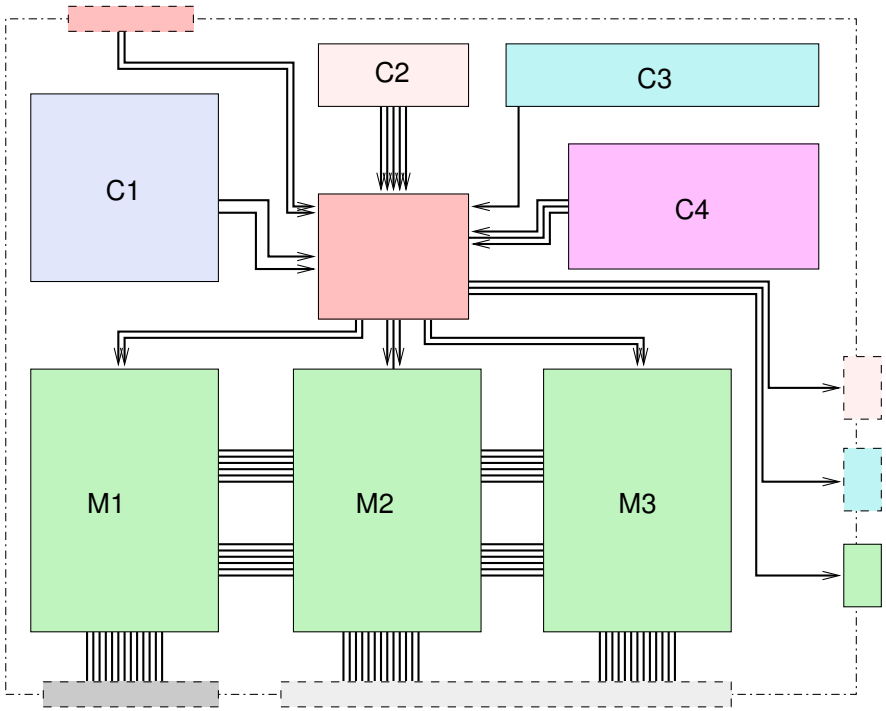
Horus: A prototype debug platform for assertion-based design

D. Borrione, L. Ferro, K. Morin-Allory, Y. Oddos, L. Pierre

TIMA Laboratory

October 17, 2007

SoC Verification



Assume Guarantee Paradigm

- Properties used to describe
  - environment behavior: assumptions
  - design behavior: assertions

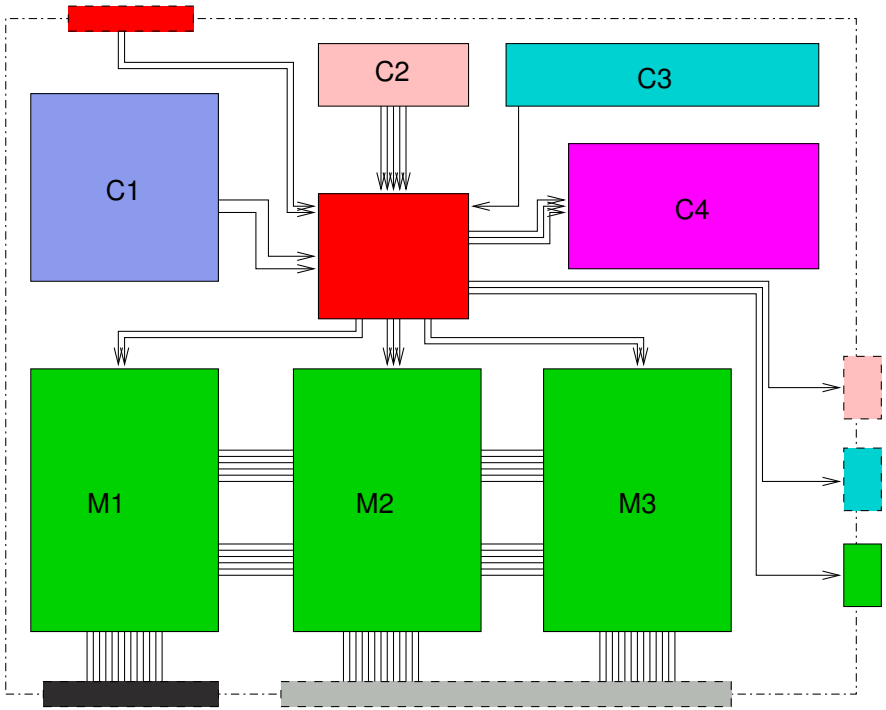
Example

```
always A ->B before C
```

For on-line testing

Synthesize assumptions and

SoC Verification



Assume Guarantee Paradigm

- Properties used to describe
  - environment behavior: assumptions
  - design behavior: assertions

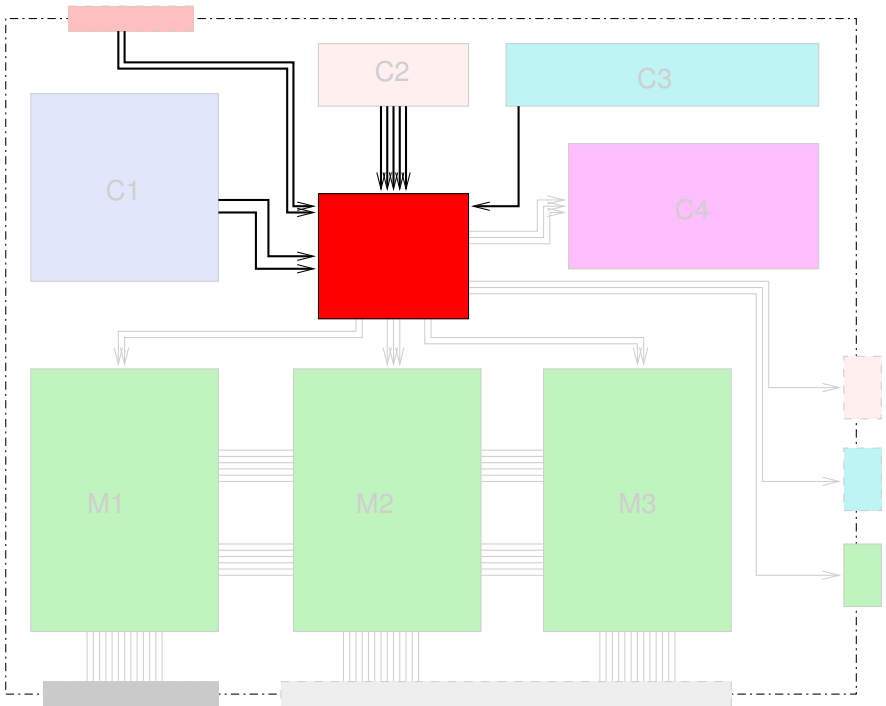
Example

```
always A ->B before C
```

For on-line testing

Synthesize assumptions and

SoC Verification



Assume Guarantee Paradigm

- Properties used to describe
  - environment behavior: assumptions
  - design behavior: assertions

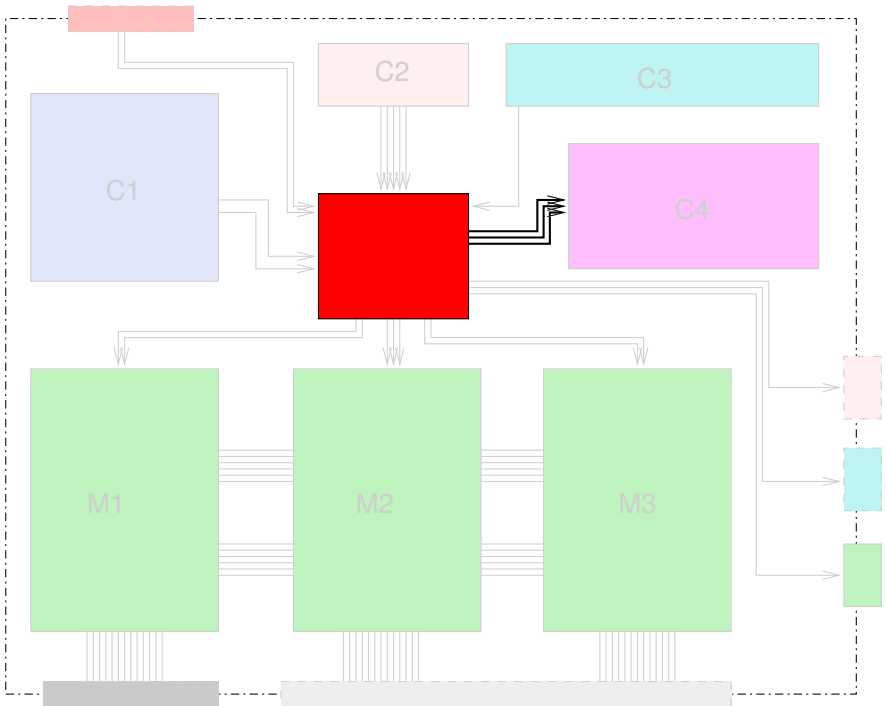
Example

```
always A ->B before C
```

For on-line testing

Synthesize assumptions and

SoC Verification



Assume Guarantee Paradigm

- Properties used to describe
  - environment behavior: assumptions
  - design behavior: assertions

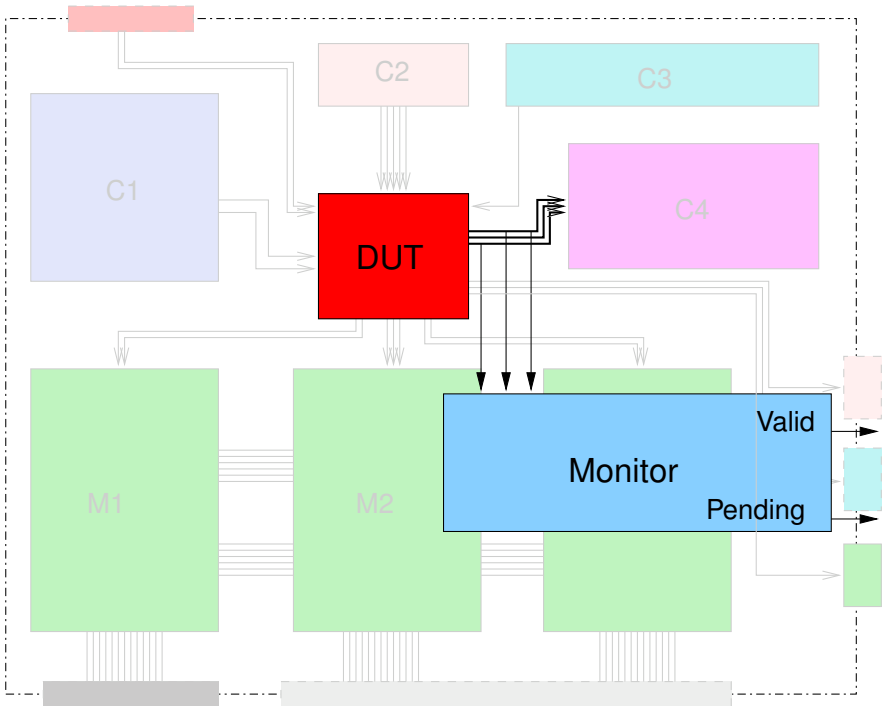
Example

```
always A ->B before C
```

For on-line testing

Synthesize assumptions and

SoC Verification



Assume Guarantee Paradigm

- Properties used to describe
  - environment behavior: assumptions
  - design behavior: assertions

