Clocking Schemes in Esterel

Laurent Arditi, Gérard Berry, Marc Perraut

Esterel Technologies

Mike Kishinevsky

Intel

www.esterel-technologies.com
Gerard.Berry@esterel-technologies.com

Goals

- Remove current single-clock limitation of Esterel
- Support clock gating at functional level
- Support GALS-like multiclock design
- Maintain mathematical semantics and formal verification
- Synthesize to multiclock ASICs or to single-clock designs

(ASICS & software / FPGA simulations from single source)

Available in current Esterel Studio 5.3
=> get our free university program!

Clock Gating

- Usually considered as a way to save power
- Partly automated by pattern-matching

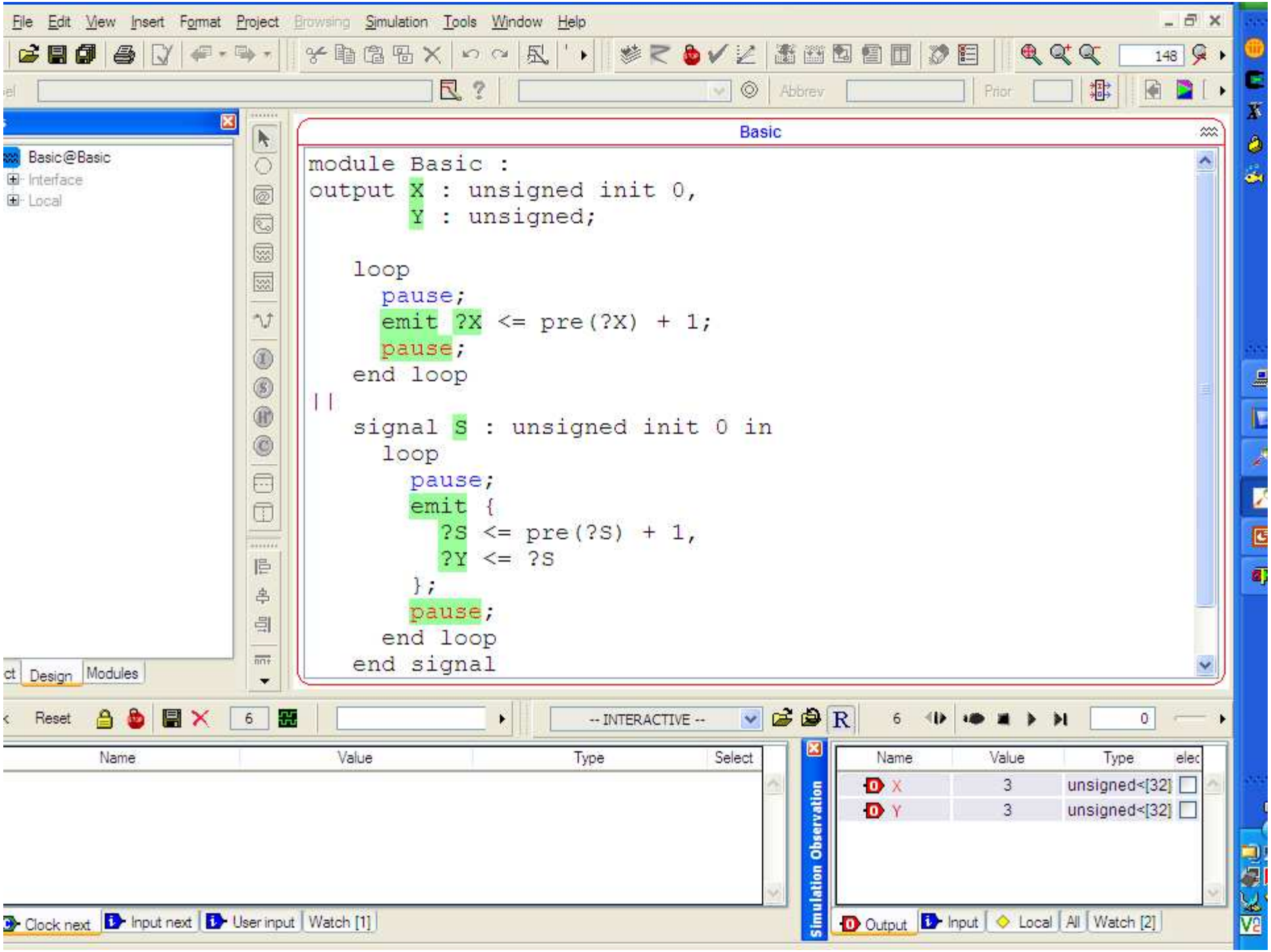
  Synopsys power compiler
- Not really in the RTL model

