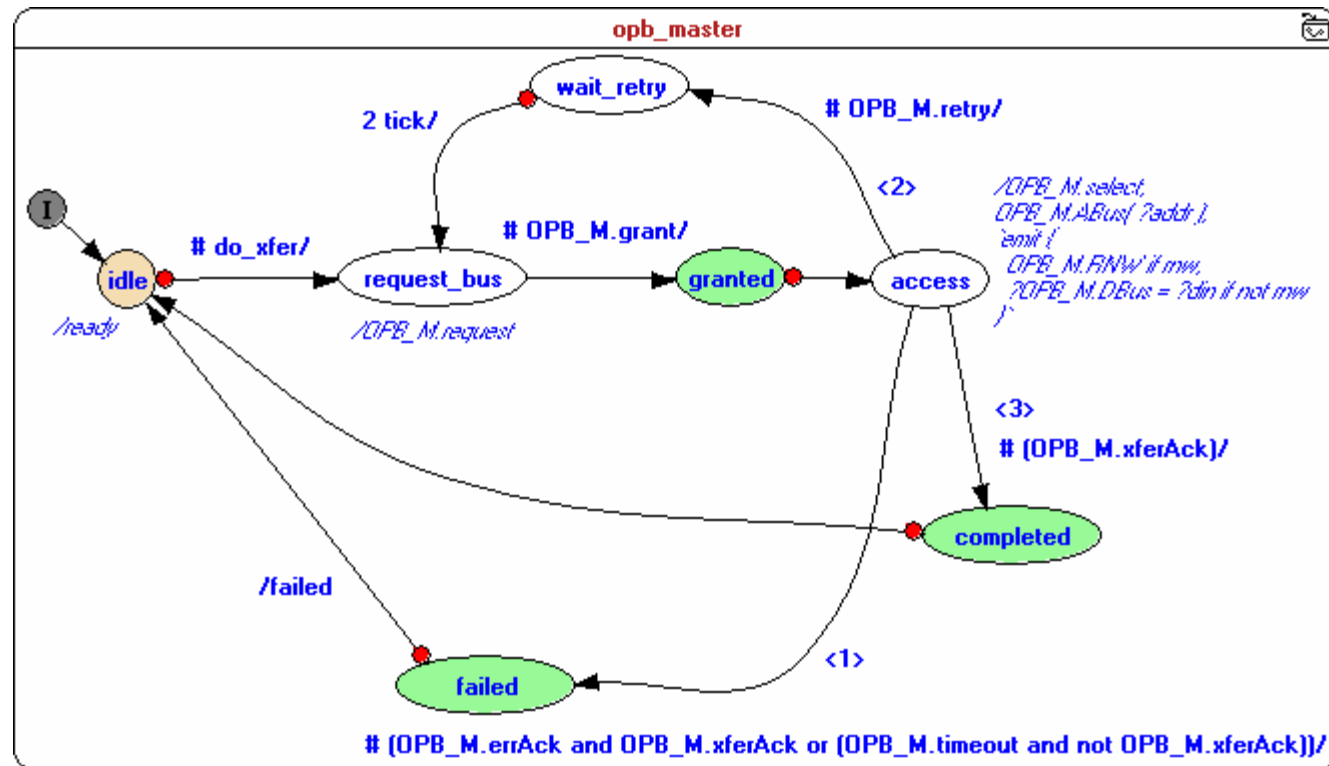
Of the FIFO :