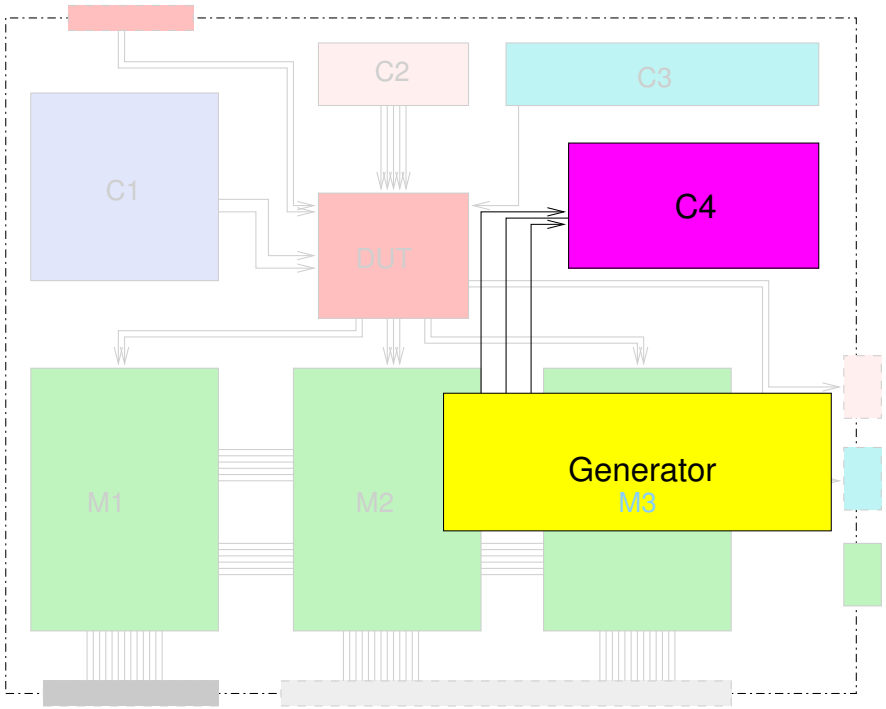
Example

```
always A ->B before C
```

For on-line testing

Synthesize assumptions and

SoC Verification



Assume Guarantee Paradigm

- Properties used to describe
  - environment behavior: assumptions
  - design behavior: assertions

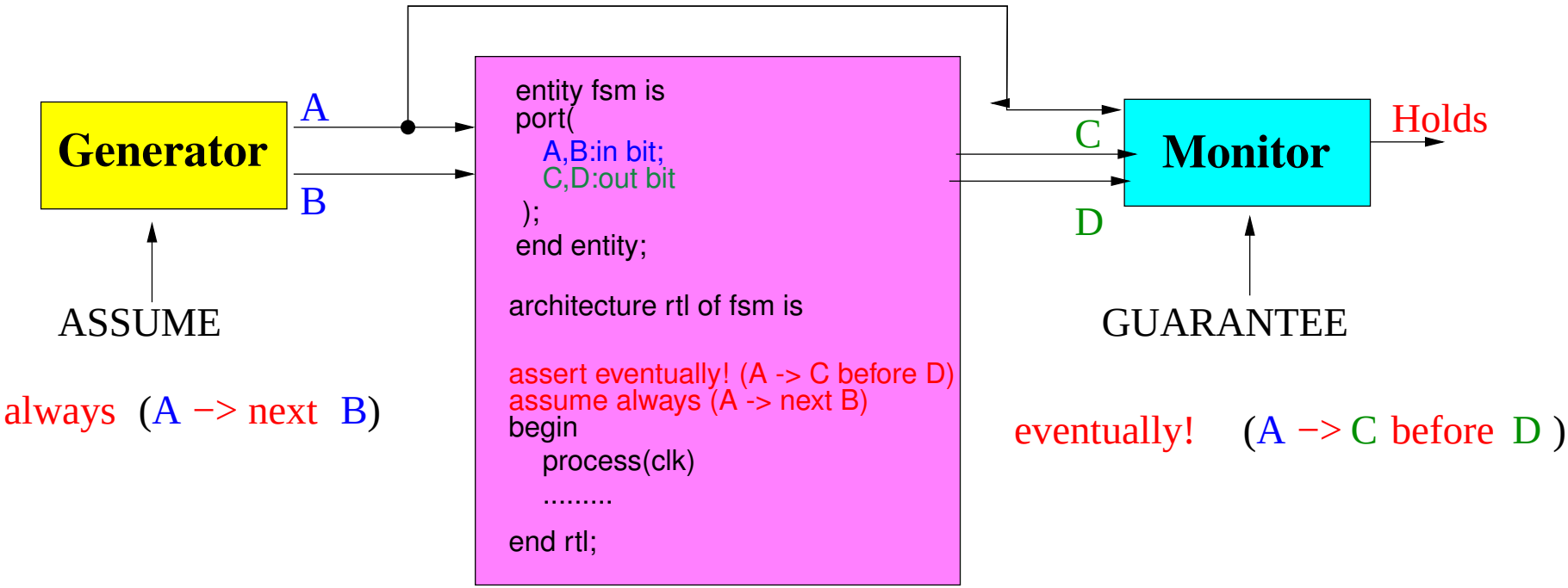
Example

```
always A ->B before C
```

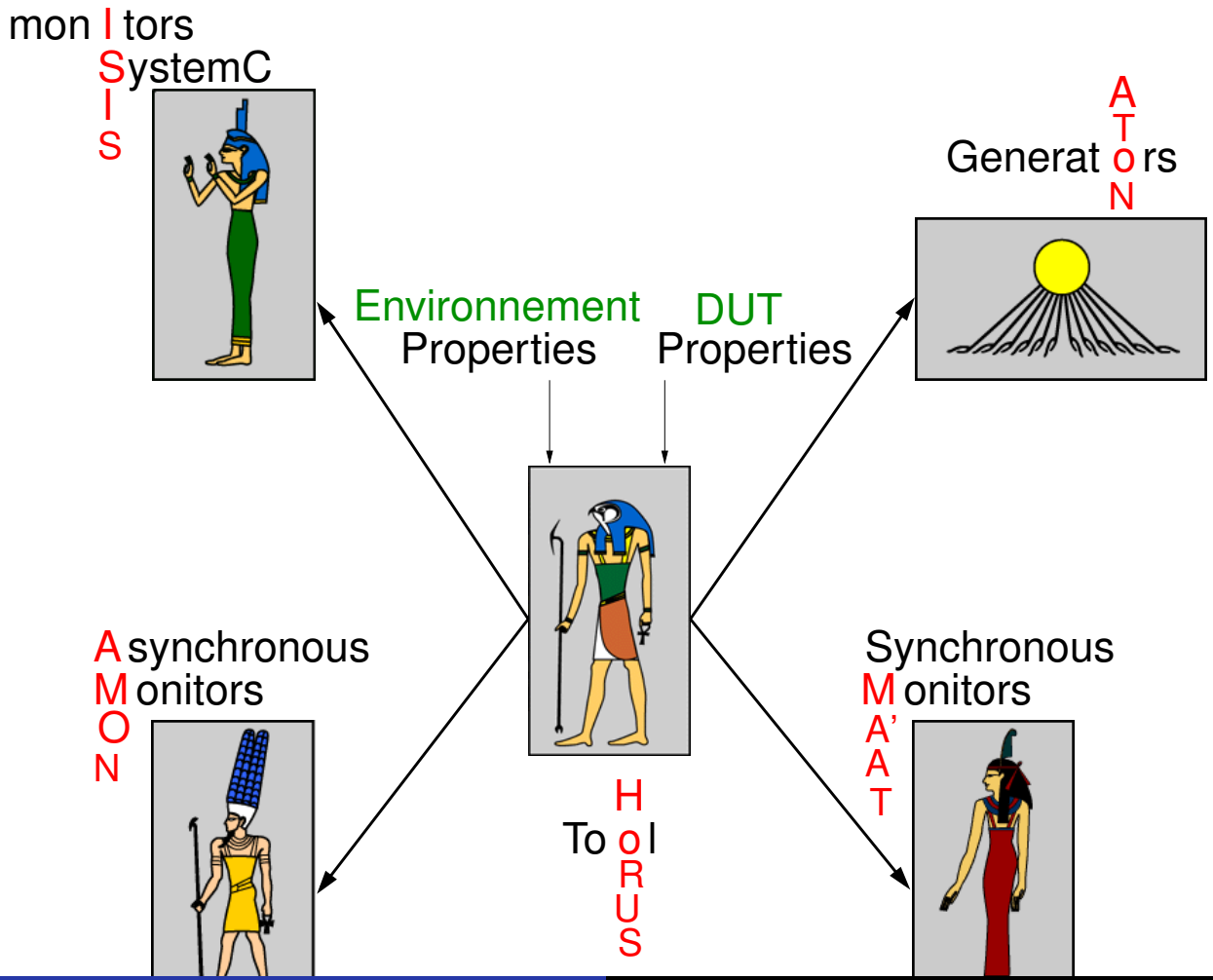
For on-line testing

Synthesize assumptions and

Monitors and Generators



Our Global Project



Our Objectives

Monitors or Generators that

- Cover all PSL operator: FL, SERE, boolean layers
- Synthesizable
- Efficient to build: Modular Architecture
- Area Efficient
- Formally proven correct

