

# TSIF: Transition System Interchange Format

E. Pastor and M.A. Peña

Jan 2004

Department of Computer Architecture  
Technical University of Catalonia (UPC)  
Barcelona, Spain

DAC



## Modeling Transition System

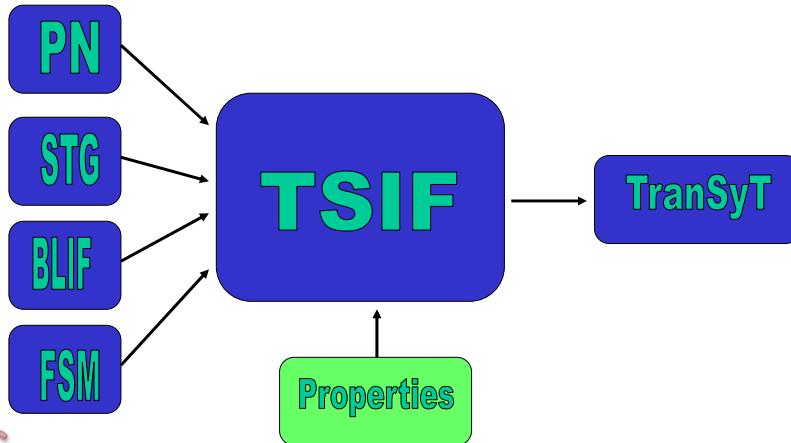
- A model to describe heterogeneous systems that behave asynchronously
  - Boolean encoded states
  - Event-driven
  - Coordinated multi-processes
  - Hierarchical
- The model is supported by a textual format:
  - TSIF: Transition System Interchange Format

DAC



## TSIF Motivation

- General verification-oriented framework to develop applications.



DAC



## Objects in a Transition System

- Based on ... formalism
- States:
  - Symbolic encoding by means of Boolean variables.
- Transitions:
  - Set of labels to describe an alphabet of possible abstract operations, i.e. state transitions.
- Observable versus internal behavior and state:
  - Input / output / internal variables and labels.

DAC



## Objects in a Transition System

- Symbolic manipulation based on functions:
  - Sets of states: Boolean functions.
  - Transition relations: Boolean relations.
  - Supported by BDDs (CUDD / PDTRAV packages).
- State transitions:
  - Executed as events that modify the state (variables).
  - Transition relations describe the evolution.
  - Transitions are always associated to a label.
- Additional functions can be associated to the TS.

## Objects in a Transition System

- Set of instantiated Transition Systems:
  - Allows different levels of refinement.
  - May include automatic abstraction.
- Coordination mechanisms:
  - State observation (sharing between observable variables).
  - Label synchronization of observable labels.

## Outline

- System definition
- Variable declaration
- Behavior declaration
- State declaration
- Process instantiation
- Process coordination
- System annotation
- Examples

## Transition System Interchange Format

- Named object that encapsulates structure and behavior.

Syntax:

```
TS <ts_name> {<ts-type>}  
...{ts description}  
END
```

- TS types could be: INTERLEAVED, SYNCHRONOUS, ...

## TSIF: Variable Declaration (1)

- State space encoded by Boolean **variables**:
  - Digital signals directly encoded.
  - Other objects could be also mapped, e.g. places in PNs, multi-valued signals, etc..
  - No direct support for integer values.
- Requires two BDD variables...
  - One to describe current-state.
  - Second to describe next-state value.

DAC



## TSIF: Variable Declaration (2)

- Variable declaration:  
Syntax:  

```
<type> VARS list-of-variables;
```

```
INPUT VARS a b c;
```

```
OUTPUT VARS d e;
```

```
INTERNAL VARS f g;
```
- Next-state variable for  $a$  is  $NS(a)$
- Each minterm represents an state.

DAC



## TSIF: Behavior Declaration (1)

- Behavior described in terms of **labels** that are executed:
  - Labels could be observed: input/output.
  - Or hidden: internal/dummy.
- Labels can be executed in different contexts: **events**.
  - Each label has a set of associated events.

## TSIF: Behavior Declaration (2)

- Label declaration:

Syntax:

```
<type> LABELS list-of-labels;
```

```
INPUT LABELS a b c;
```

```
OUTPUT LABELS d;
```

```
INTERNAL LABELS f g;
```

- The space of names for labels is independent from the space of names for variables.
- If they coincide ...

## TSIF: Behavior Declaration (3)

- Each event:
  - Must have a **Transition Relation** to describe the state change, i.e. variable change.
  - Other optional objects may be associated: EF, FF, delay, failure conditions,...
- Some required rules:
  - At least one event per label.
  - At least one event must assign values for every variable.