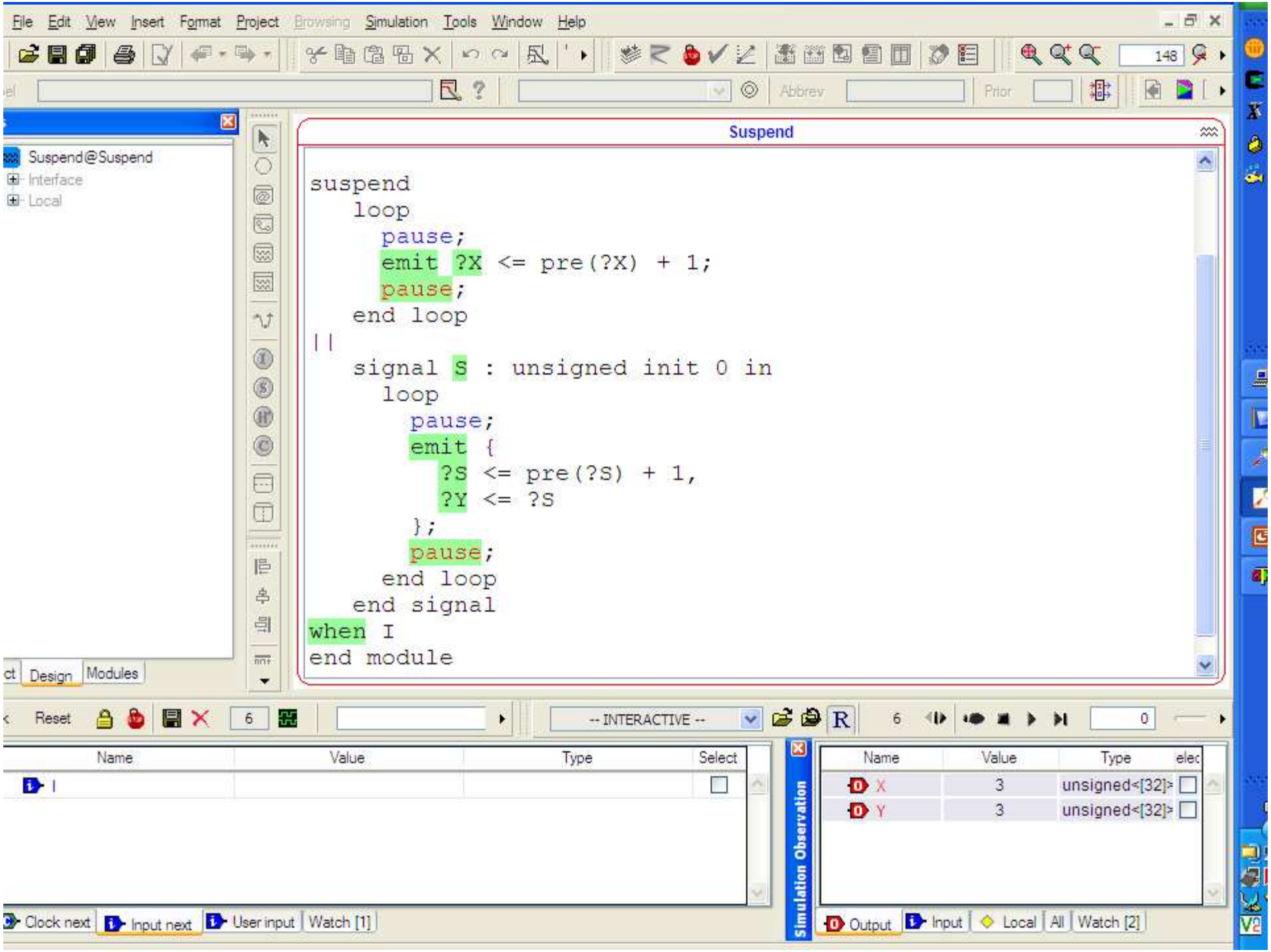
Esterel view

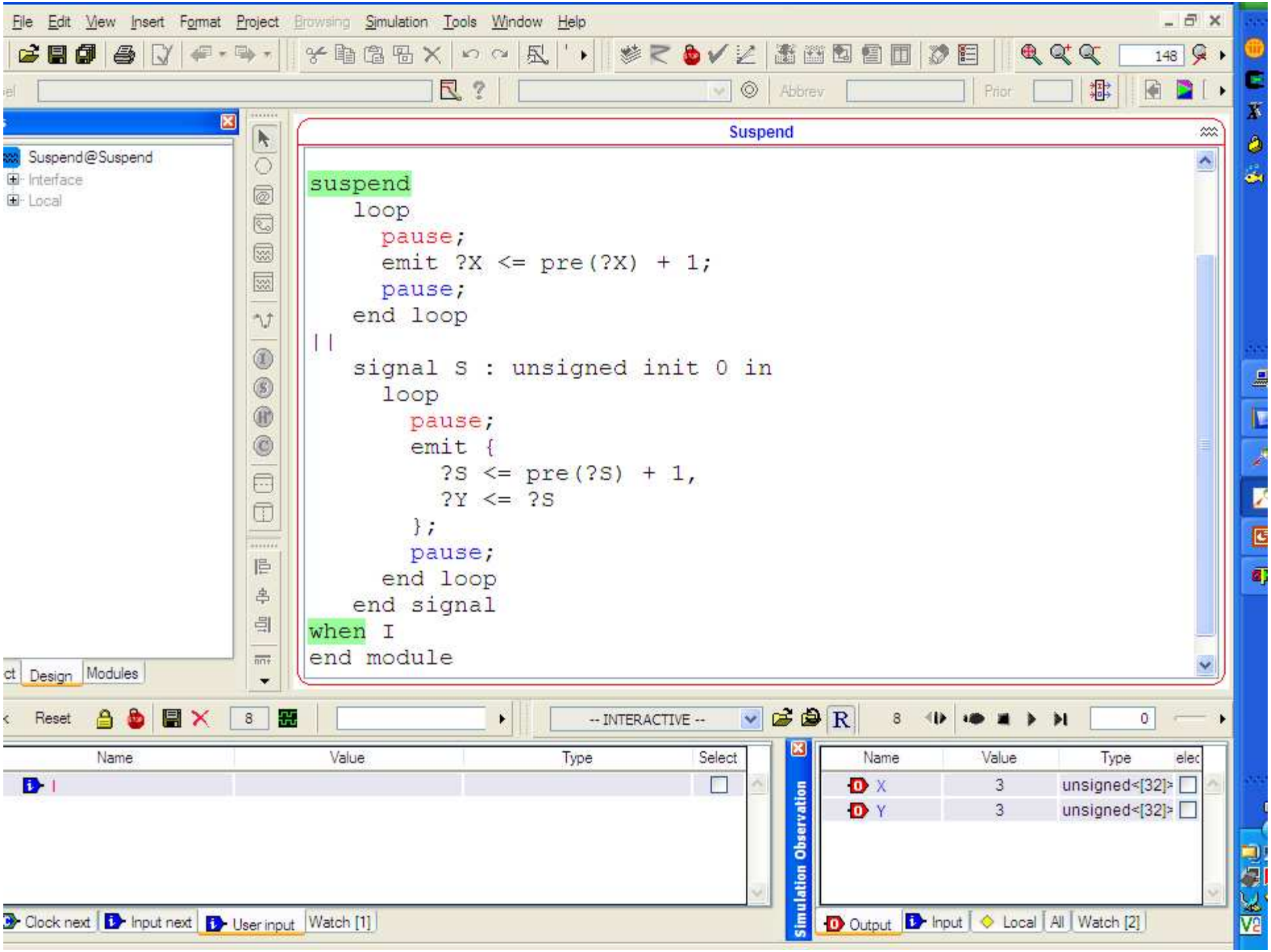
- Results from new weak suspend statement (K. Schneider)