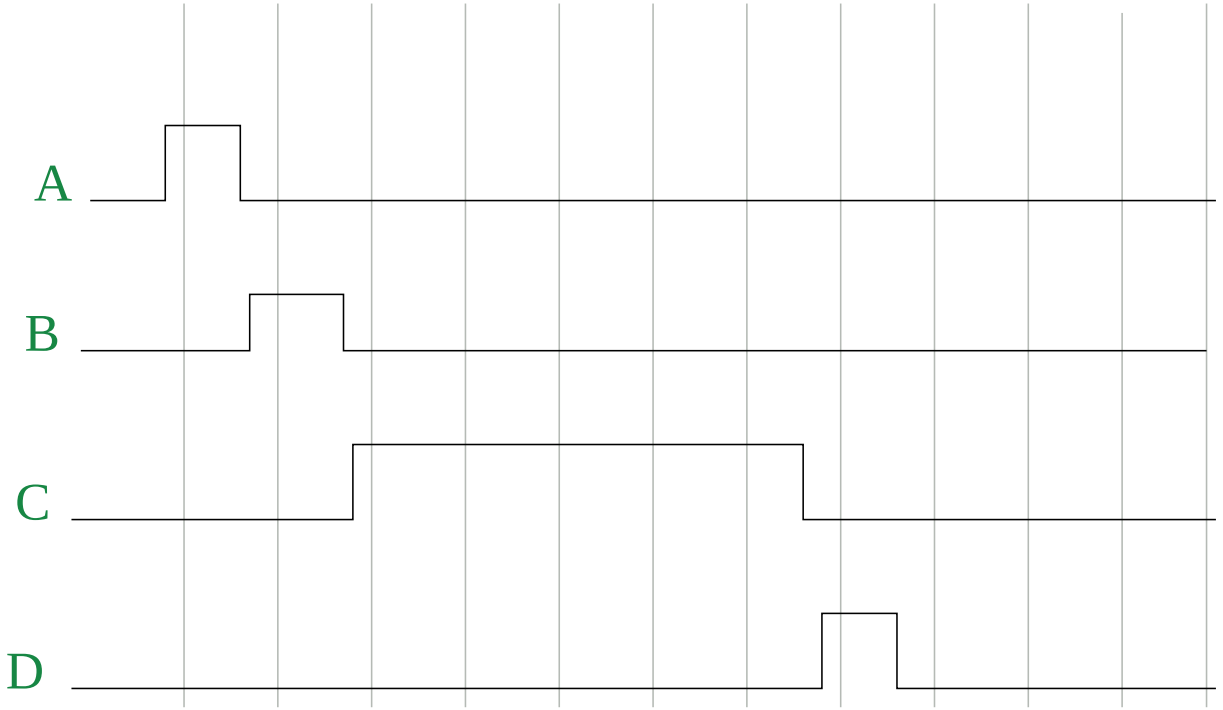
Outlines

1. PSL
2. Monitors and Generators Construction
3. Experimental Results
4. The Horus Platform

Introductory example

Example

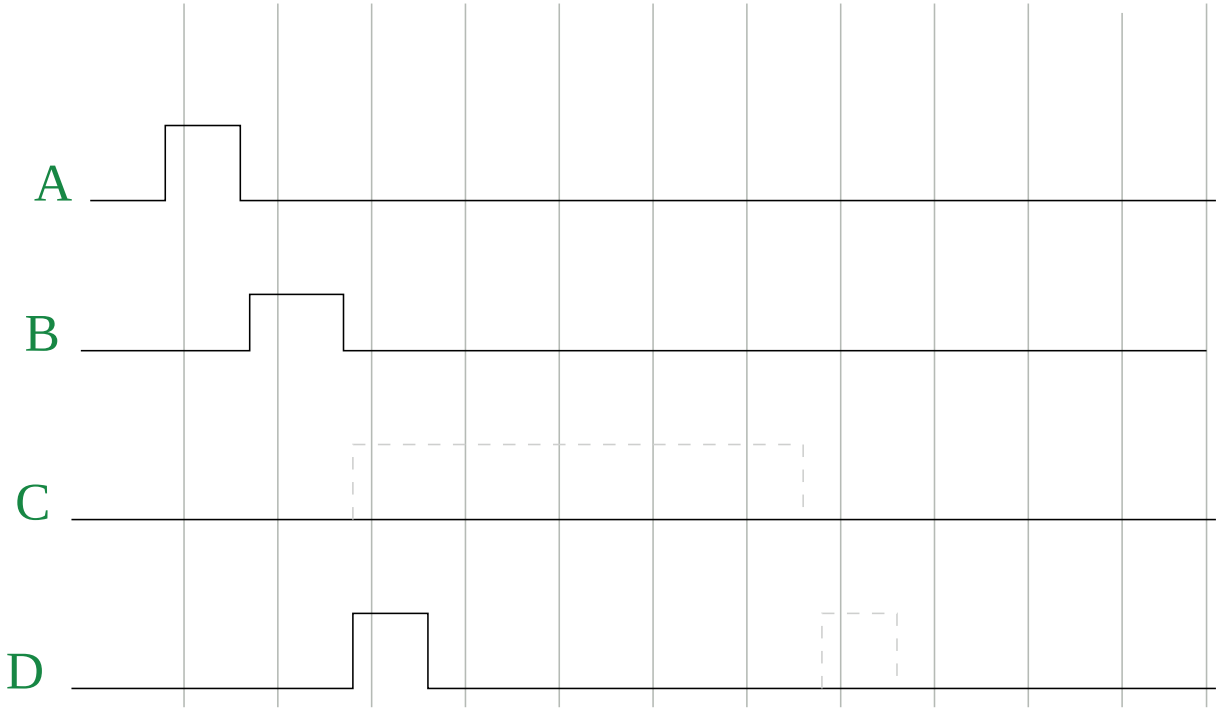
PSL property : **always A -> next { B ; C [*]; D}**



Introductory example

Example

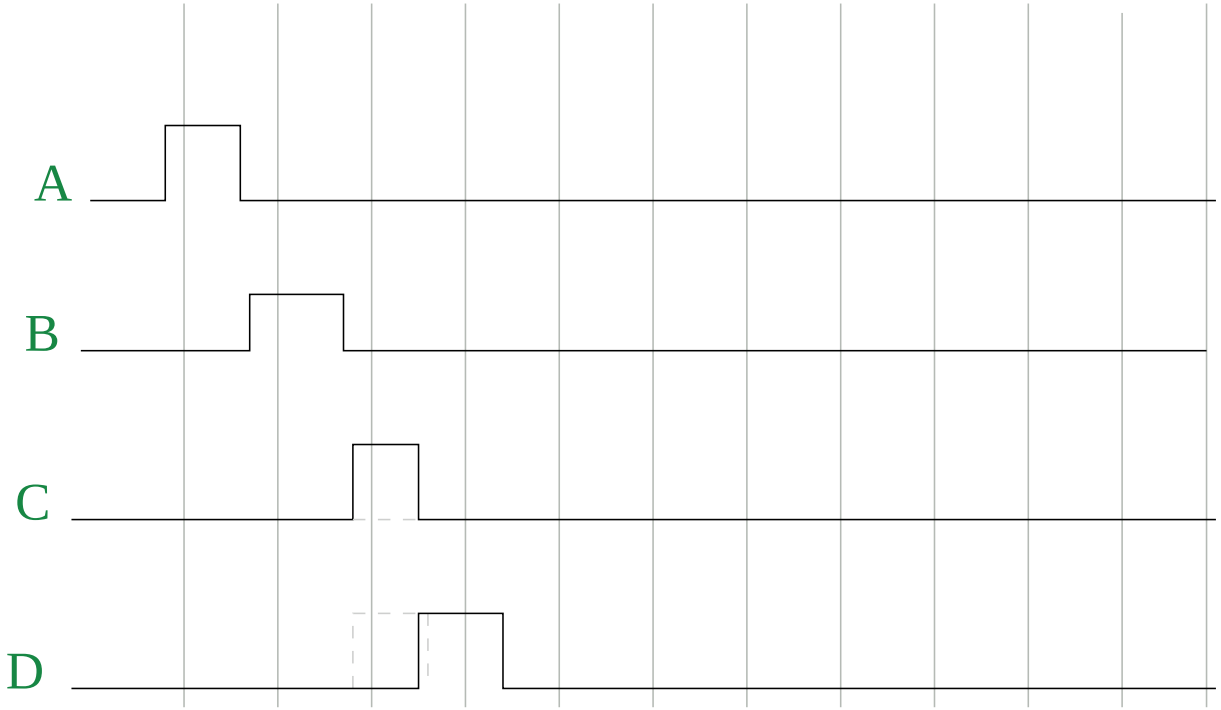
PSL property : **always A -> next { B ; C [*]; D}**



Introductory example

Example

PSL property : **always A -> next { B ; C [*]; D}**



Outlines

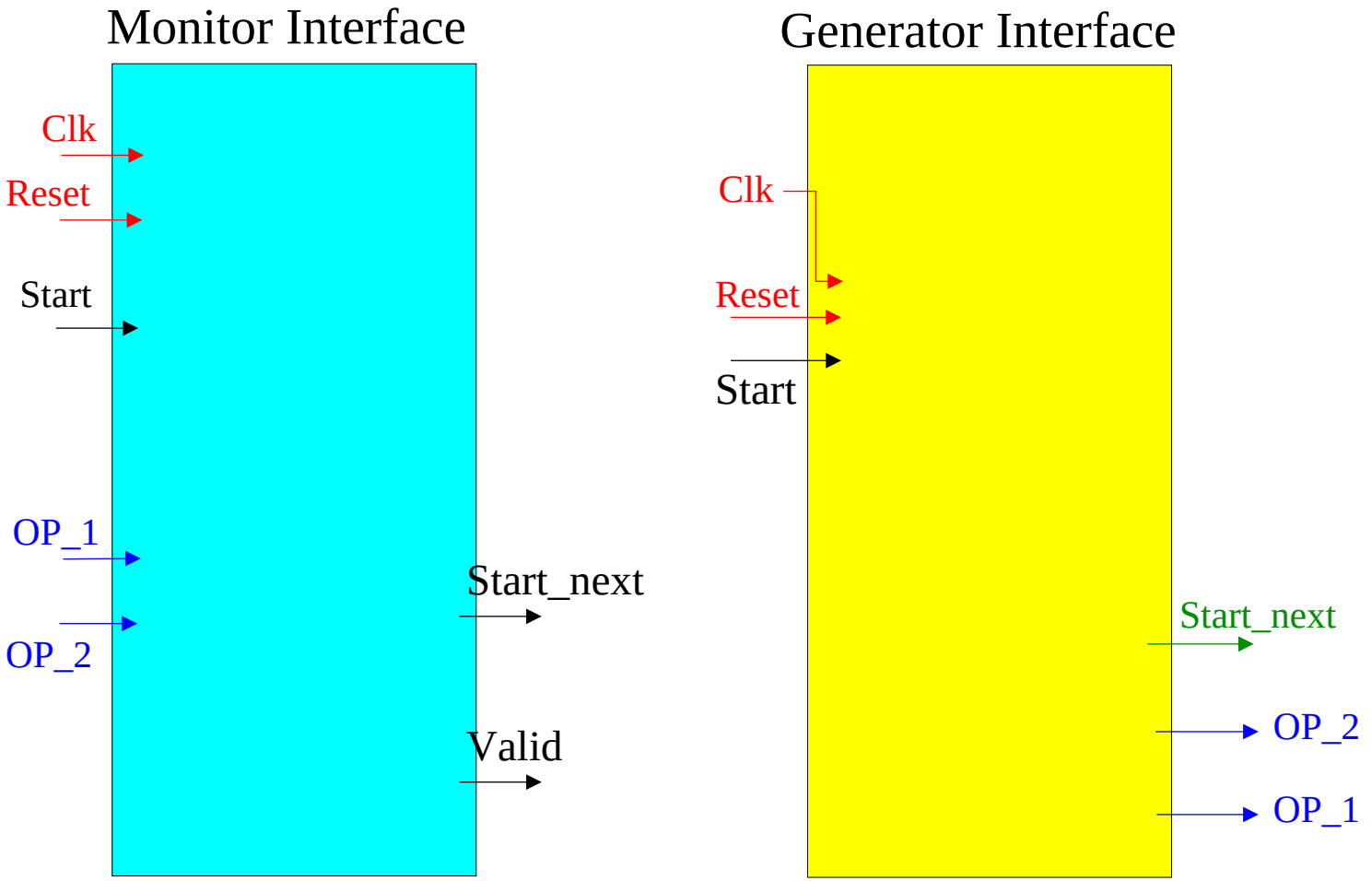
1. PSL
2. Monitors and Generators Construction
3. Experimental Results
4. The Horus Platform

