

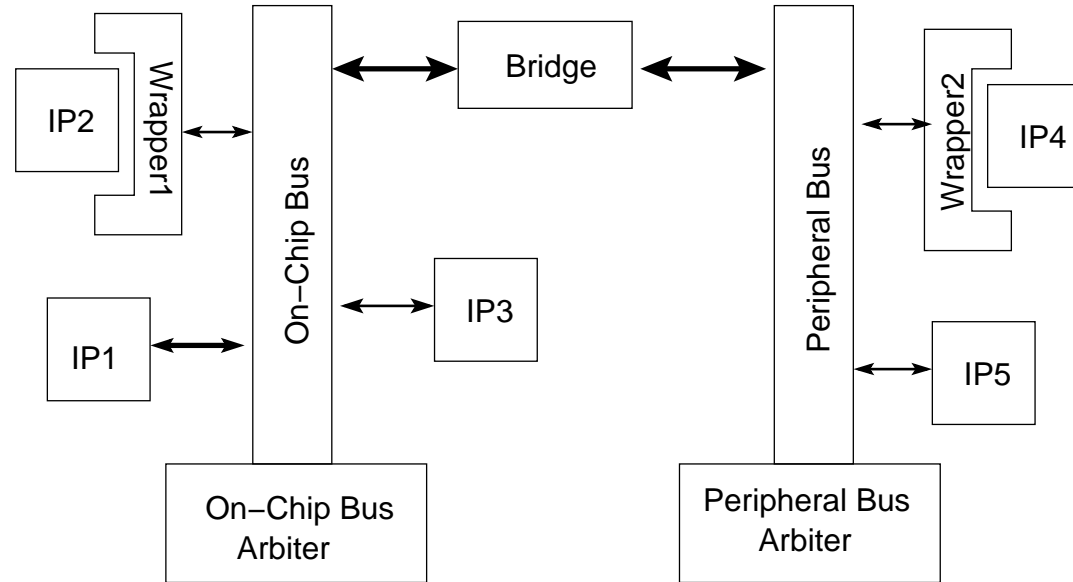
# **Automated Interface Synthesis for Mismatched Protocols**

Vijay D'silva & S. Ramesh & Arcot Sowmya

Center for Formal Design and Verification of Software, IIT Bombay

School of Computer Science and Engineering, University of New South Wales

# Systems-on-Chip Architecture

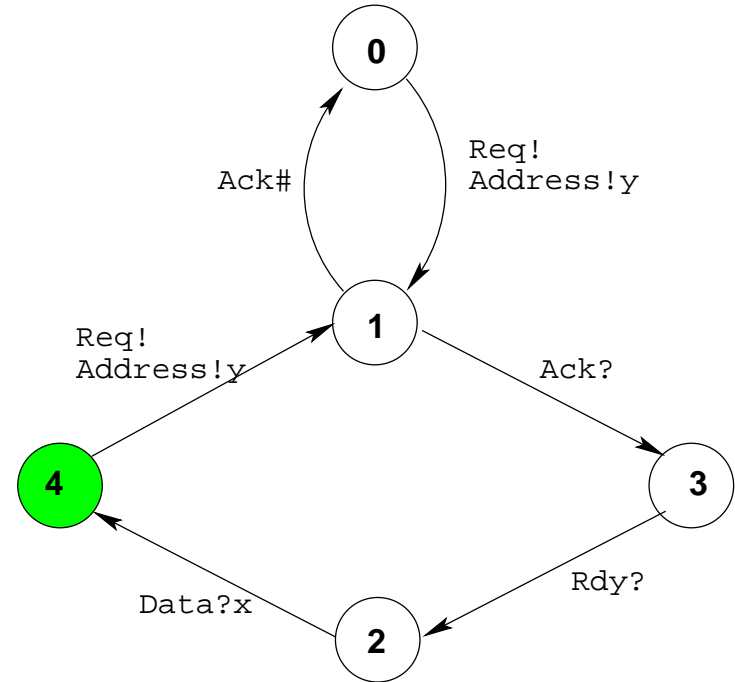
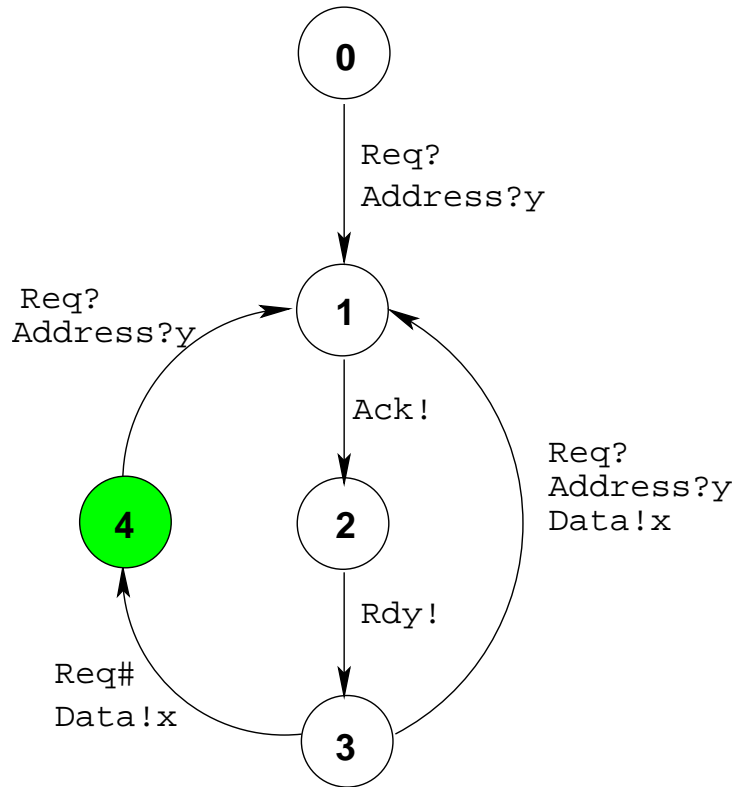


- Set of interconnected IP blocks.
- IP Blocks are pre-verified.
- Ideal Scenario: *Plug-n-Play* style reuse.