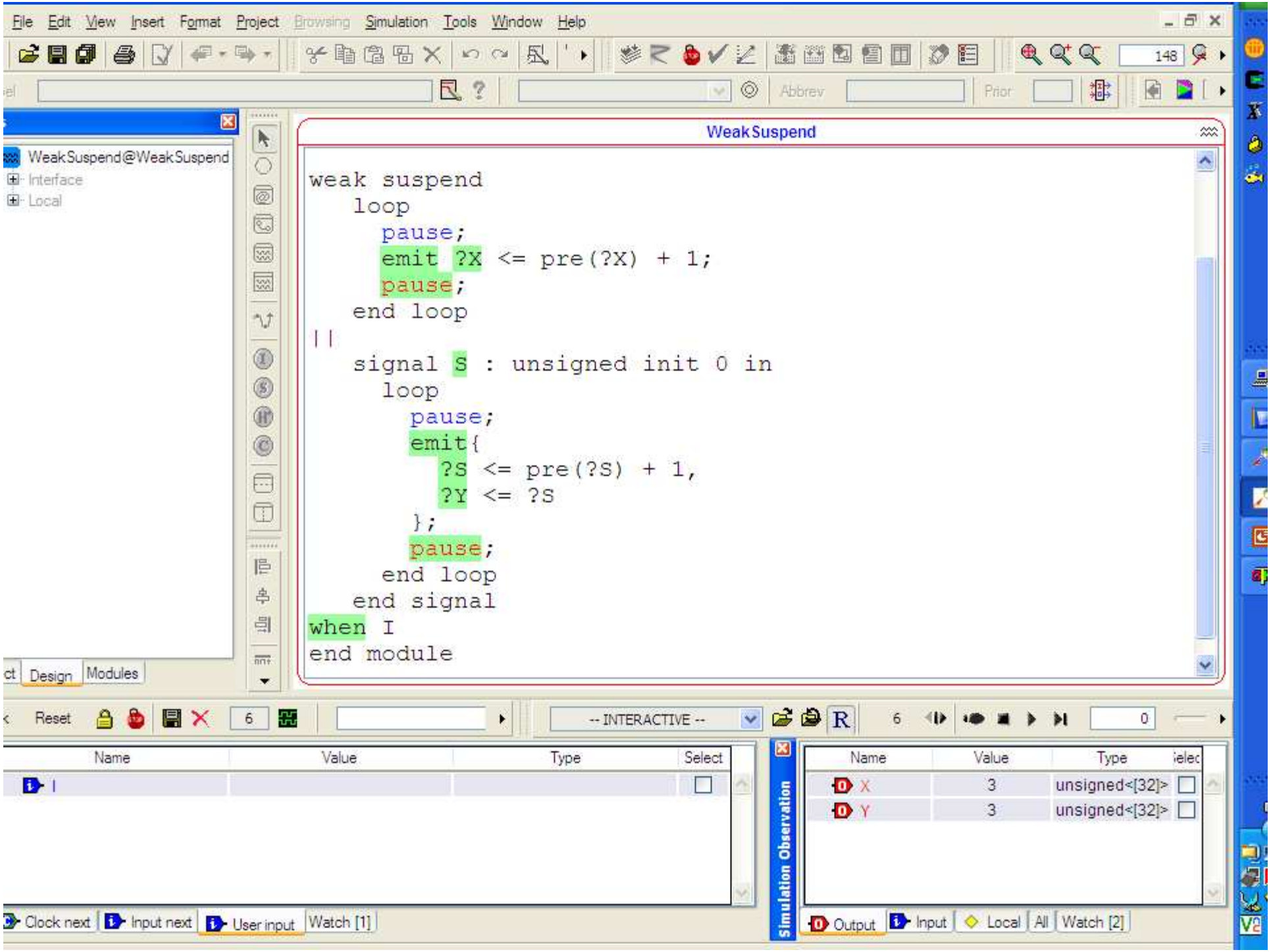
  combinational transition performed, but no state change
- Semantics ok, fits well with scoping