The method

Based on

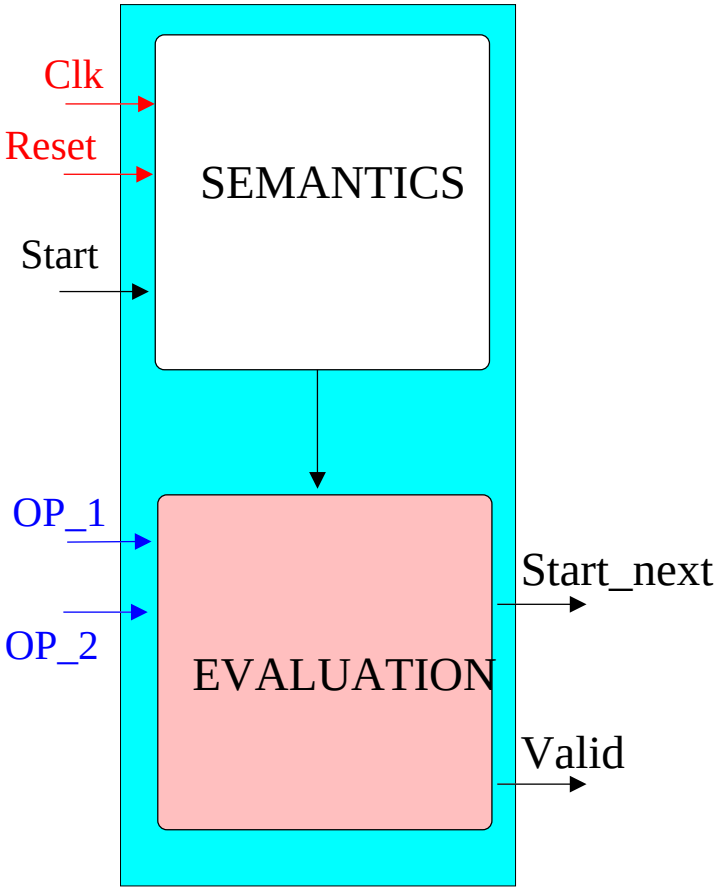
- Elementary components library
- Interconnection scheme: Syntax directed