# The Real Scenario

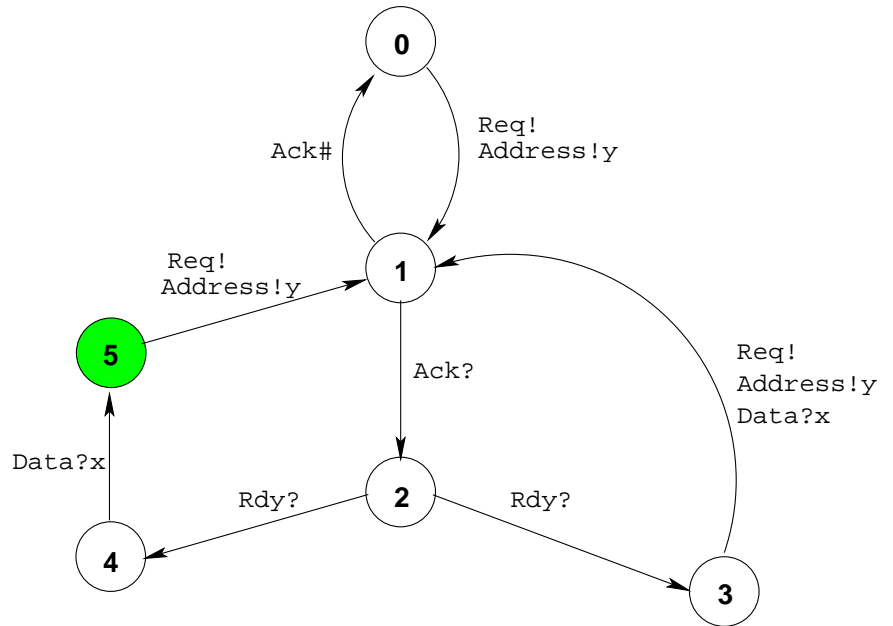
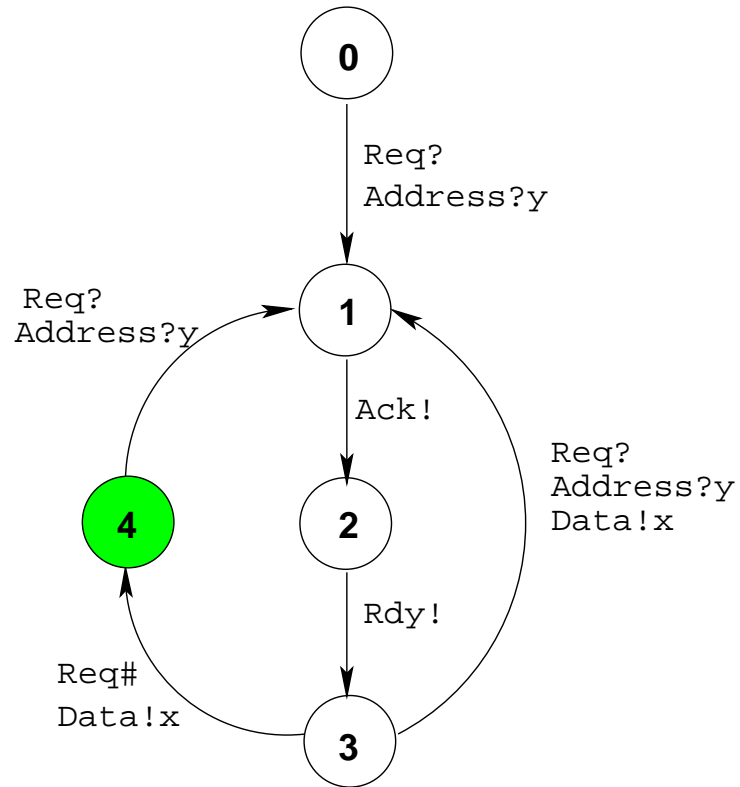
- Bus and socket based architectures.
- No standard exists.
- IP blocks are designed with different communication protocols.
- These protocols may not be 'matching'
- Bridges and wrappers required for integration.
- Manual Design of Glue Logic requires verification
- The focus of our work: synthesis of glue logic or interface
- From protocol specifications
- Interface correct by construction