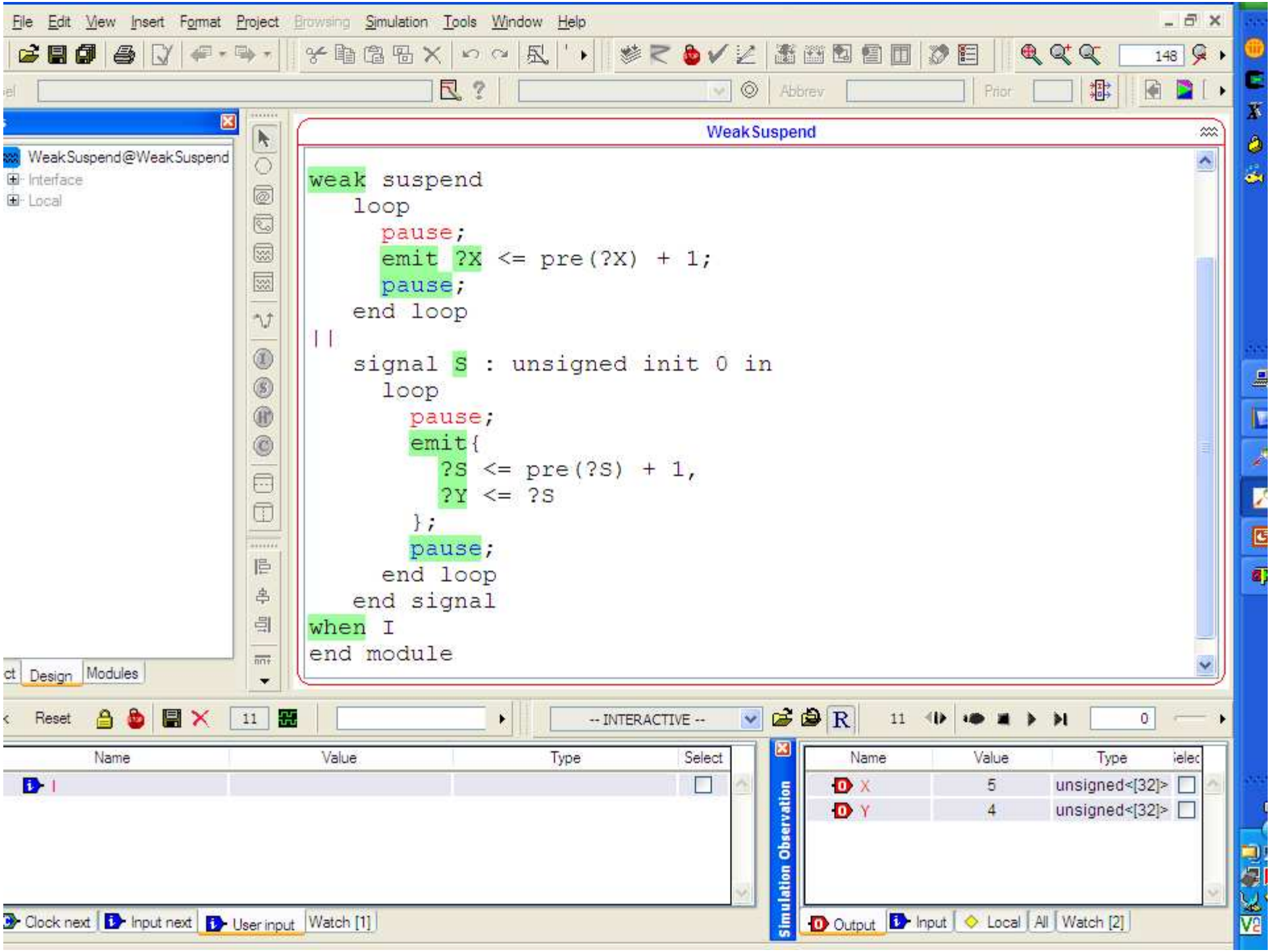
  applies to states of all control and objects declared inside
- Synthesis to actual clock gating or to enabling logic