Primitive Component

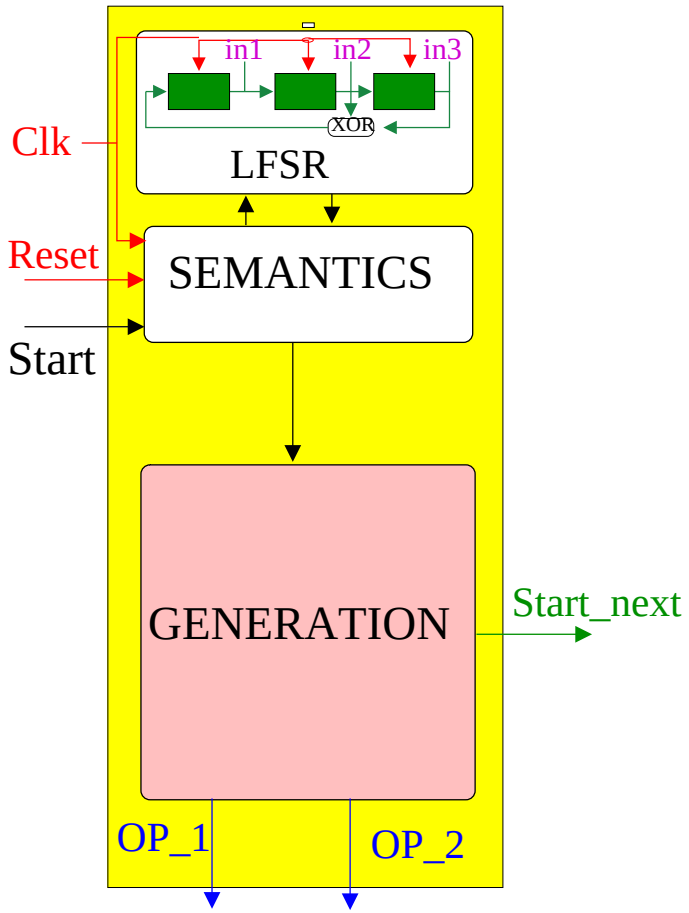


Primitive Component

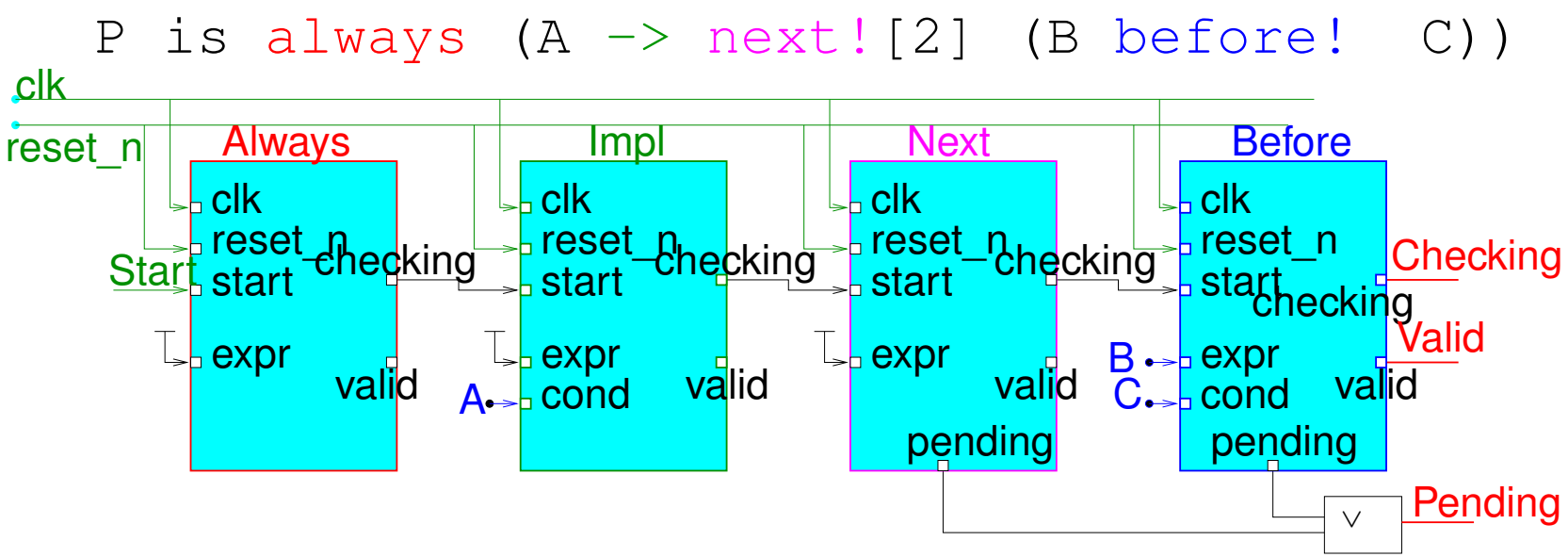
Monitor Architecture



Generator Architecture

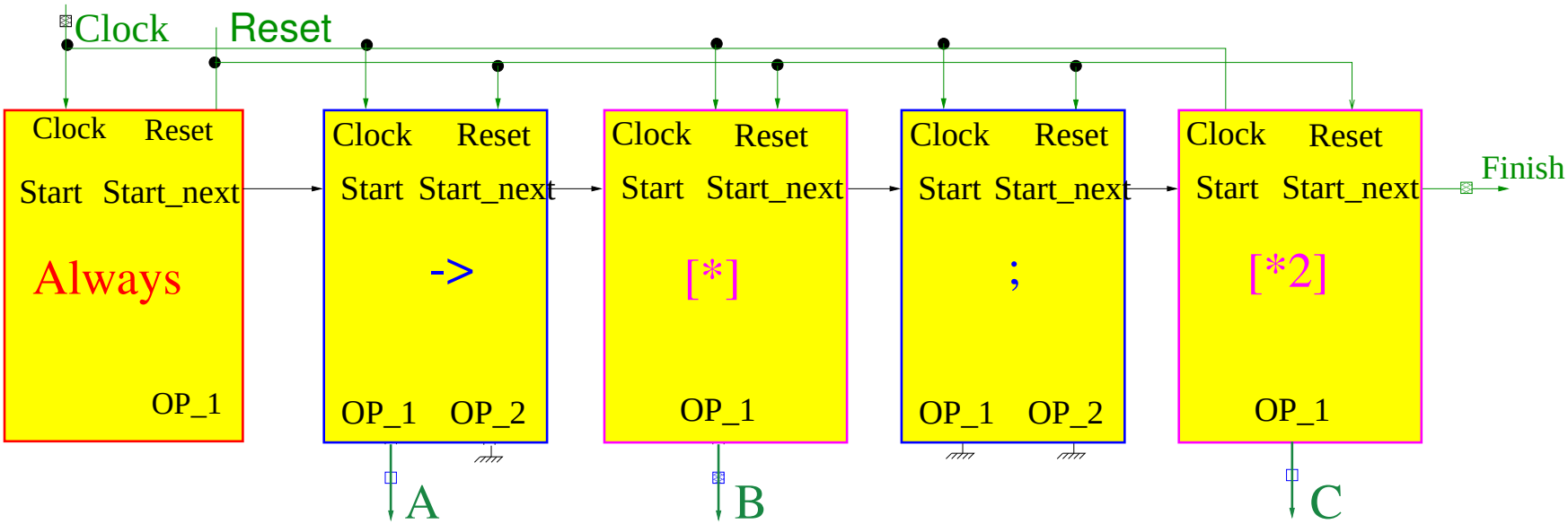


A complex monitor



A Complex Generator

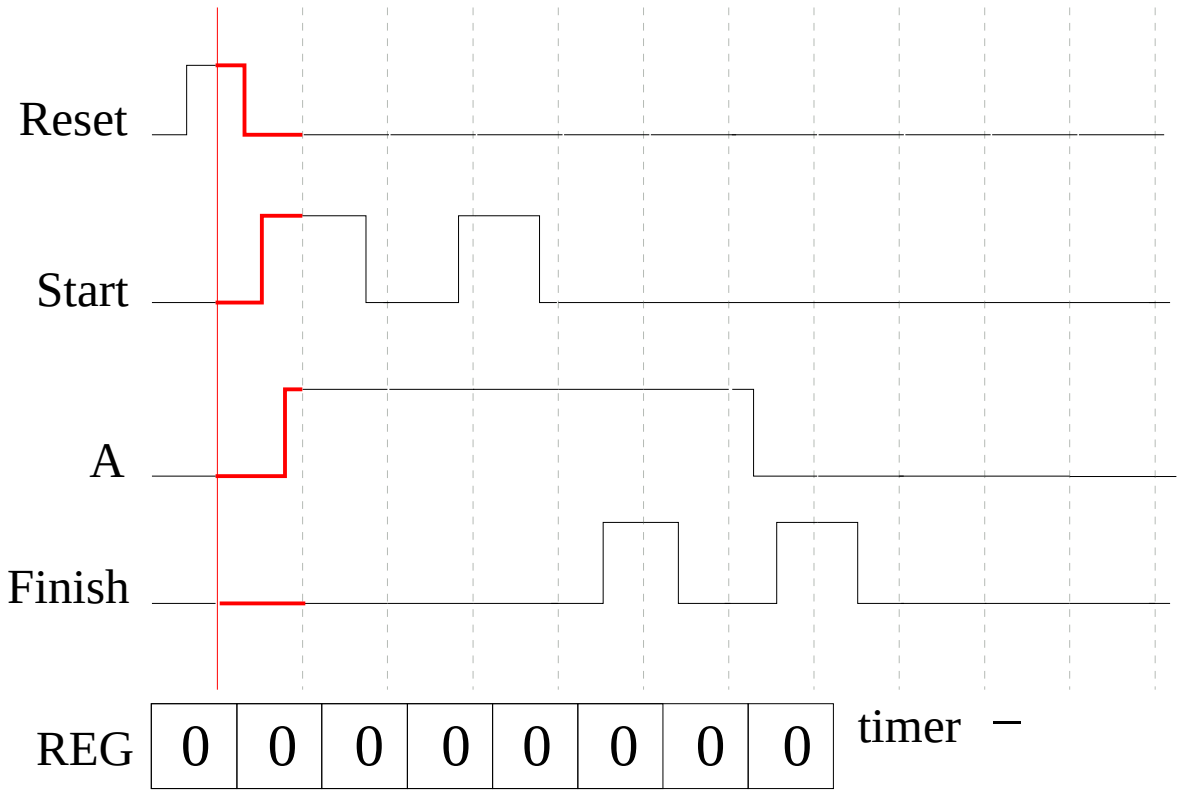
always A -> { B[*] ; C[*2]}



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
    REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else     As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else     Start_next<='0'
end if
end if
```

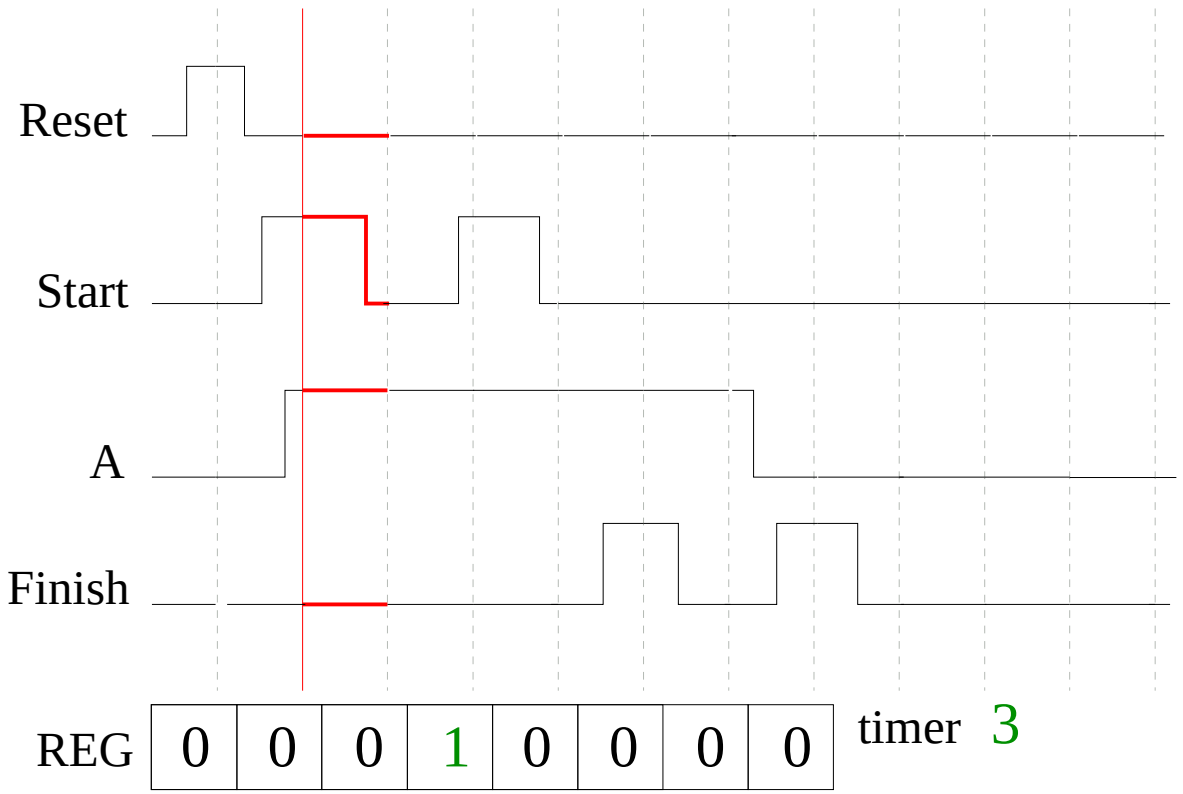
A<=Start or As;



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
    REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else     As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else     Start_next<='0'
end if
end if
```

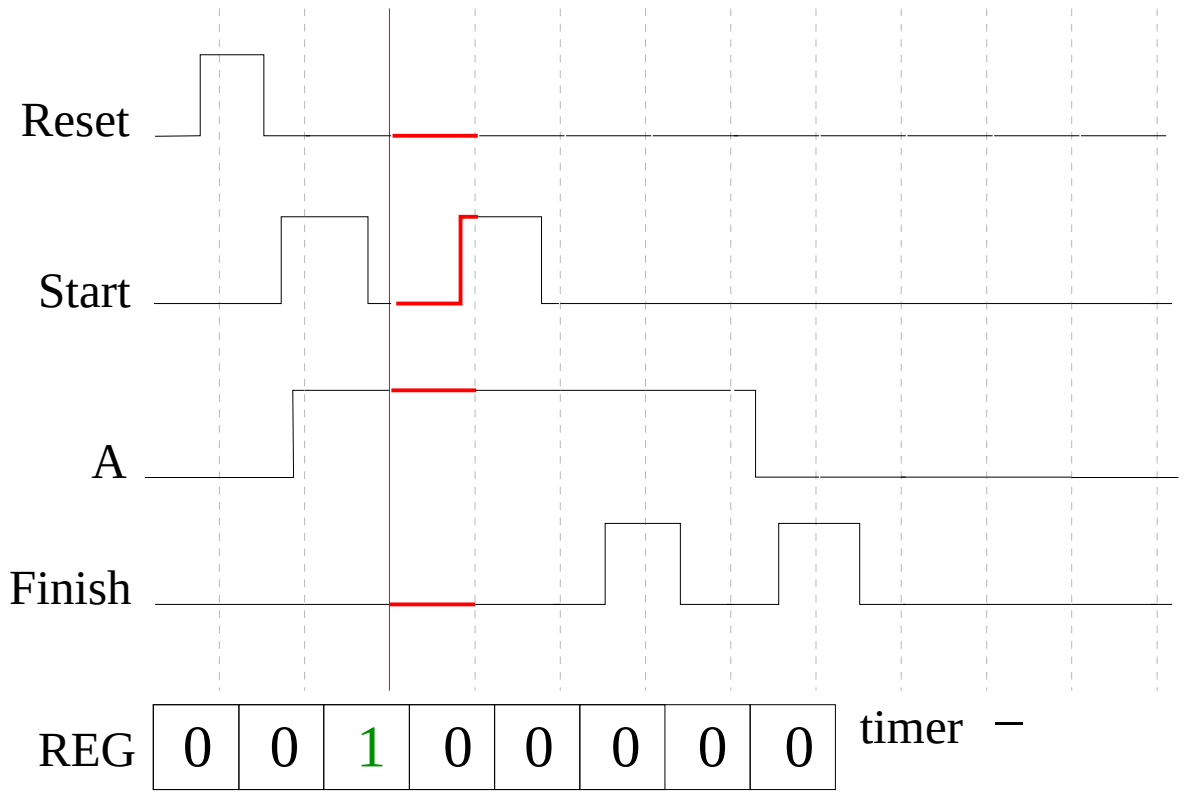
A<=Start or As;



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
    REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else     As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else     Start_next<='0'
end if
end if
```

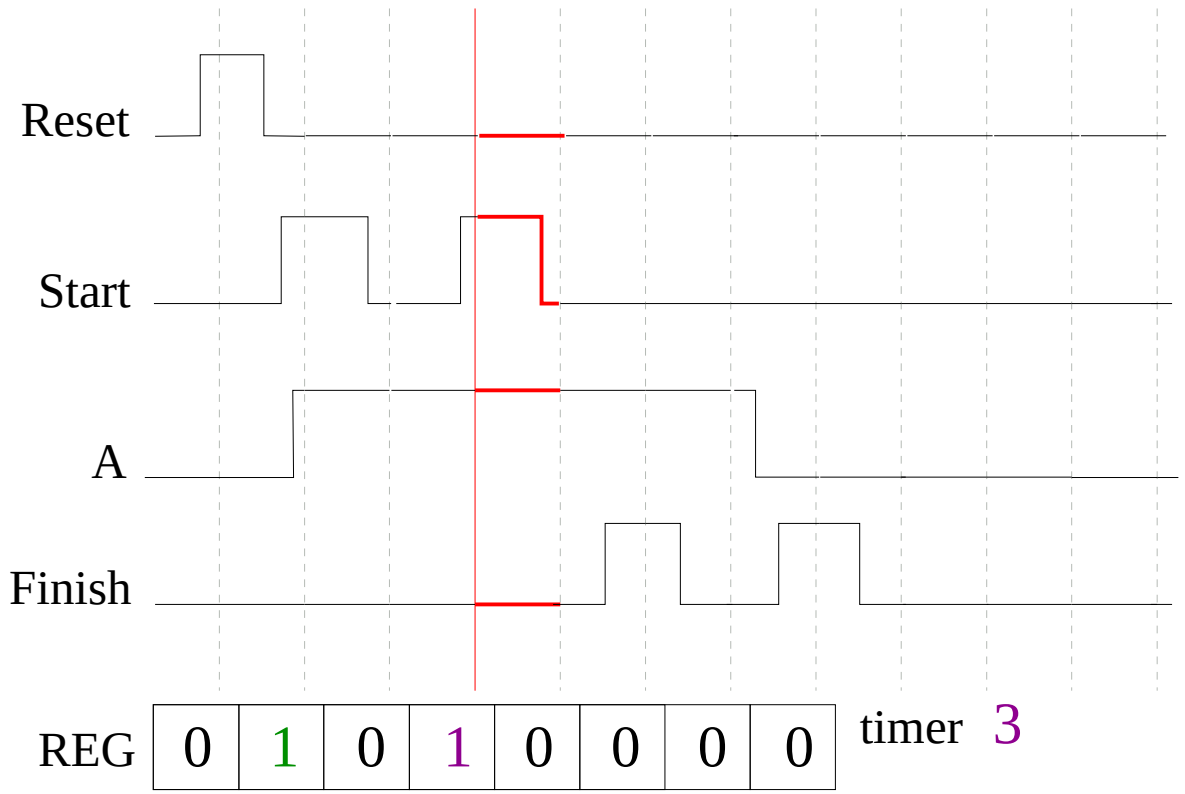
A<=Start or As;



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
   REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else     As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else     Start_next<='0'
end if
end if
```