# A Protocol Pair



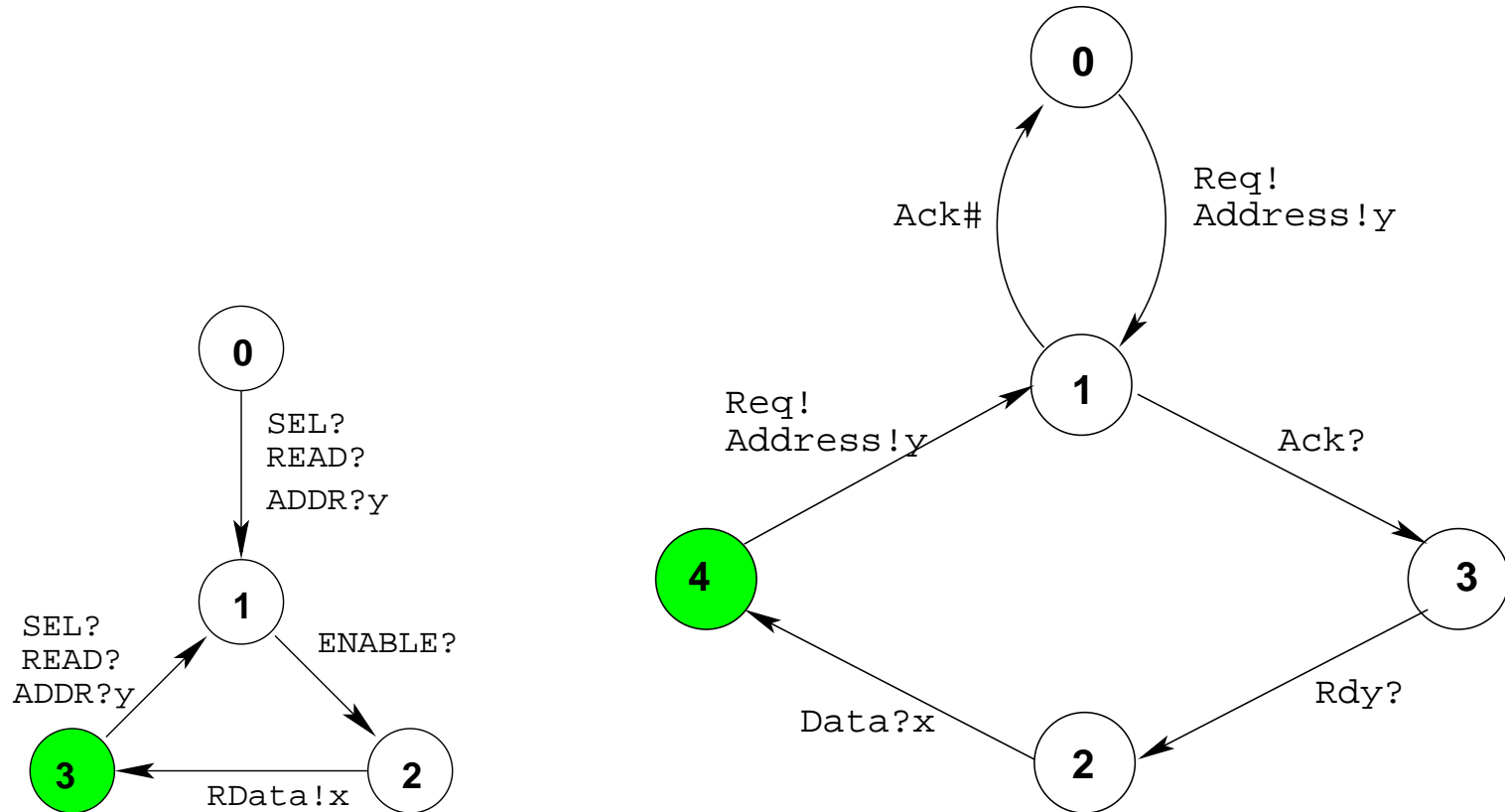
These two protocols can communicate and are matching.

# A Protocol Pair



So can this pair.

# A Mismatched Protocol Pair

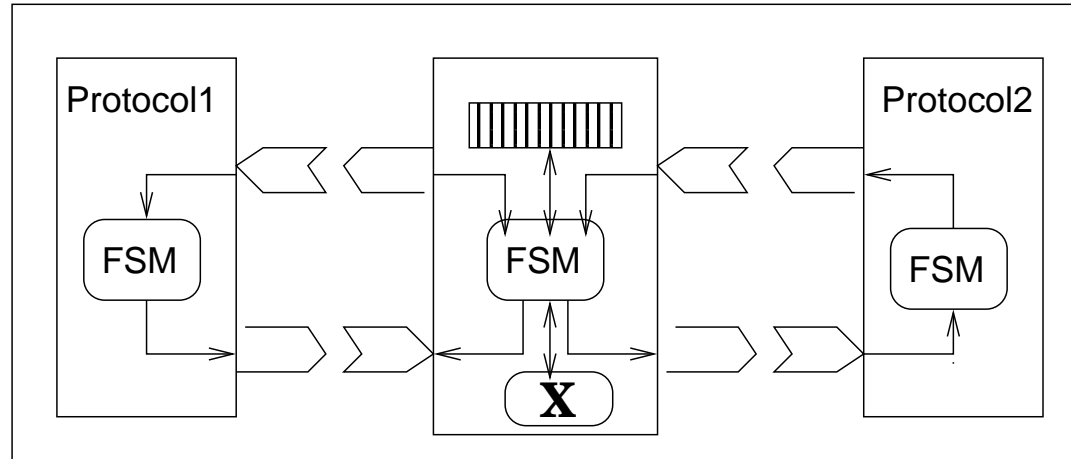


- Connect Req to SEL and READ?
- What about Ack and ENABLE?
- Other problems..