proving that only a read access can make it exit the “full” state

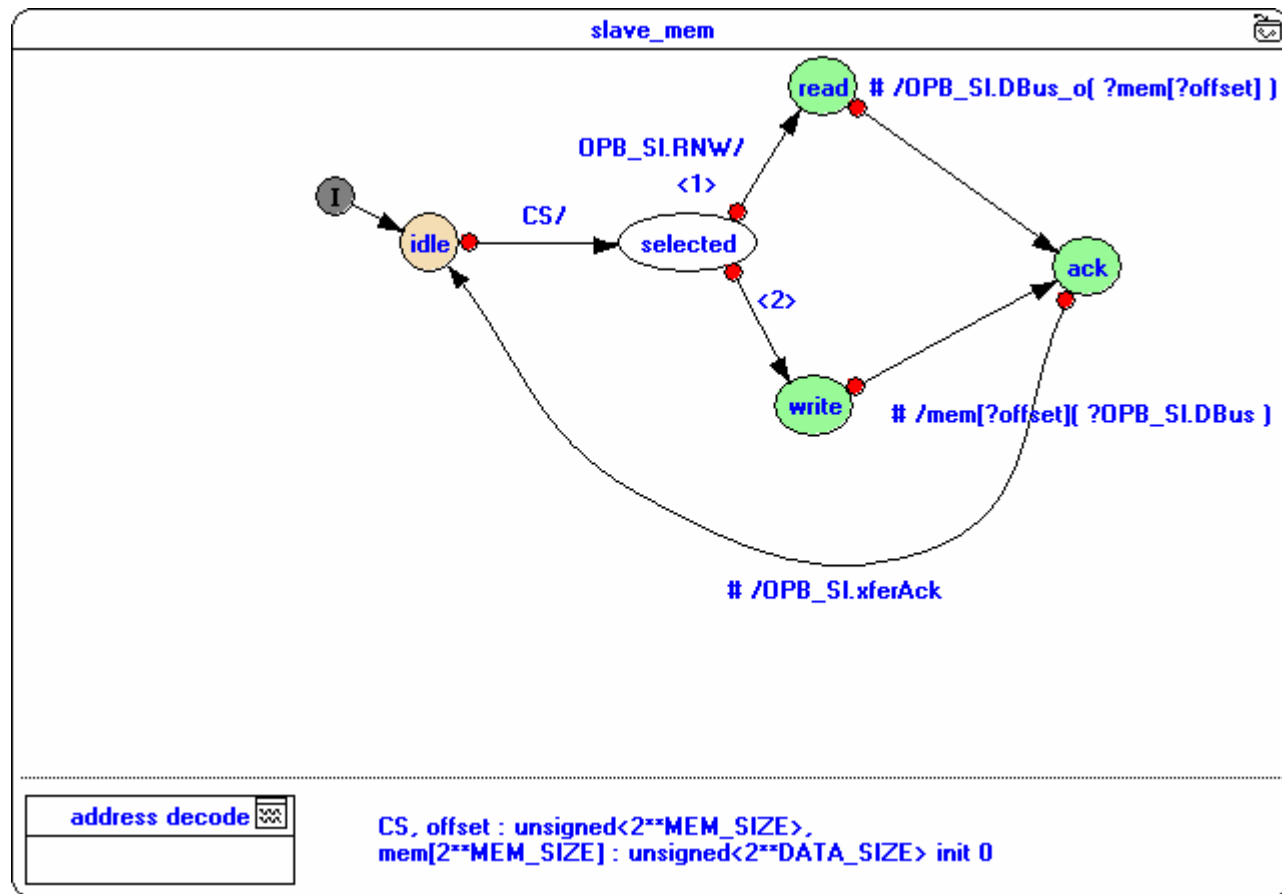


Proven in less than 2 seconds