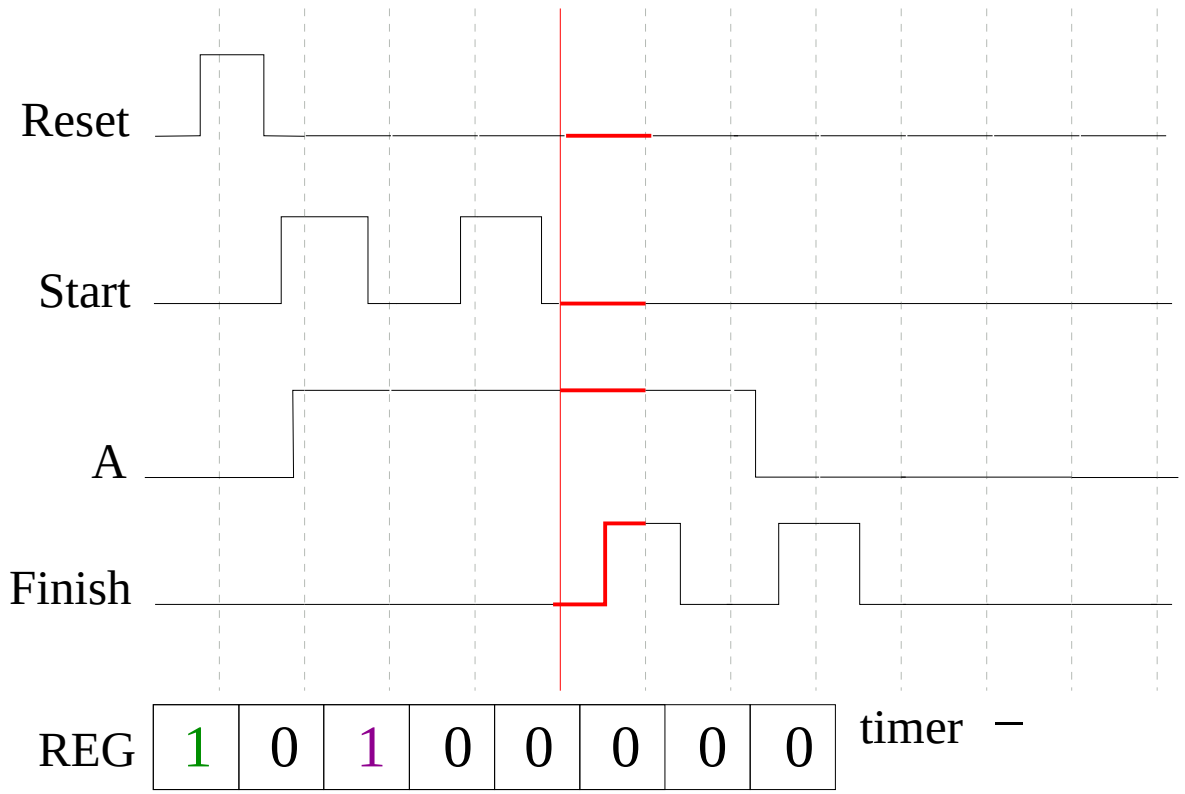
A<=Start or As;



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
    REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else      As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else      Start_next<='0'
end if
end if
```

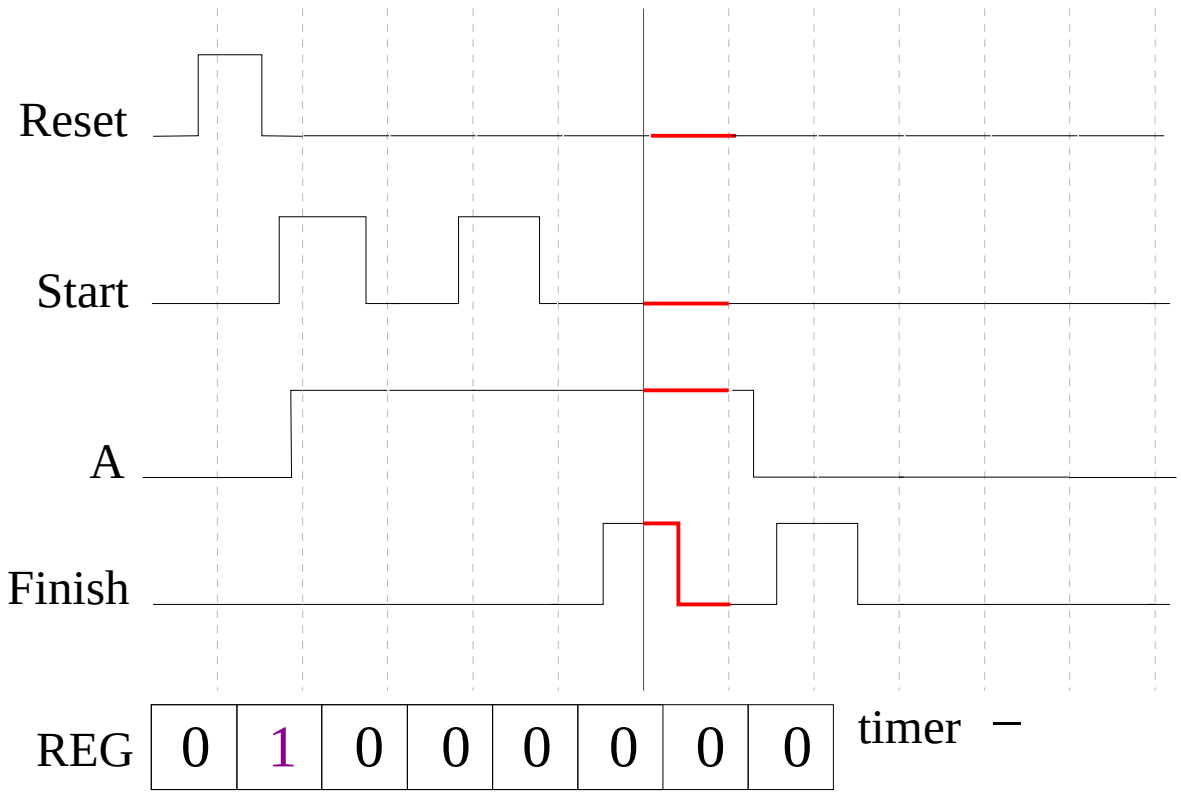
A<=Start or As;



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
    REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else     As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else     Start_next<='0'
end if
end if
```

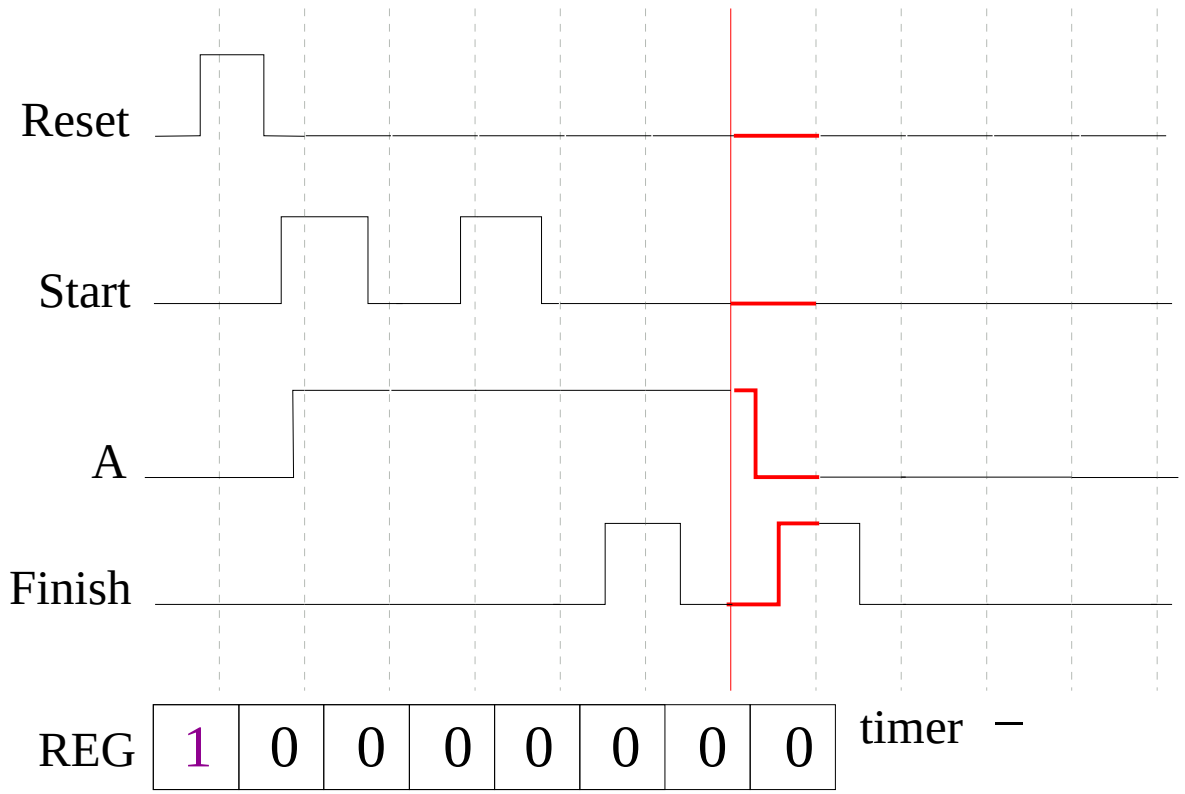
A<=Start or As;



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
    REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else     As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else     Start_next<='0'
end if
end if
```

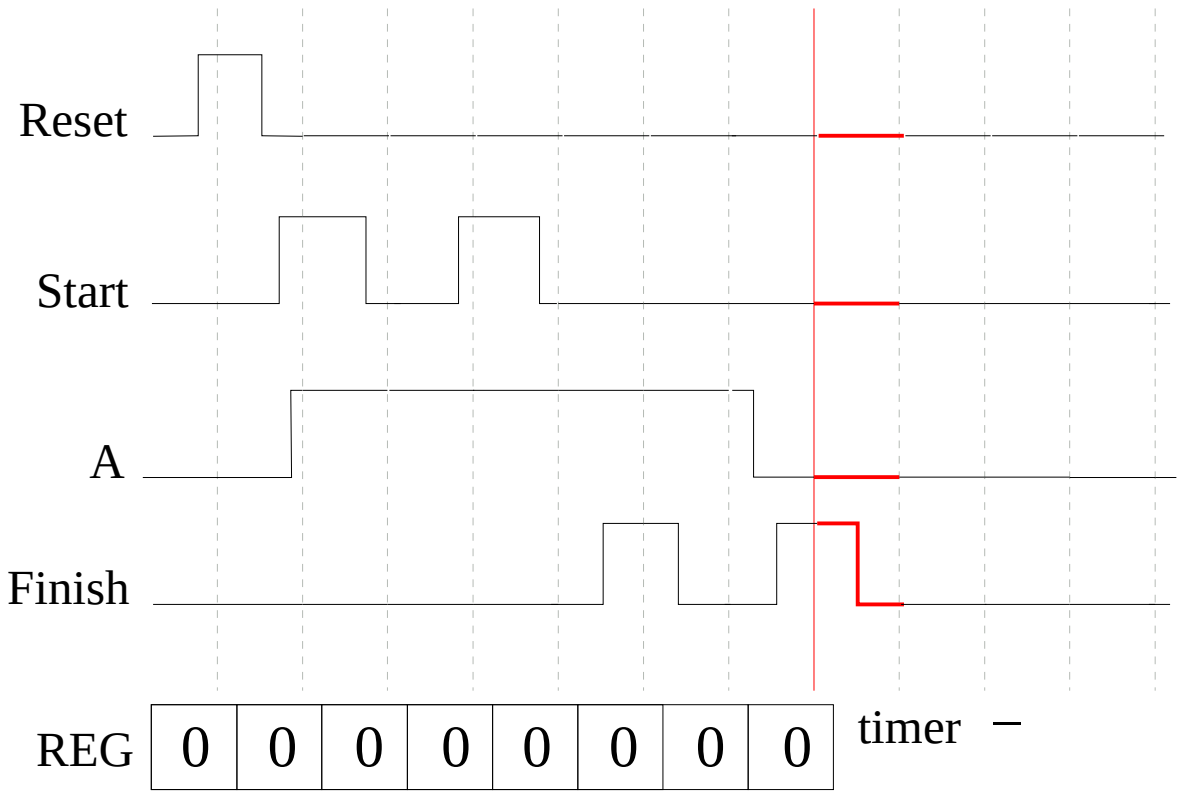
A<=Start or As;



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
    REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else     As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else     Start_next<='0'
end if
end if
```