# Causes of Protocol Mismatch

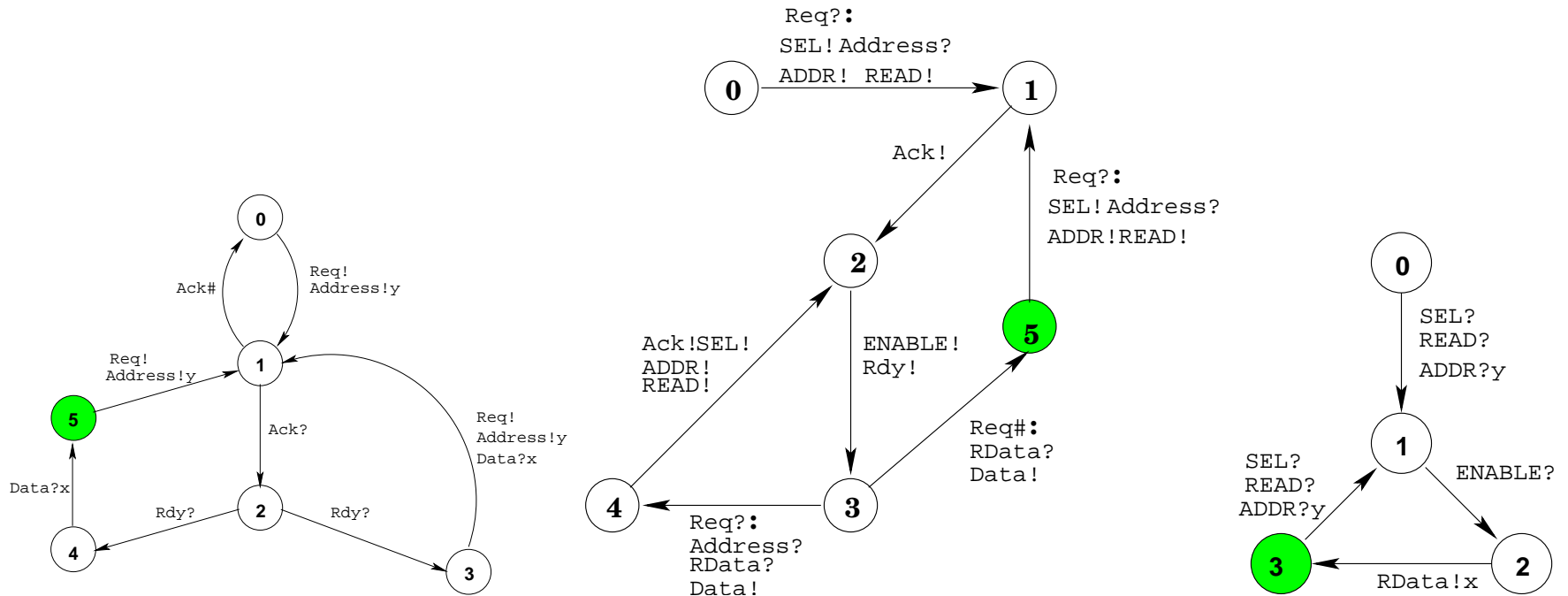
- Signalling conventions vary.
- Bus widths differ.
- Data types differ.
- Clock speeds differ.
- Different complexity and functionality supported.

# What is an Interface?



- A FSM similar to the protocols.
- Two phased.
- Has a counter and data buffers.
- Can perform type transformation.

# Example of an Interface.



Transition from state 3 to 4 for pipelined transfers.

# Our Approach

Given the models:

- Check if a protocol pair matches.
- Synthesise interfaces to resolve protocol mismatches.
- Requires data bus mapping and type transformation information.
- Interface is correct by construction.

Protocol Modelling:

- Restrict our focus to fully synchronous architectures.
- Finite State Machine based representation of protocols.
- No non-deterministic protocols.

# Synchronous Protocol Automata

A *Synchronous Protocol Automaton*  $P$  is a tuple

$$(Q, \Sigma, \Delta, V, \mathcal{A}, \longrightarrow, clk, q_0, q_f)$$

- $Q$  is a set of control states
- $\Sigma = \Sigma^I \cup \Sigma^O$  are control channels.
- $\Delta = \Delta^I \cup \Delta^O$  are data channels.
- $V$  is a set of internal variables.
- $\mathcal{A}$  is the set of actions.
- $\longrightarrow \subseteq Q \times clk? \times \mathcal{A} \times Q$  is the state transition relation.
- $q_0$  and  $q_f$  are the initial and final states.

# Matched Protocol Pairs: Intuitively

- They should be able to complete transactions.
- Information read should not be lost.
- Junk data should not be read.
- Cycle accurate compatibility.

Some notation:

- $\text{permit}(S_1, S_2)$  - for two actions  $S_1$  and  $S_2$  if they are *complementary*.
- $\text{blocking}(q)$  - if all transitions from state  $q$  are guarded.

Note: All states are either blocking or have no guarded transitions.