File Edit View Insert Format Project Browsing Simulation Tools Window Help
Basic@Basic
Interface
Local
Design
Modules
Basic
module Basic :
output X : unsigned init 0,
Y : unsigned;
loop
pause;
emit ?X <= pre(?X) + 1;
pause;
end loop
||
signal S : unsigned init 0 in
loop
pause;
emit {
?S <= pre(?S) + 1,
?Y <= ?S
};
pause;
end loop
end signal
Reset
-- INTERACTIVE --
Name
Value
Type
Select
Clock next
Input next
User input
Watch [1]
Simulation Observation
Name
Value
Type
elec
X
3
unsigned<[32]
Y
3
unsigned<[32]
Output
Input
Local
All
Watch [2]


File Edit View Insert Format Project Browsing Simulation Tools Window Help
Suspend@Suspend
Interface
Local
Design
Modules
Suspend
suspend
loop
pause;
emit ?X <= pre(?X) + 1;
pause;
end loop
||
signal S : unsigned init 0 in
loop
pause;
emit {
?S <= pre(?S) + 1,
?Y <= ?S
};
pause;
end loop
end signal
when I
end module
Reset
-- INTERACTIVE --
Name Value Type Select
I
Clock next
Input next
User input
Watch [1]
Simulation Observation
Name Value Type
X 3 unsigned<[32]>
Y 3 unsigned<[32]>
Output
Input
Local
All
Watch [2]


File Edit View Insert Format Project Browsing Simulation Tools Window Help
Suspend@Suspend
Interface
Local
Design
Modules
Suspend
suspend
loop
pause;
emit ?X <= pre(?X) + 1;
pause;
end loop
||
signal S : unsigned init 0 in
loop
pause;
emit {
?S <= pre(?S) + 1,
?Y <= ?S
};
pause;
end loop
end signal
when I
end module
Reset
-- INTERACTIVE --
Name
Value
Type
Select
I
Clock next
Input next
User input
Watch [1]
Simulation Observation
Name
Value
Type
X
3
unsigned<[32]>
Y
3
unsigned<[32]>
Output
Input
Local
All
Watch [2]


File Edit View Insert Format Project Browsing Simulation Tools Window Help
WeakSuspend@WeakSuspend
Interface
Local
Design
Modules
Weak Suspend
weak suspend
loop
pause;
emit ?X <= pre(?X) + 1;
pause;
end loop
||
signal S : unsigned init 0 in
loop
pause;
emit{
?S <= pre(?S) + 1,
?Y <= ?S
};
pause;
end loop
end signal
when I
end module
Reset
-- INTERACTIVE --
Name
Value
Type
Select
I
Clock next
Input next
User input
Watch [1]
Simulation Observation
Name
Value
Type
X
3
unsigned<[32]>
Y
3
unsigned<[32]>
Output
Input
Local
All
Watch [2]


File Edit View Insert Format Project Browsing Simulation Tools Window Help
WeakSuspend@WeakSuspend
Interface
Local
Design
Modules
Weak Suspend
weak suspend
loop
pause;
emit ?X <= pre(?X) + 1;
pause;
end loop
||
signal S : unsigned init 0 in
loop
pause;
emit{
?S <= pre(?S) + 1,
?Y <= ?S
};
pause;
end loop
end signal
when I
end module
Reset
-- INTERACTIVE --
Name
Value
Type
Select
I
Clock next
Input next
User input
Watch [1]
Simulation Observation
Name
Value
Type
X
5
unsigned<[32]>
Y
4
unsigned<[32]>
Output
Input
Local
All
Watch [2]

Immediate Weak Suspension

- by default, suspension ignored at start instant
- immediate variant to handle start instant

```
weak suspend
  p
when immediate exp
```

```
trap Done in
   loop
      trap Immediate in
         {
            p
         ||
            if exp then exit Immediate
         }
         exit Done
      end trap;
      pause
   end loop
end trap
```

Suspend Formal Semantics

$$\frac{s \notin E \quad p \xrightarrow[E]{E' \;\; k} p'}{s \supset p \xrightarrow[E]{E' \;\; k} s \supset p'}$$

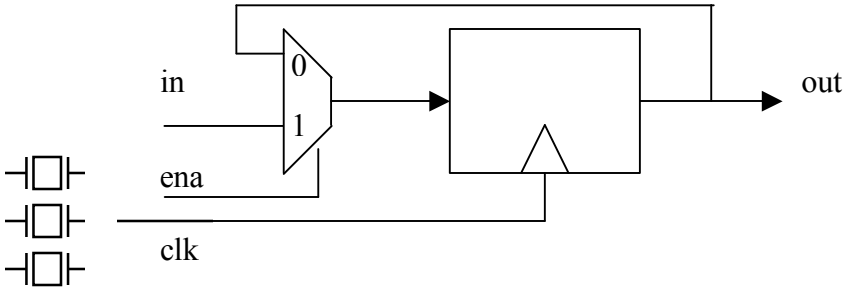
$$\frac{s \in E}{s \supset p \xrightarrow[E]{\emptyset \;\; 1} s \supset p}$$