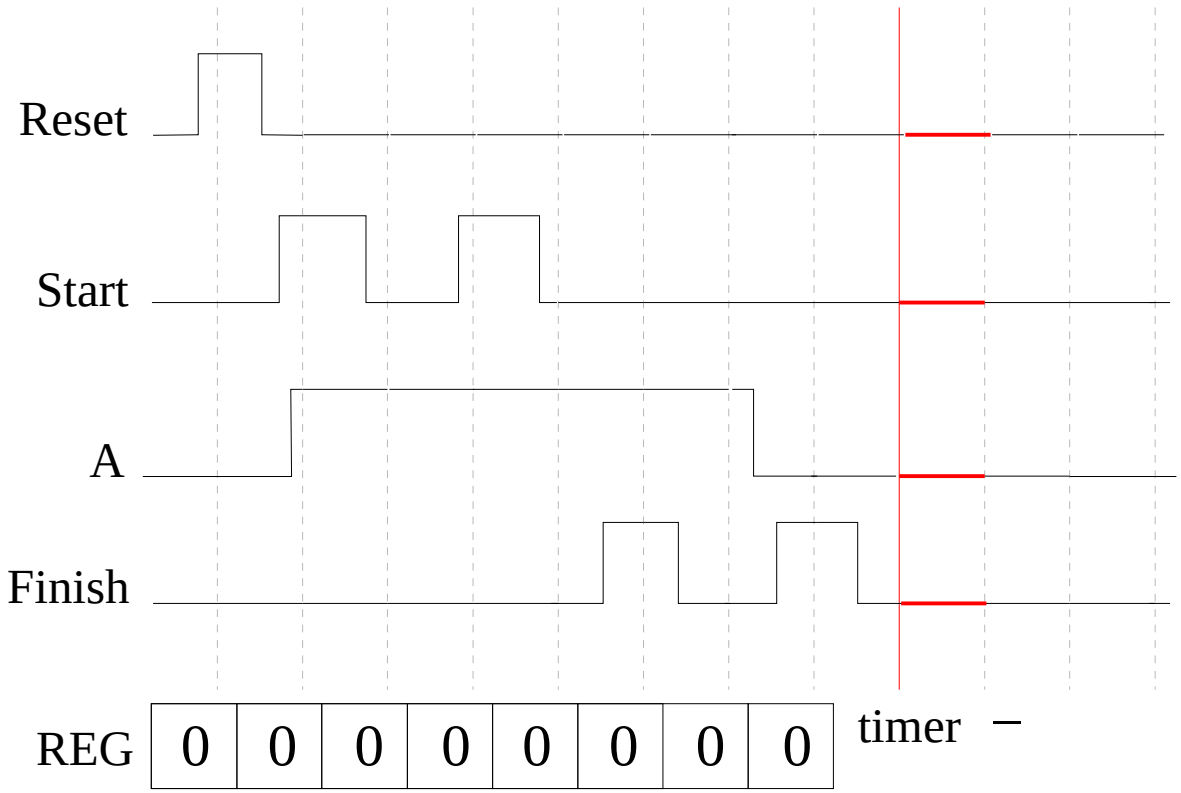
A<=Start or As;



An example: {A[+]}

```
If Reset='1' then
    As<='0';Start_next<='0'
else if Start='1' then
    REG(timer):='1'
end if
if REG=ONE or
REG=ZERO then
    As<='0'
else     As<='1'
end if
if REG(0)='1' then
    Start_next<='1'
else     Start_next<='0'
end if
end if
```

A<=Start or As;



Validation

Monitors:

- FL: formal verification of the library and the interconnection scheme with a theorem prover (PVS)
- SERE : definition of a formal model based on Petri net, formal verification in progress.

Generator

- Simulation of a monitor and a generator
- Model Checking of the library
- To do: formal verification

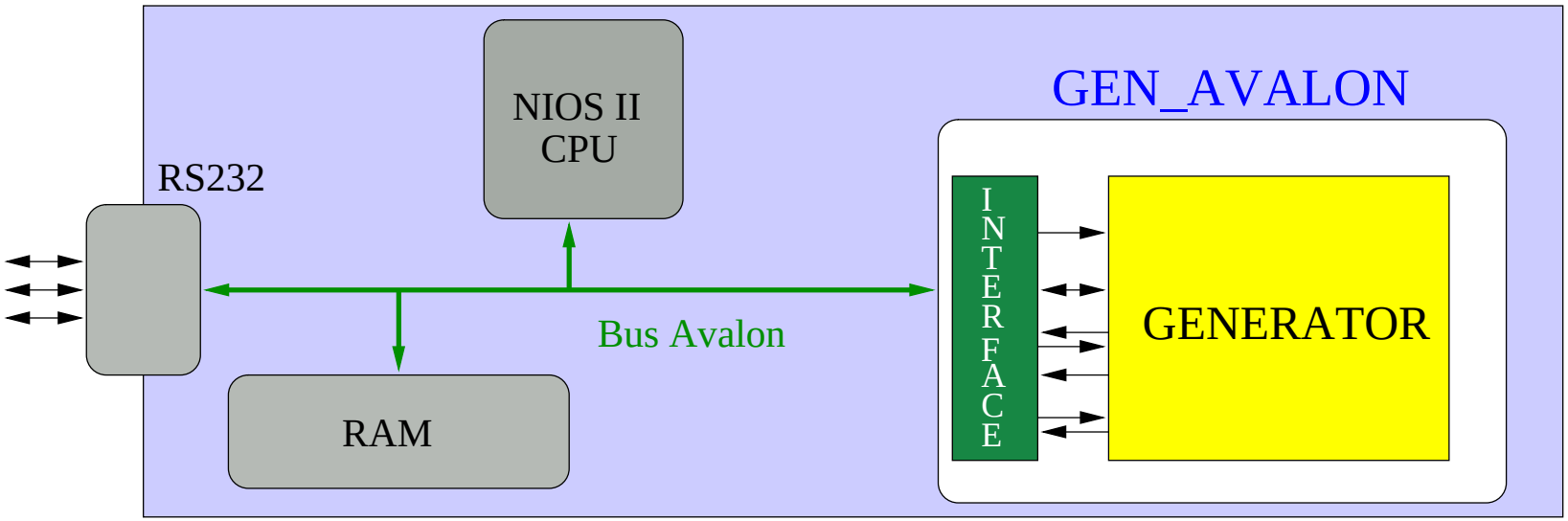
Outlines

1. PSL
2. Monitors and Generators Construction
3. Experimental Results
4. The Horus Platform

Emulation: NIOS Architecture

Complexity: CPU+RS232+RAM

- Logic Cells: 2126
- Registers: 1036
- Frequency: 50 Mhz



Synthesis Results

Monitor

```
always a -> ({b;c} until!  d)
```

- Reg: 5, LC: 8, Freq.: 439 MHz

Generator

```
always a -> next_a[4 to 8](a and b)
```

- Reg: 29, LC: 41, Freq.: 78 Mhz

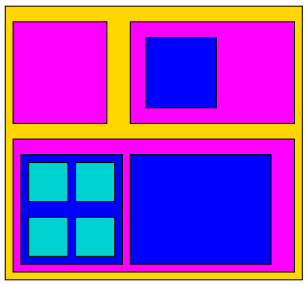
Outlines

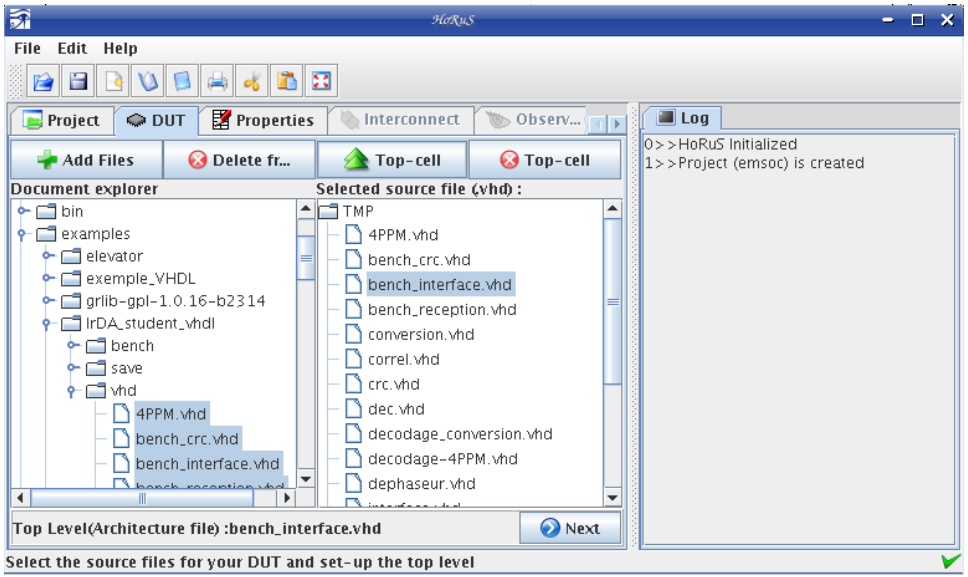
1. PSL
2. Monitors and Generators Construction
3. Experimental Results
4. The Horus Platform