# Matched Protocols : Formally

1.  $\langle r_f, t_f \rangle \in \mathcal{R}$
2. if  $\langle r, t \rangle \in \mathcal{R}$  and  $\neg\text{blocking}(r)$  and  $\neg\text{blocking}(t)$  then, whenever  $r \xrightarrow{S_1} r'$  and  $t \xrightarrow{S_2} t'$  it holds that  $\text{permit}(S_1, S_2)$  and  $\langle r', t' \rangle \in R$
3. if  $\langle r, t \rangle \in \mathcal{R}$  and  $\neg\text{blocking}(r)$  and  $\text{blocking}(t)$  then, whenever  $r \xrightarrow{S_1} r'$  it holds that there exists  $S_2, t' : (t \xrightarrow{S_2} t' \text{ and } \text{permit}(S_1, S_2))$  and for all such  $S_2, t' : \langle r', t' \rangle \in R$
4. if  $\langle r, t \rangle \in \mathcal{R}$  and  $\text{blocking}(r)$  and  $\text{blocking}(t)$  then, whenever  $r \xrightarrow{S_1} r'$  and  $t \xrightarrow{S_2} t'$  such that  $\text{permit}(S_1, S_2)$  it holds that  $\langle r', t' \rangle \in R$

A protocol pair  $P_1$  and  $P_2$  with initial states  $r_0$  and  $t_0$  is said to match if there exists a transaction relation  $\mathcal{R}$  such that  $\langle r_0, t_0 \rangle \in \mathcal{R}$ .

# Synthesis Algorithm

**Input** : Two protocols. Data bus mapping.

**Output** : Design space of the interface.

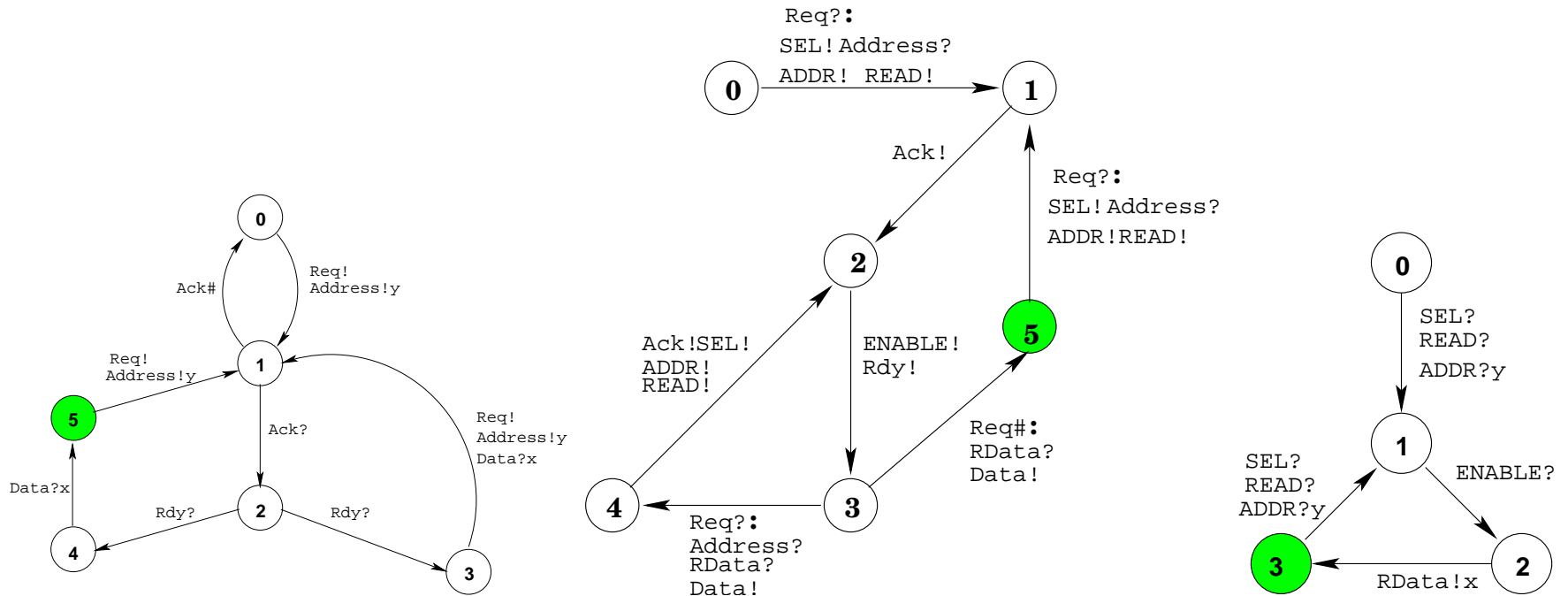
- A state in the interface corresponds to sets of states in the protocols.
- Transition in the interface corresponds to transitions in the protocols.
- Data which is read is saved in a buffer till it is written out.
- Data is written out only when available.

# Synthesis Algorithm

Given transitions  $r \xrightarrow{S_1} r'$  and  $t \xrightarrow{S_2} t'$  in the protocols