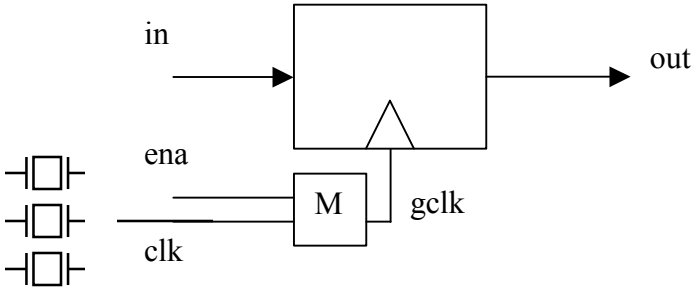
Weak Suspend Formal Semantics

$$\frac{s \notin E \quad p \xrightarrow[E]{E' \;\; k} p'}{s \supseteq p \xrightarrow[E]{E' \;\; k} s \supseteq p'}$$

$$\frac{s \in E \quad p \xrightarrow[E]{E' \;\; k} p'}{s \supseteq p \xrightarrow[E]{E' \;\; \max(k,1)} s \supseteq p}$$

Weak Suspend Implementation





enabling logic

FGGA, software, formal verification

clock gating

ASIC

esterelv7 compiler option

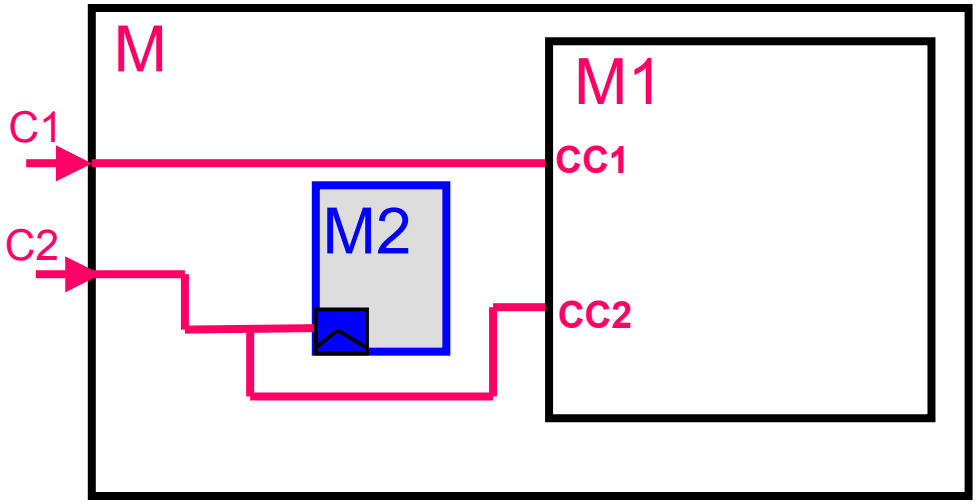
Clocks and Multiclock Units

Clocks

- special signal declared clock
- can clock the states of conventional single-clock modules
- can be downsampled or muxed
- no other combinational or sequential calculation allowed

Multiclock units

- module interface + clock interface
- can only do the following:
  - perform combinational (unclocked) calculations
  - run clocked modules
  - run multiclock units (hierarchy)
  - define new clocks



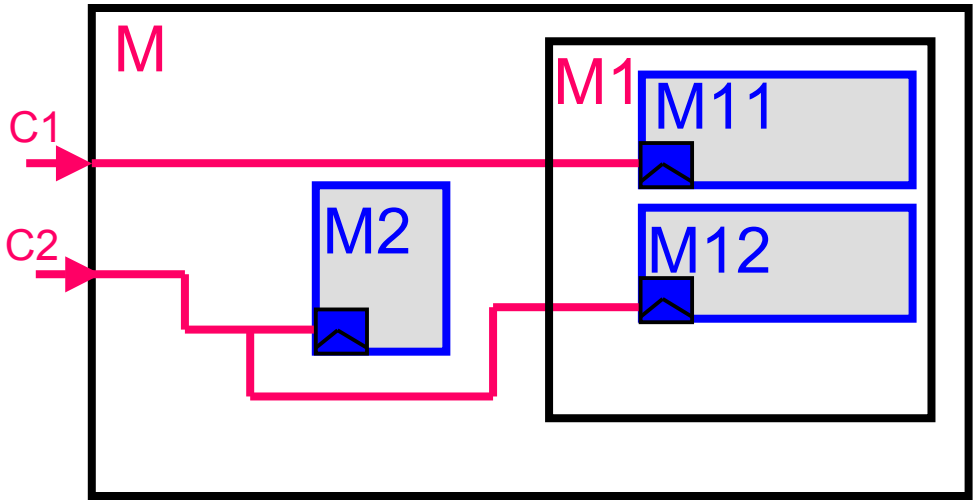
clock as a primitive special signal

```
multiclock M:
input C1,C2: clock;
...
 run M1[C1/CC1, C2/CC2]
||
 run M2[clock C2]
end module
```

```
multiclock M1:
input CC1,CC2:clock;
   ...


end module
```

```
module M2:
...
end module
```



hierarchical multiclock design

```
multiclock M:
input C1,C2: clock;
...
 run M1[C1/CC1,C2/CC2]
||
 run M2[clock C2]
end module
```