Principle

Step 1 : Load the files of the DUT

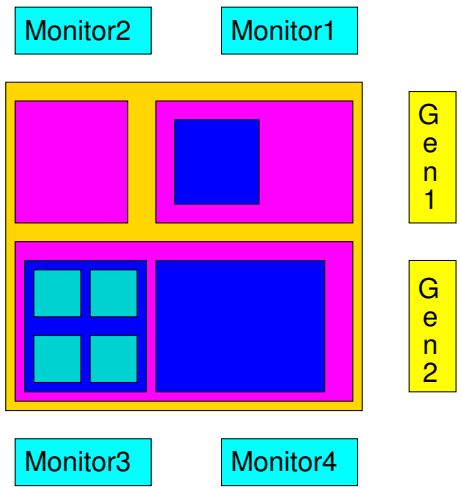
TOP Level

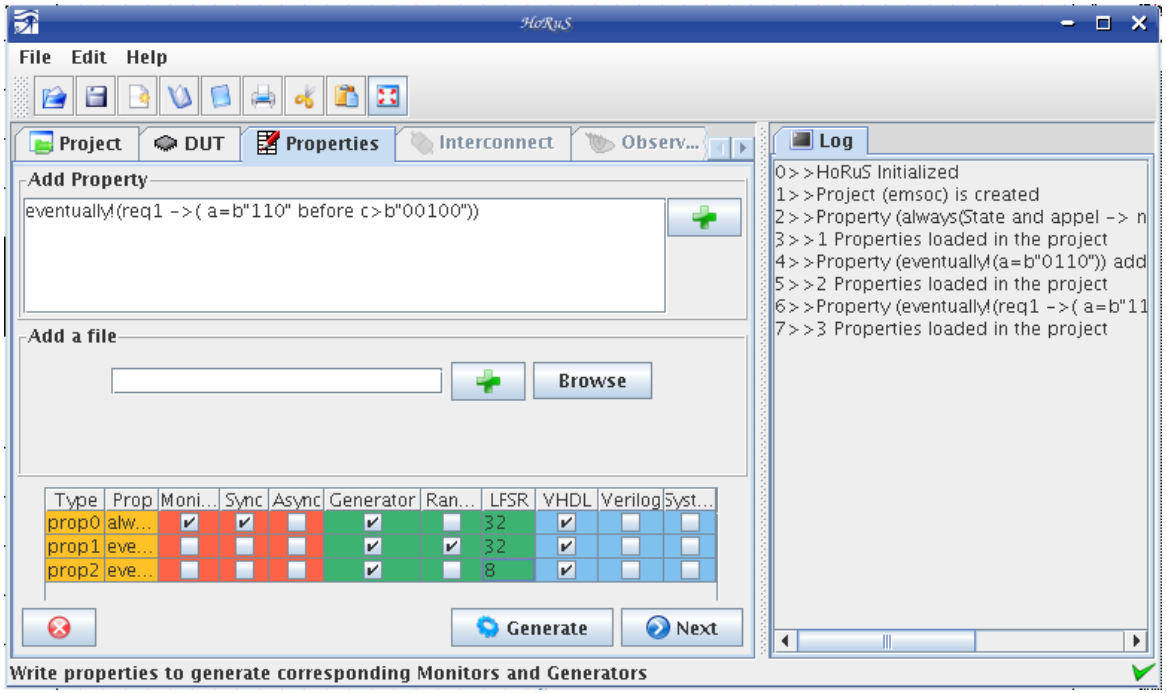




Principle

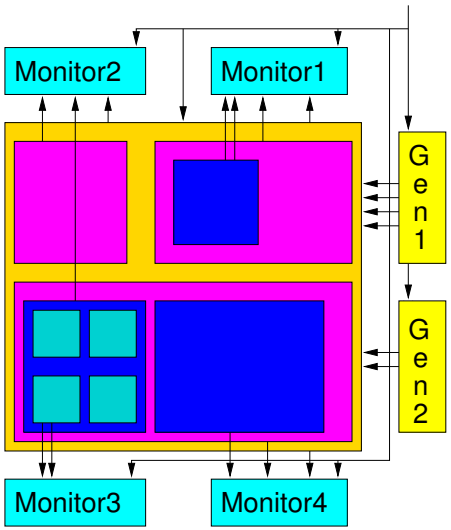
Step 2 : Produce monitors and generators from properties

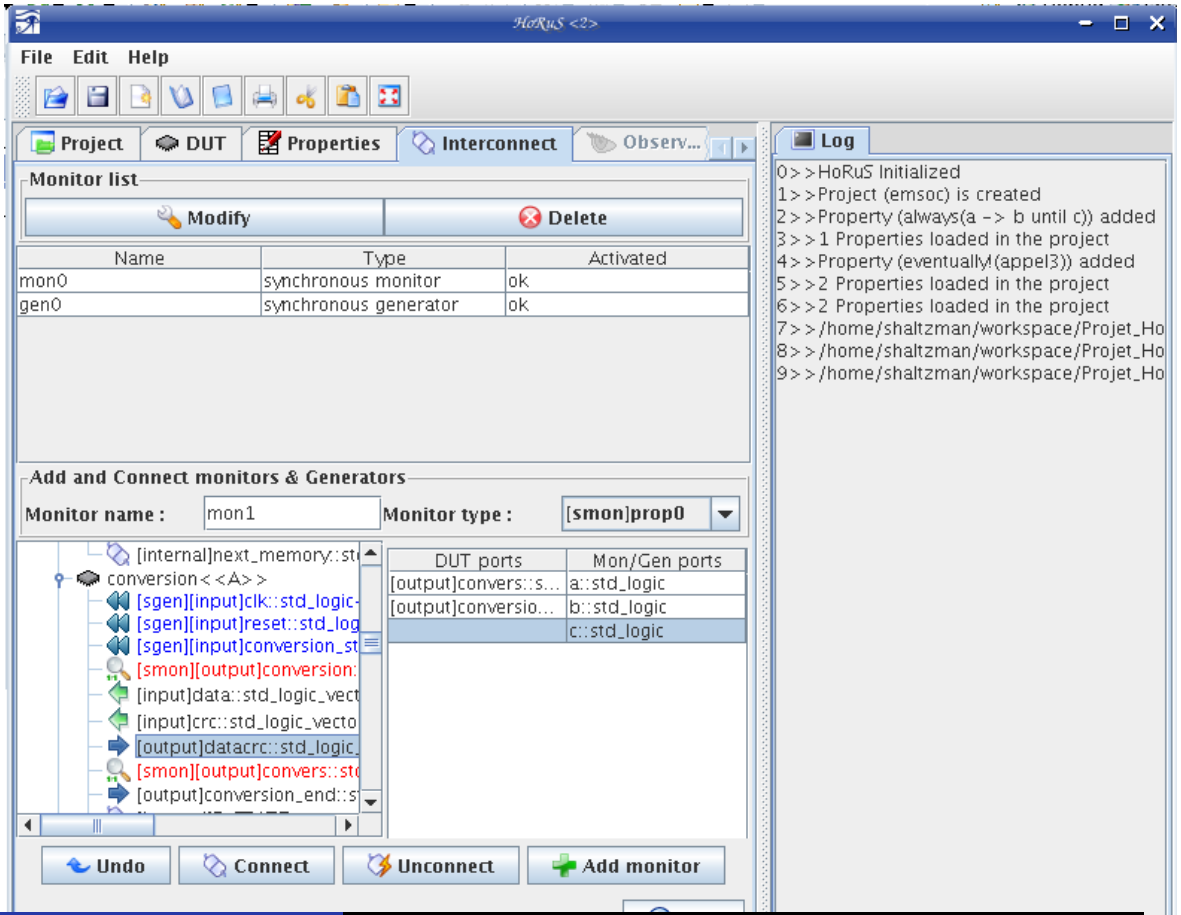




Principle

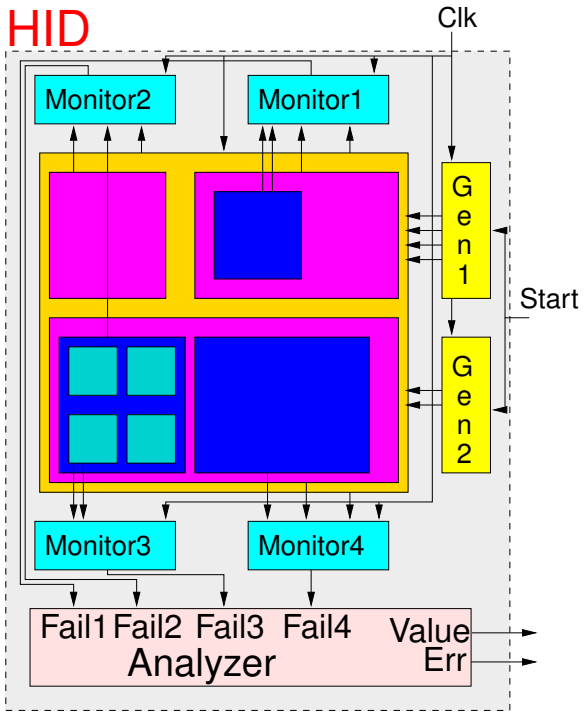
Step 3 : Connect monitors and generators to the DUT

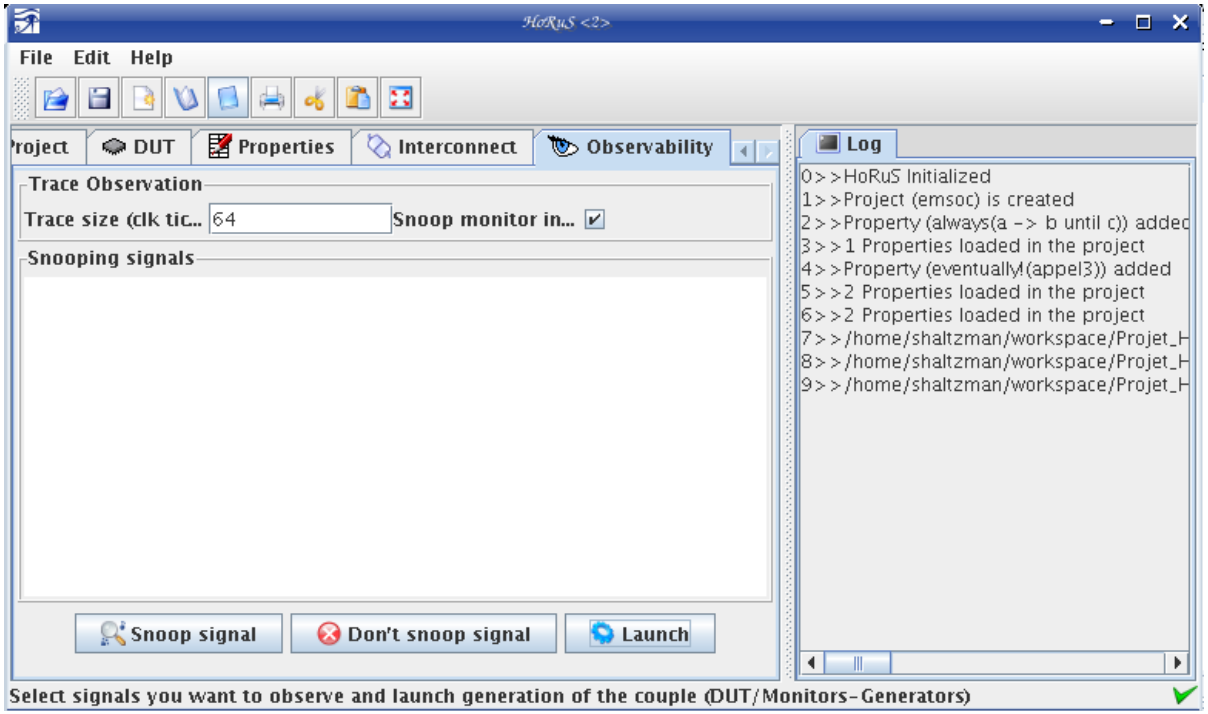




Principle

Step 4 : Build the **H**ardware **I**nstrumented **D**esign





Conclusion

- A method to generate hadware monitoror generator for PSL operators
- A platform to automatically interconnect and instrument designs
- Comparison with other tools (Focs, Dialite) for area. Our monitors are smaller

Futur Works

- Proof of correctness of generators
- Combining monitors and generators to synthesize designs
- Debug
- Extension to TLM
- Asynchronous monitoring