## TSIF: Behavior Declaration (4)

- Event declaration:

Syntax:

```
EVENT <event-name> <label-name>
...{event description}
END
```

- The name of the event is just informative.
- An event name cannot be shared between events of the same label, no restriction between labels...

## TSIF: Behavior Declaration (5)

- Dynamics described by Boolean Relations:
  - **Transition relation** (TR) for the actual variable change.
  - **Enabling function** (EF) for the subset of states where the TR applies.
  - **Firing function** (FF) for lazy behavior.
  - Other functions at your will. Each function associated to a given “package”.
- Functions described with the **EQN format**:
  - Basic logic operators, but extended with special sets.

DAC



## TSIF: Behavior Declaration (6)

- Function declaration:  
Syntax:  

```
EQN <f-name> [<f-type>]  
expression1; expression2; ...
```
- Predefined function and package names:
  - Reachability analysis (RGA): TR EF FF ISTATE UBOUND
  - Region analysis (REGION)...
- Partitioned functions, f-types: CONJUNCTIVE, DISJUNCTIVE.

DAC



## TSIF: Behavior Declaration (7)

### ■ Function syntax:

|        |                  |  |                    |
|--------|------------------|--|--------------------|
| expr:: | var              |  | NS(var)            |
|        | (expr)           |  | expr'              |
|        | expr + expr      |  | expr * expr        |
|        | expr expr        |  | expr = expr        |
|        | TR(label)        |  | EF(label)          |
|        | FF(label)        |  | FAIL(label)        |
|        | TR(event, label) |  | EF(event, label)   |
|        | FF(event, label) |  | FAIL(event, label) |
|        | ...              |  |                    |

## TSIF: State declaration (1)

### ■ Additional sets can be declared by means of functions:

- Initial state
- Upper bounds on the state space
- Invariants

## TSIF: Process instantiation (1)

- A TS may include instances:
  - references to a **formal** model that ...
  - are instantiated with an **actual** name.
- The formal TS must be already present to check the semantic correctness.
- The same formal TS can be instantiated several times but with different “actual” names.
- Only input/output labels and variables are observable.

DAC



## TSIF: Process instantiation (2)

- Instance declaration:
  - Syntax:

```
PROCESS <formal-ts>: list-of-actual-ts;
```
- For a given Transition System:
  - the name of the actual instance must be unique.
- An instance can be referenced by using:
  - `<ts-name>.<instance-name>...`

DAC



## TSIF: Process coordination (1)

- Instances at the same scope level can be coordinated.

Two mechanisms:

- Synchronization:

Simultaneous execution of events in different instances.

Specified by a set of labels that must be synchronized.

- Sharing:

Total or partial observation of the state of an instance.

A second instance can take decisions based on that information.

State sharing is equivalent to variable sharing.

DAC



## TSIF: Process coordination (2)

- Variable sharing:

Syntax:

```
SHARED <var-name>: <inst-name>.<var-name> ... ;
```

- Variables must be in the scope of the TS.
- Variables must be visible (INPUT or OUTPUT).
- Shared variables are associated to another variable at the current hierarchy level.

DAC



## TSIF: Process coordination (3)

### ■ Label synchronization:

Syntax:

```
SYNCH <lab-name>: <inst-name>.<lab-name> ... ;
```

- Labels must be in the scope of the TS.
- Labels must be visible (INPUT or OUTPUT).
- Synchronized labels are associated to another label at the current hierarchy level.

DAC



## TSIF: Process coordination (4)

### ■ Semantics of the coordination.

Open problems:

- How many “active” variables should be accepted when sharing?
- How many “active” labels should be accepted when synchronizing?

### ■ Label synchronization semantics...

“Output label synchronizes with input labels”

- All labels executed simultaneously:

$$TR(\text{synch}) = \prod TR(\text{labels})$$

- But, when is the synchronization enabled?

$$EF(\text{synch}) = \sum EF(\text{labels})$$

DAC



## TSIF: System annotation (1)

### ■ PACKAGES:

- RGA
  - ✓ reachability analysis + failure analysis
- REGION
  - ✓ generalized region analysis (ER, SR, etc.)
  - ✓ initial support for region theory
- DELAY
  - ✓ min-max delay model