# Specification of master behavior ...

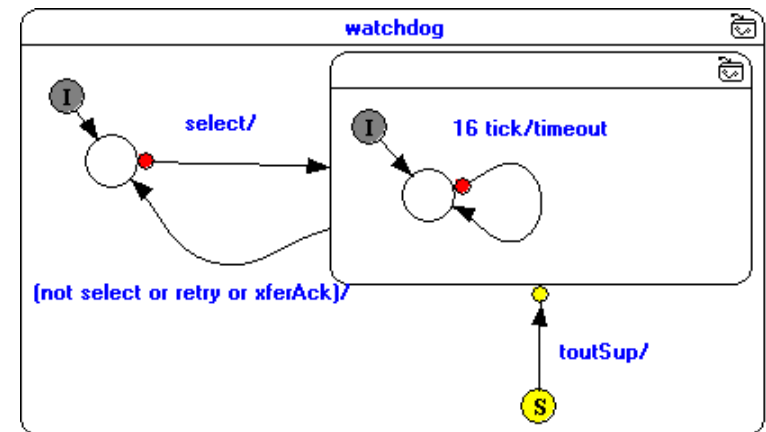
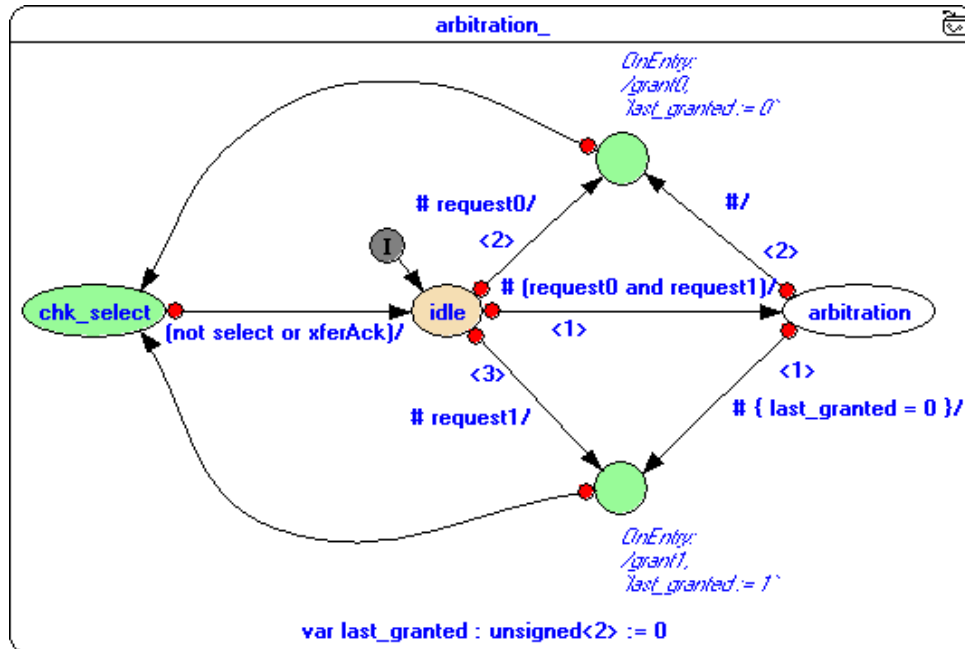