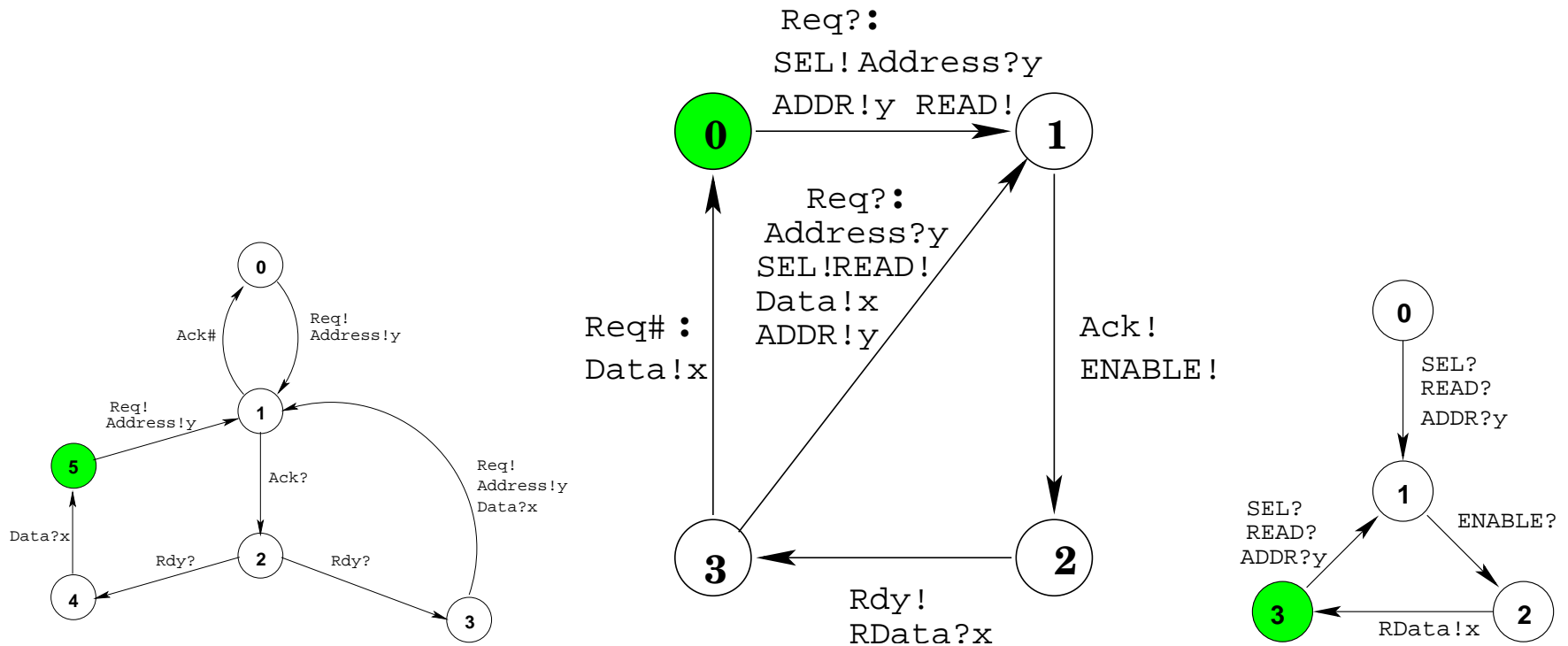
- Compute next states -  $Q_n$ .
- Compute action to be performed  $A_n$ .
- If  $\text{valid}(A_n, X)$  then
  - Add state, transition to interface.
  - Update counters in  $X$ .
- Do until all reachable states are visited.
- Remove loops with unbounded reads.
- Remove paths to states in which data is expected but not output.

# An Interface



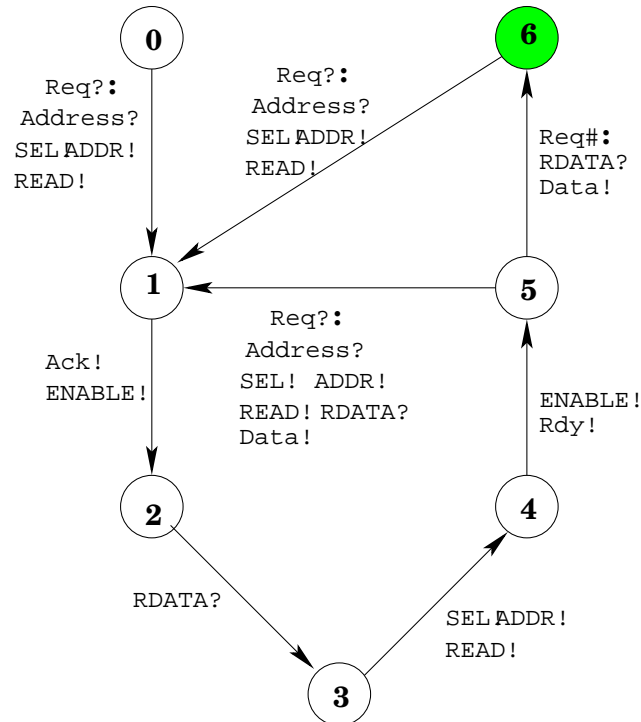
Transition from state 3 to 4 for pipelined transfers.

# Another Interface



Lower latency for transfer but requires a buffer. Designer will have to choose between such tradeoffs.