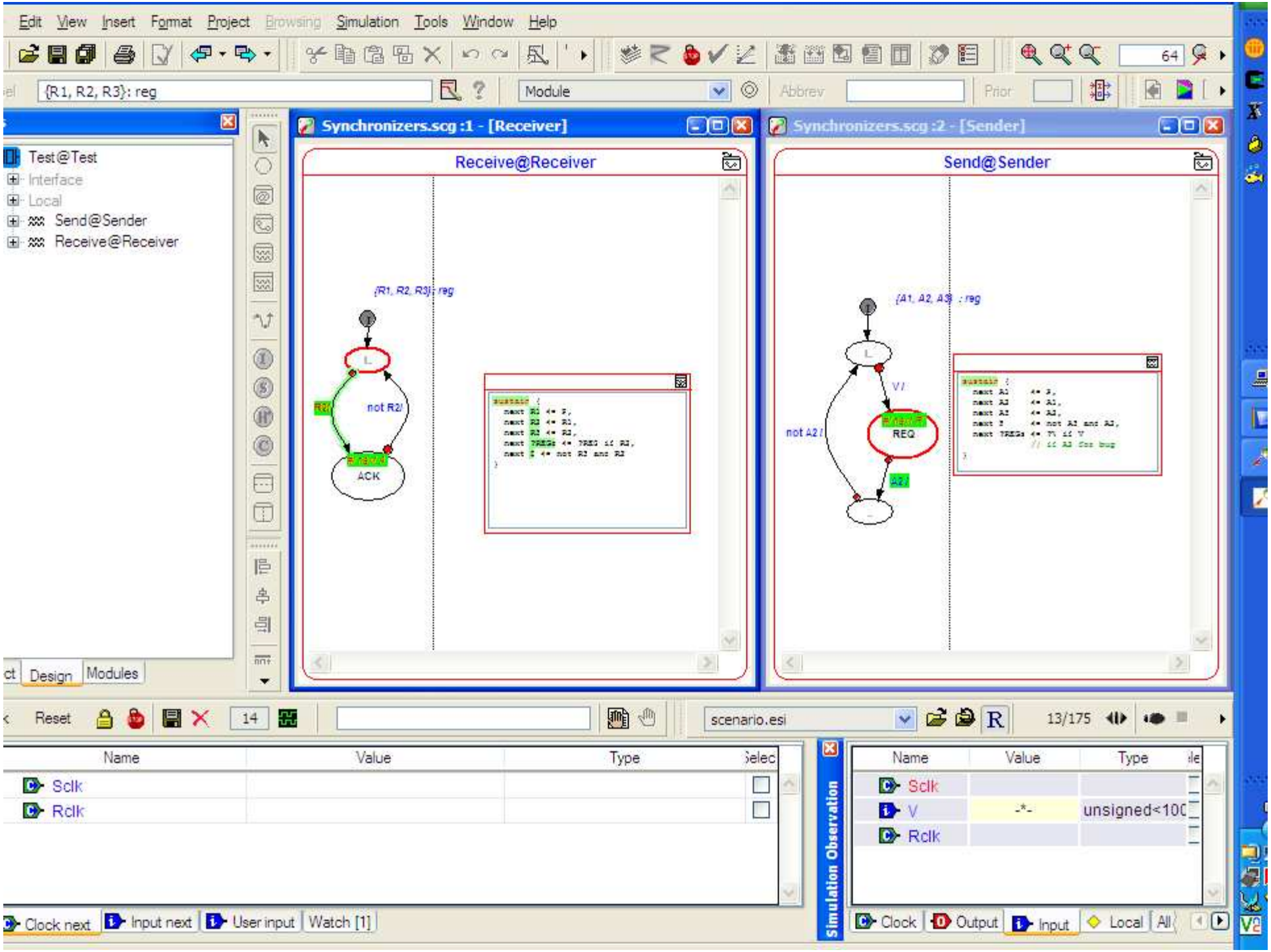
```
multiclock M1:
   input CC1,CC2:clock;
   ...
    run M11[clock CC1]
   ||
    run M12[clock CC2]
end module
```