# ... slave



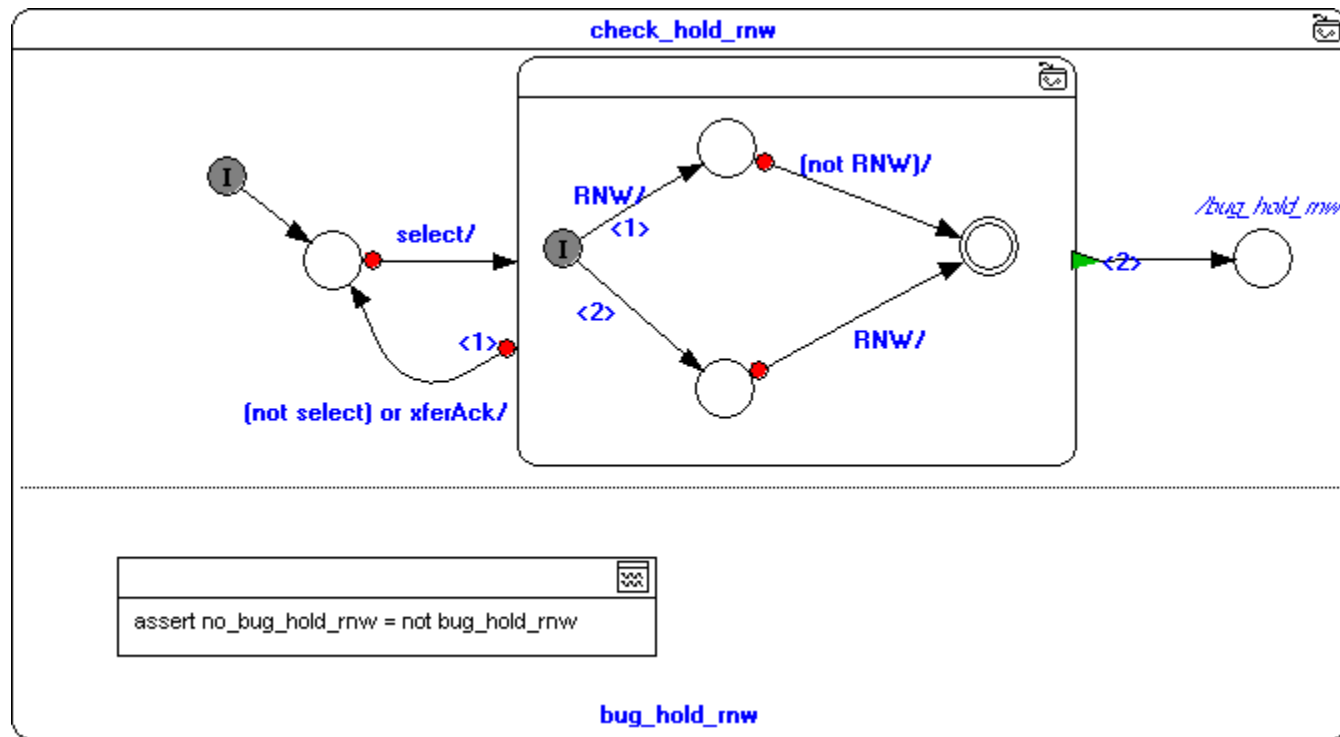


# and arbiter



# OPB Protocol violations

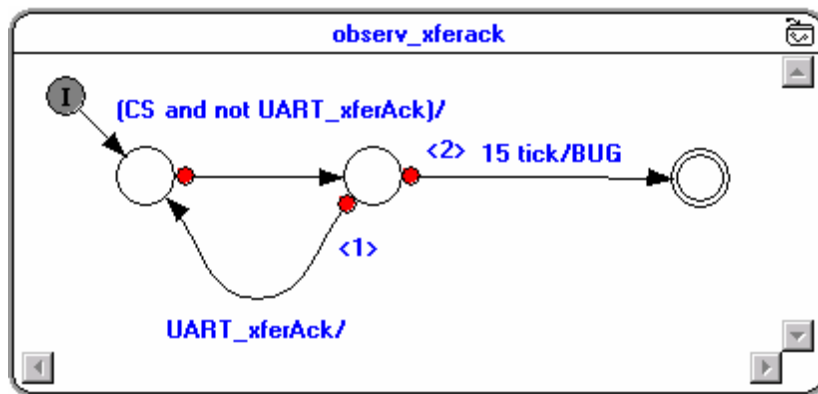
e.g. Checking that RNW doesn't change during a transaction :



# Formal verification

Of the OPB slave interface :