# Data width and type mismatches

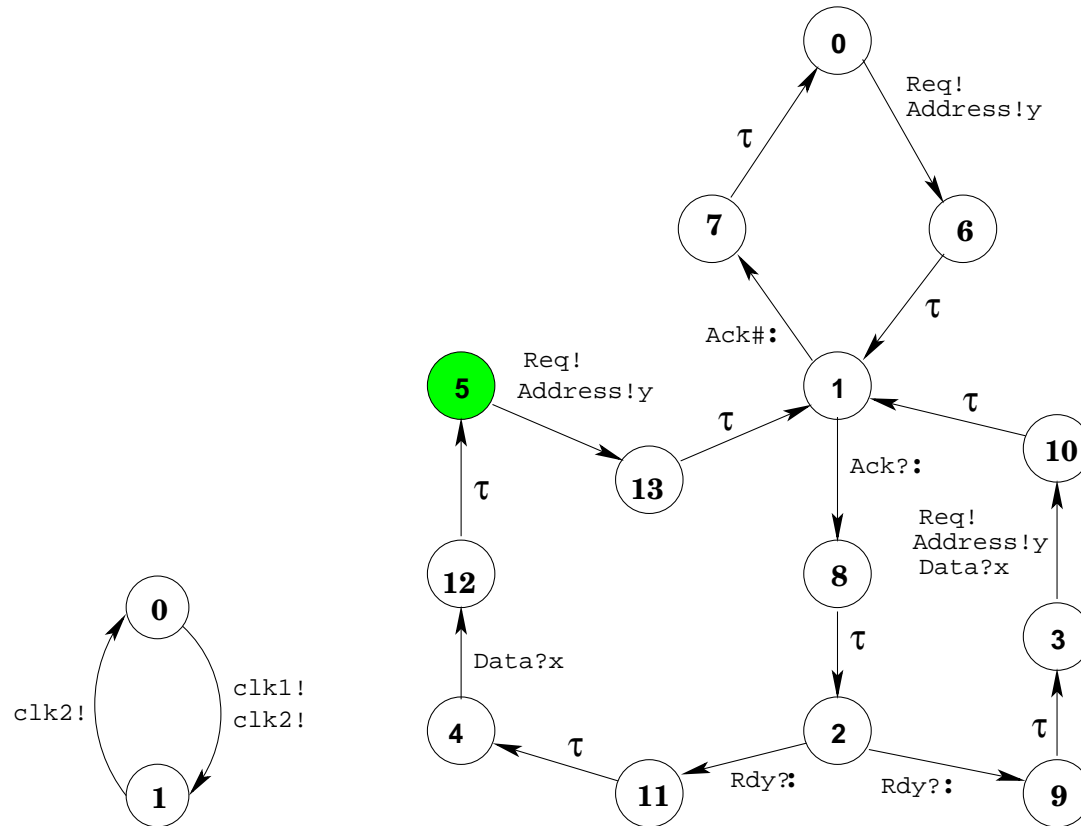


Data is twice as wide as RData.

Assumption: Type transformation is instantaneous.

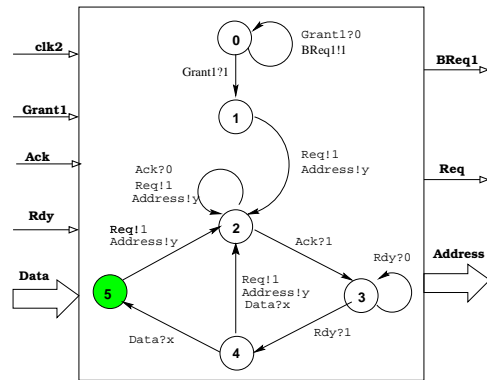
Can add any required latency using the predicate  $\text{valid}(A, X)$

# Systems with Multiple Clocks

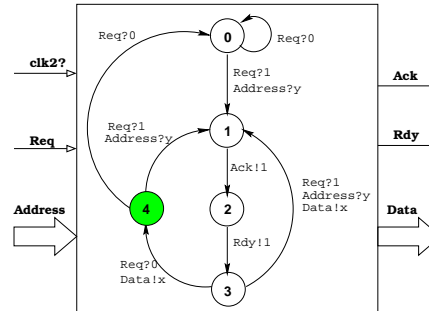


- Clock FSM.
- Compose with protocol automaton. (*oversampling*)
- Now apply algorithm.