### ■ Feed packages with two types of information:

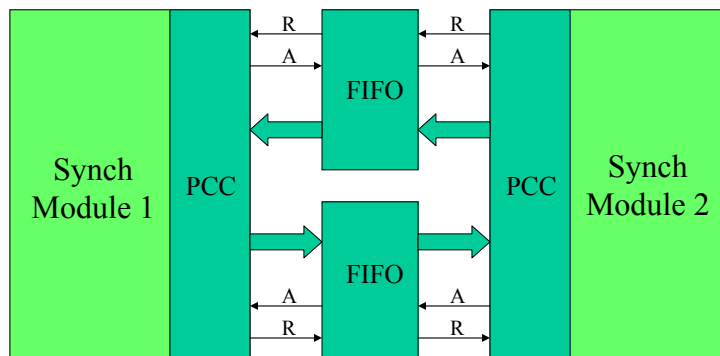
- EQN functions + free-form strings, e.g.
  - {RGA: [CHECK\_PER]}
  - {DELAY: [MIN = 1.0; MAX = 3.0;]}

DAC



## Examples: GALS communication

- Synchronous modules communicating via asynchronous FIFO channels.

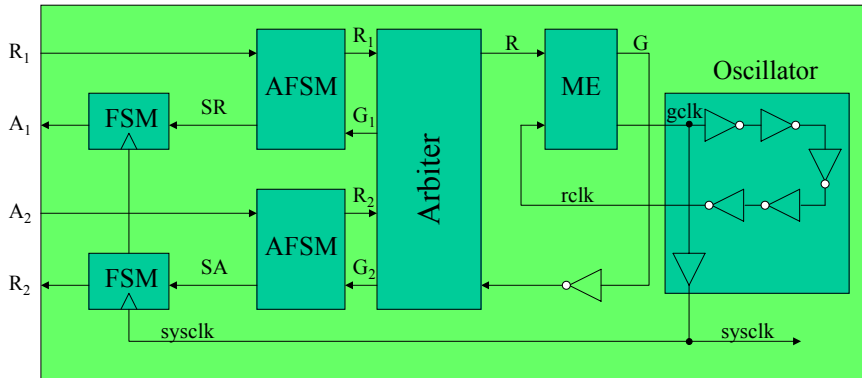


DAC



## Examples: GALS communication

- Heterogeneous system composed of circuits and automata.



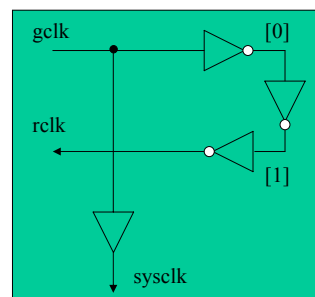
DAC



## Examples: ring oscillator

- BLIF description:

```
.model oscillator
.inputs gclk
.outputs rclk sysclk
.gate inv A=gclk O=[0]
.gate inv A=[0] O=[1]
.gate inv A=[1] O=rclk
.gate buf A=gclk O=sysclk
.end
.initial !gclk !sysclk rclk
```



DAC

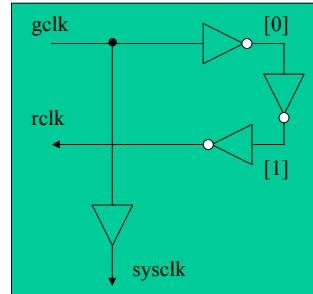


## Examples: ring oscillator

### ■ TSIF description: module interface

```
TS oscillator INTERLEAVED
```

```
INPUT VARS gclk;  
OUTPUT VARS rclk sysclk;  
INTERNAL VARS [0] [1];  
  
INPUT LABELS gclk;  
OUTPUT LABELS rclk sysclk;  
INTERNAL LABELS [0] [1];
```



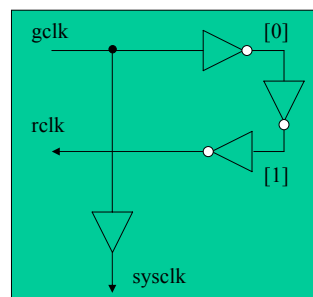
DAC



## Examples: ring oscillator

### ■ TSIF description: events

```
EVENT rise [0]  
EQN TR NS([0]) [0]' (gclk');  
END  
  
EVENT fall [0]  
EQN TR NS([0])' [0] (gclk)';  
END  
  
or:  
  
EVENT switch [0]  
EQN TR NS([0]) = gclk';  
END
```



DAC



## Examples: ring oscillator

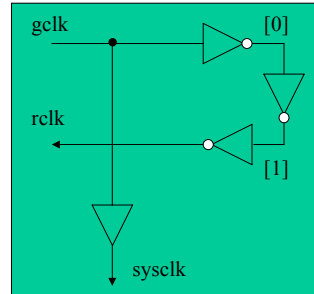
### ■ TSIF description: events

```
EVENT rise [0]
EQN TR NS([0]) [0]' (gclk');
END

EVENT fall [0]
EQN TR NS([0])' [0] (gclk)';
END
```