proving that it won't cause bus timeouts



Results		
Type	Name	Status
observer	observ_xferac_BUG	Always Absent

Verification engine used: Prover SL plug-in

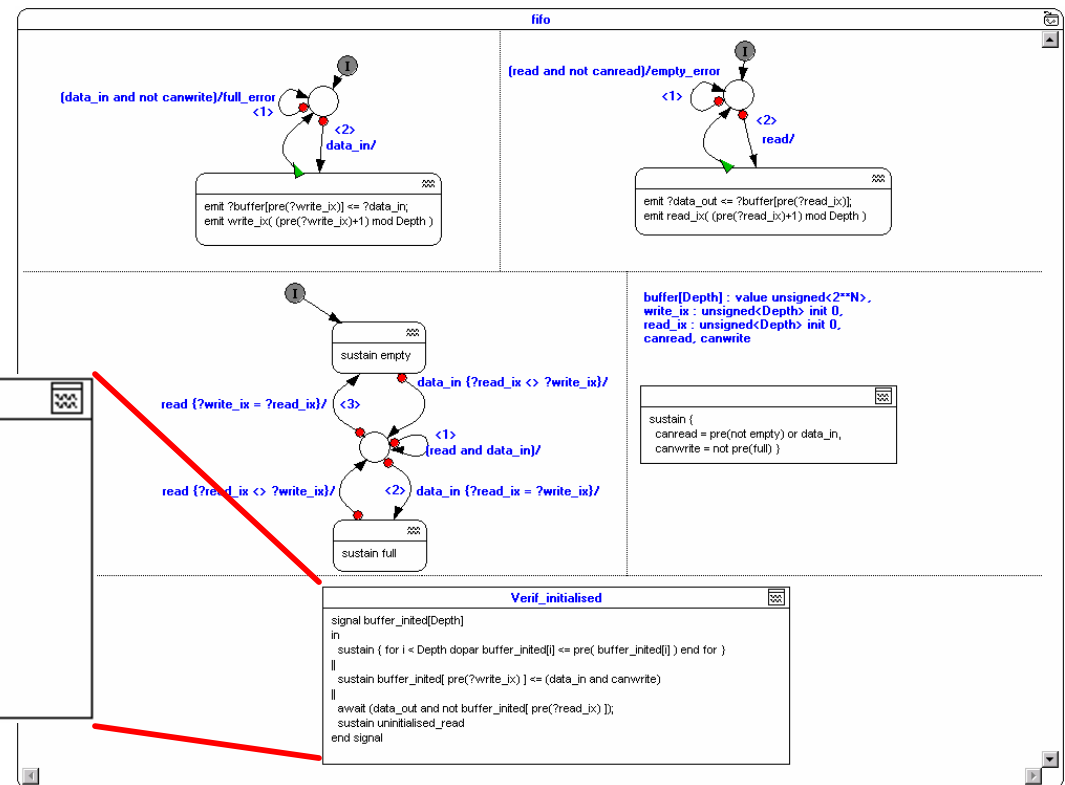
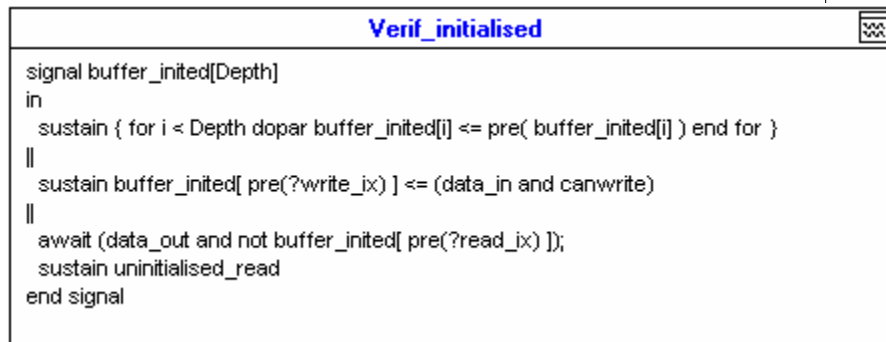
Proven in less than 2 seconds