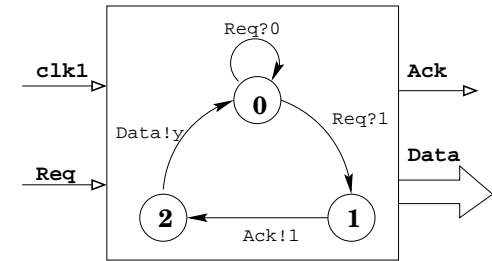
# System Level Modelling



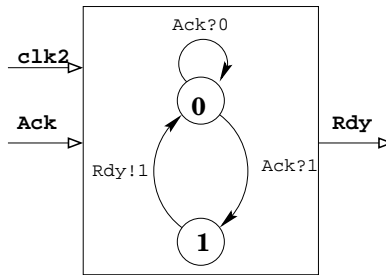
**Protocol 1**



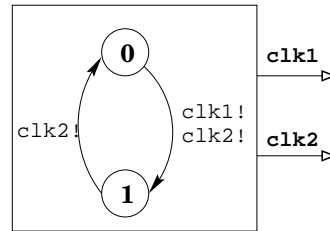
**Protocol 2**



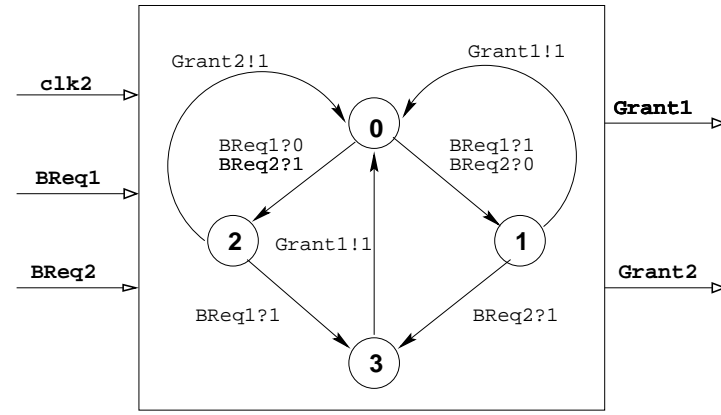
**Peripheral**



**Bridge**



**Clock**



**Arbiter**

# What this allows

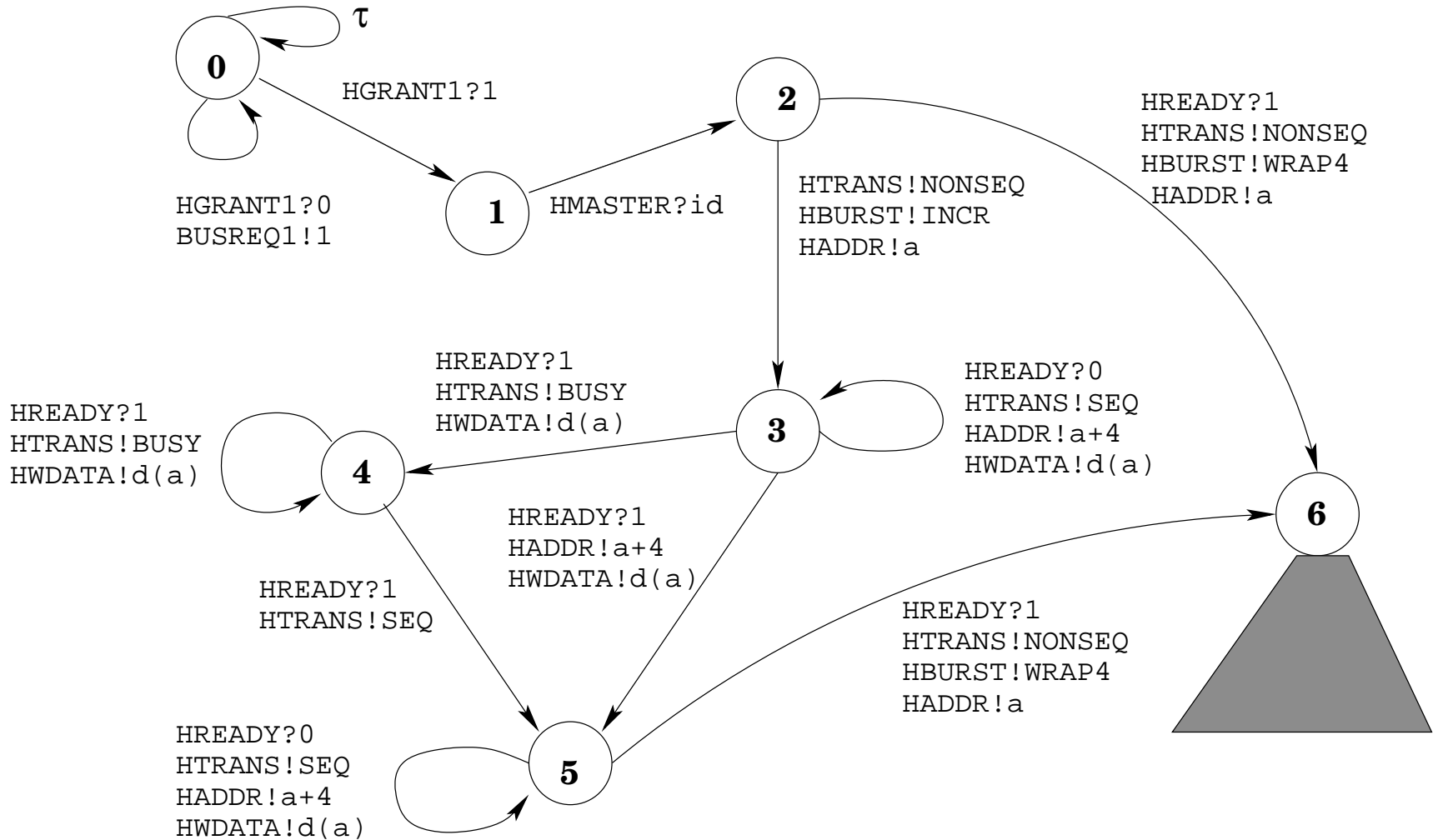
- System level compatibility checks
- Correct by construction of bridges and wrappers.
- Direct translation into any HDL or synchronous language.
- Code will be simulation-ready.

# Experimental Results

- Modelled (parts of) some SoC bus architectures.
- Synthesised interfaces for the same.

Master	Slave	Clocks	BWidth	<i>I</i> States	<i>I</i> Transitions
AHB	PLB	1:1	1:1	7	12
OCP	AHB	1:1	1:1	8	18
OCP	ASB	1:1	1:1	11	24
OCP	PLB	1:1	1:1	8	12
AHB	PLB	1:2	1:1	12	15
OCP	AHB	1:1	1:2	21	74
OCP	ASB	1:1	2:1	16	65
OCP	PLB	1:1	3:1	17	32

# The AMBA High-Speed Bus



# Current Work

- Documentation of various protocols used in the industry.
- Implementation of a tool for performing such modelling, analysis and synthesis.
- Analysis of existing industrial code to extract protocol components and synthesise wrappers.
- Using the technique to check correctness of existing wrappers.