### ■ TSIF description: events

```
EQN ISTATE gclk' [0] [1]' rclk sysclk';
```



## Examples: ring oscillator

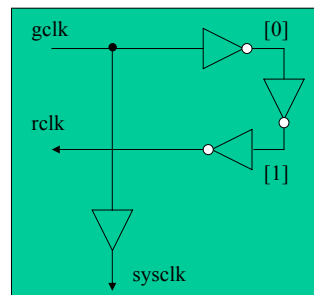
### ■ TSIF description: process

```
TS inv INTERLEAVED
INPUT VARS a;
OUTPUT VARS b;
INTERNAL LABELS b;

EVENT rise b
EQN TR NS(b) b' a';
END

EVENT fall b
EQN TR NS(b)' b a;
END

END
```



# Examples: ring oscillator

## ■ TSIF description: process

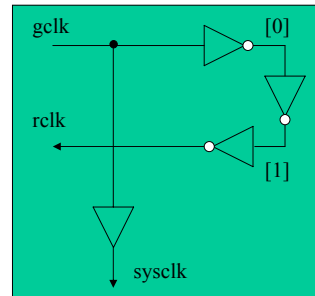
```

TS oscillator INTERLEAVED
INPUT VARS gclk;
OUTPUT VARS rclk;
INTERNAL VARS [0] [1];

PROCESS inv: [0];
PROCESS inv: [1];
PROCESS inv: rclk;

SHARED gclk : [0].a;
SHARED [0] : [0].b [1].a;
SHARED [1] : [1].b rclk.a;
SHARED rclk : rclk.b;

EQN ISTATE gclk' [0] [1]' rclk;
END
    
```



# Examples: ASTG specification

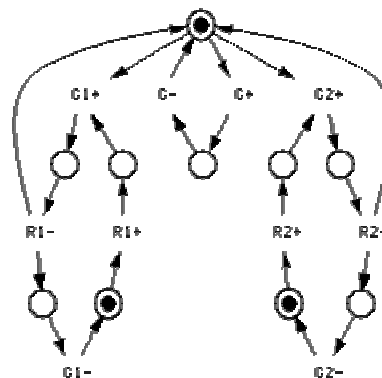
## ■ STG: Signal Transition Graph

- Petri net like specification
- Transitions represent signal switches
- ASTG: syntax used by SIS / Petrifly

## ■ ASTG description:

```

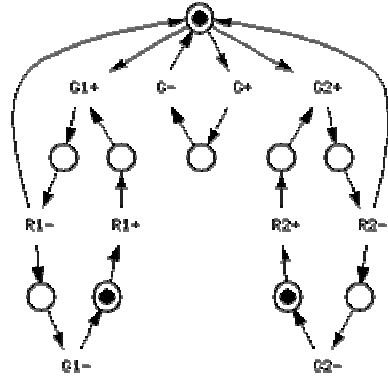
.model gME
.inputs R1 R2 G
.outputs G1 G2
.graph
choice      G1+ G2+ G+
G+          G-      choice
G-          G1+    R1-
...
R1+        G1+
G2+        R2-
...
R2+        G2+
.marking {<G1-,R1+> <G2-,R2+> choice}
.initial !R1 !G1 !R2 !G2 !G
.end
    
```



## Examples: ASTG specification

### ■ TSIF description (interface):

```
TS gME INTERLEAVED
INPUT VARS R1 R2 G;
OUTPUT VARS G1 G2;
INTERNAL VARS choice pin0 pin1 pin2;
INTERNAL VARS pin3 pin4 pin5 pin6;
INTERNAL VARS pin7 pin8;
INPUT LABELS R1 R2 G;
OUTPUT LABELS G1 G2;
```



DAC

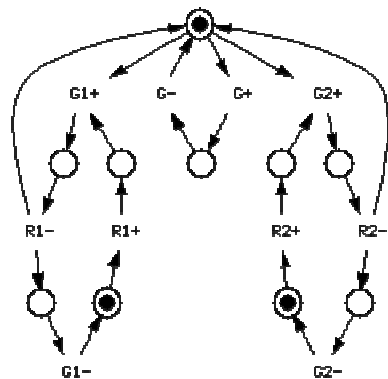


## Examples: ASTG specification

### ■ TSIF description (event description):

```
EVENT R1+ R1
EQN TR NS(R1) = R1'; R1 = 0;
pin3 NS(pin3)' pin4' NS(pin4);
END

EVENT R1- R1
EQN TR NS(R1) = R1'; R1 = 1;
choice' NS(choice);
pin1 NS(pin1)' pin2' NS(pin2);
END
```



DAC