# Formal verification

Of the FIFO : proving that only initialized data is returned

Using an internal observer to access internal signals

No constraint on input signals



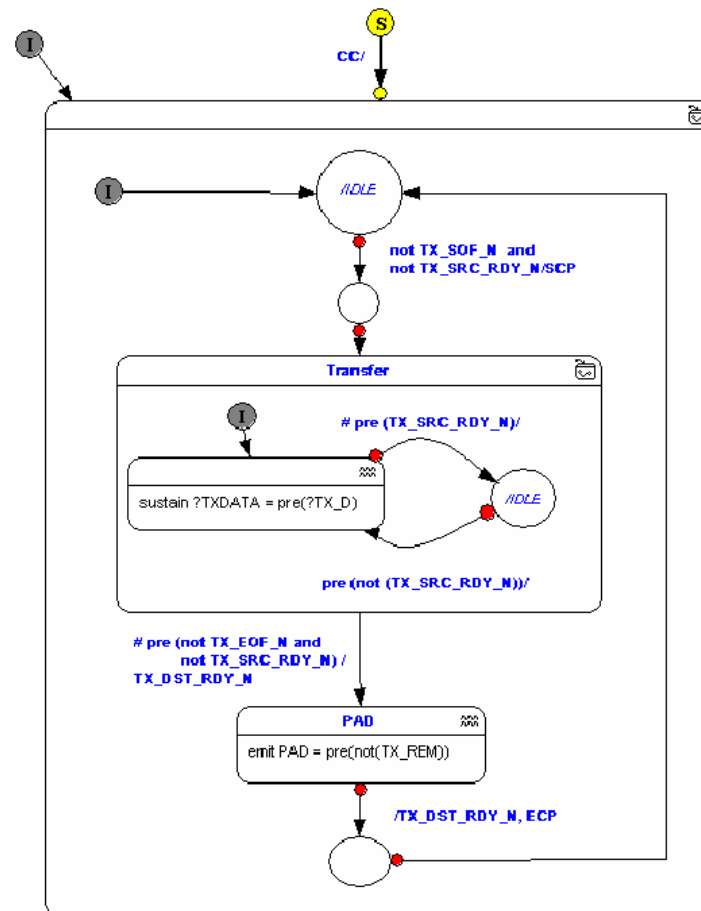
Proven in 30 seconds



# Interactive Deadlock Demo



# Other examples (LocalLink, Aurora, ...)





# Positive Conclusions

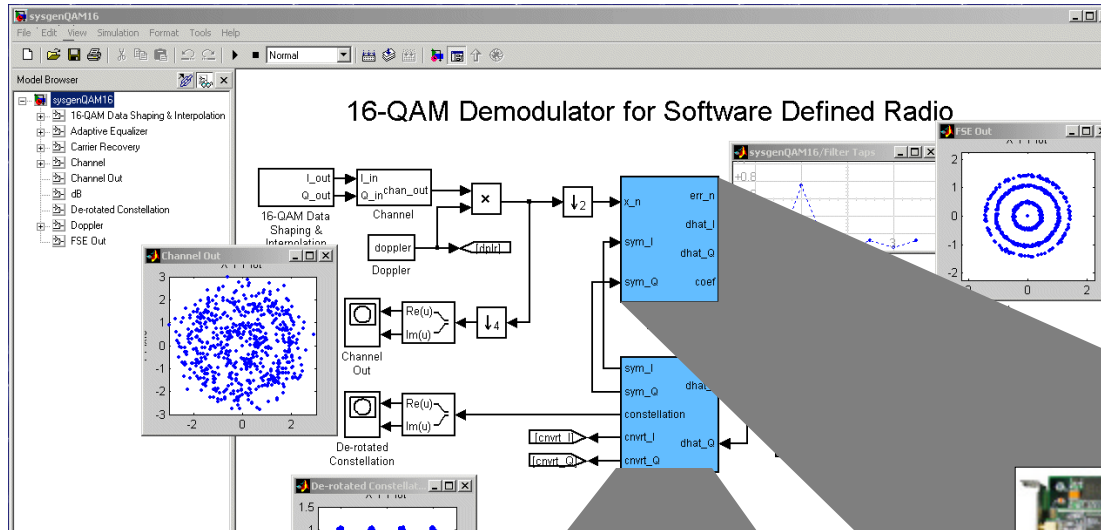
- Control-based calculations can be implemented in hardware using a software style specification in Esterel (“computing without processors”).
- Synchronous observers provide an additional verification technique to simulation, assertion languages (Sugar/OpenVERA etc.) and permits co-verification.
- Co-synthesis allows HW/SW trade-offs to be explored.
- VHDL/RTL provide poor interface between high systems and back-end tools.

# Next Steps

- Currently working on:
  - Xilinx Link Layer protocol (LocalLink, Aurora).
  - TX portion of 10 gigabit ethernet MAC.
- Wire-speed high level processing of gigabit and 10-gigabit traffic.
- Language enhancements to better support HW design.
- Interface synthesis (a la CoWare)
- Control for System Generator

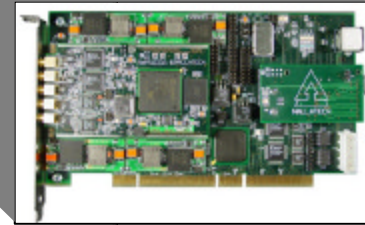


# System Generator



System Generator extends Simulink to support external simulation engines

- Hardware acceleration
- Mixed-mode HDL/data flow



Hardware in the loop co-simulation

HDL co-simulation