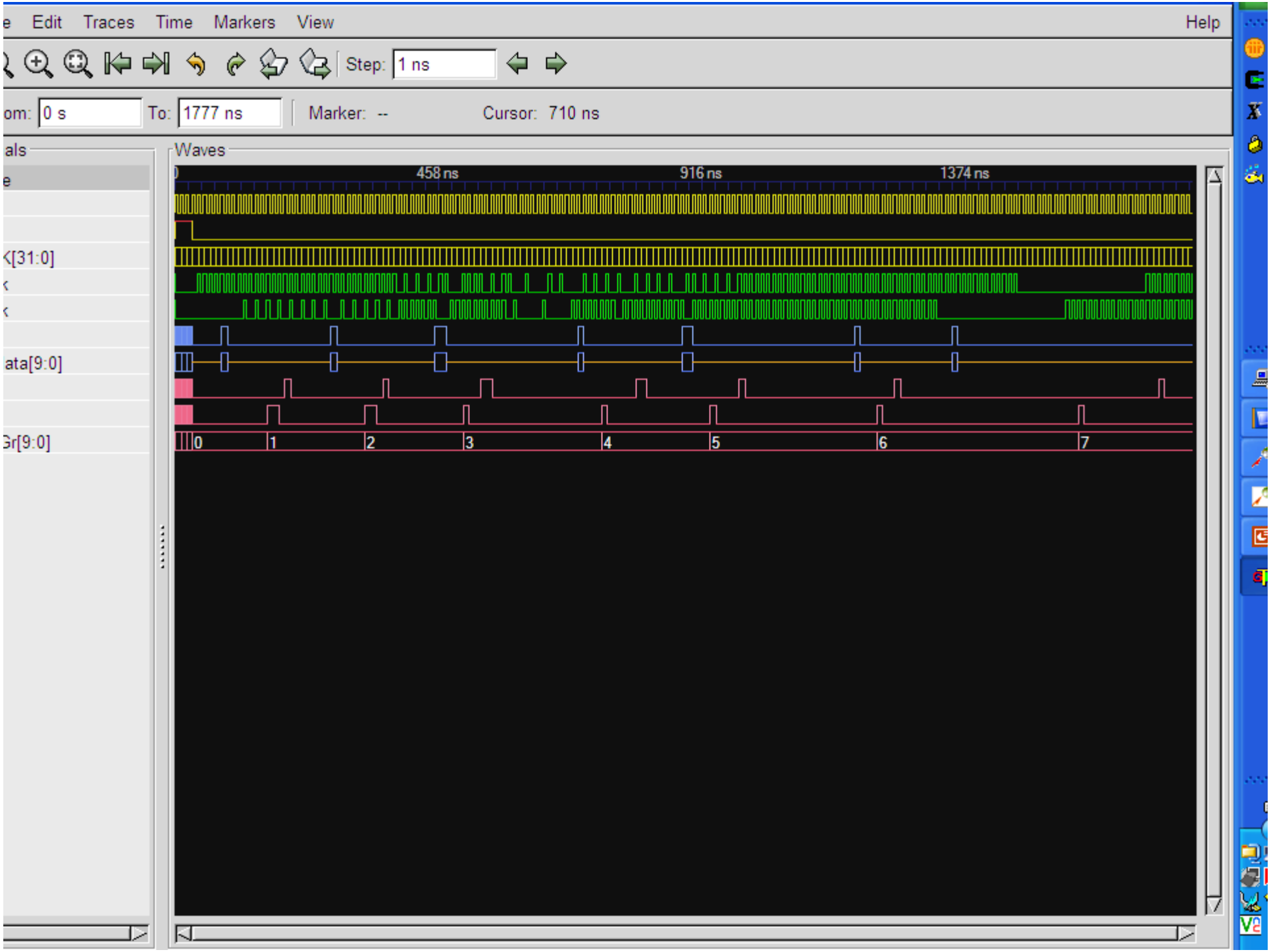
```
module M11:
...
end module
```

```
module M12:
...
end module
```

```
module M2:
...
end module
```

```
multiclock Multi :
input {C1, C2} : clock;
output C3 : clock;
input I, J;
output X, Y;
signal C4 : clock in
   sustain {
      C4 <= C1 if I,
      C3 <= mux(J, C2, C4)
   }
||
   run M1 [clock C1]
||
   run M2 [clock C4]
||
   run MultiSub
end multiclock
```


Edit View Insert Format Project Browsing Simulation Tools Window Help
{R1, R2, R3}: reg
Module
Abbrev
Prior
Test@Test
Interface
Local
Send@Sender
Receive@Receiver
Design
Modules
Synchronizers.scg :1 - [Receiver]
Receive@Receiver
{R1, R2, R3}: reg
L
R2/
not R2/
ACK
Synchronizers.scg :2 - [Sender]
Send@Sender
{A1, A2, A3} : reg
L
V/
not A2/
REQ
A2/
Reset
14
scenario.esi
13/175
Name
Value
Type
Selec
Sclk
Rclk
Clock next
Input next
User input
Watch [1]
Simulation Observation
Name
Value
Type
Sclk
V
-*-
unsigned<10
Rclk
Clock
Output
Input
Local
All


Edit
Traces
Time
Markers
View
Help
Step:
1 ns
om:
0 s
To:
1777 ns
Marker: --
Cursor: 710 ns
als
Waves
458 ns
916 ns
1374 ns
[31:0]
ata[9:0]
Gr[9:0]
0
1
2
3
4
5
6
7

Multiclock Semantics

Simple translation to Classic Esterel:

1. Add universal simulation clock
2. View declared clocks as conventional signals
3. Transform clocked run into immediate weak suspend

```
   run M [ clock C ]
=>
   weak suspend
      run M
   when immediate (not C)
```

Running Multiclock Units in Modules

```
module M :
   ...
   signal S1, S2 in
      every 2 tick do emit S1 end
   ||
      every 3 tick do emit S2 end
   ||
      mcrun Multi [ S1 / C1, S2 / C2, ...]
end module
```

- Makes it possible to simulate clock generators
- An to use Esterel as a multiclock testbench generation language

Clock Constraints

- Electricity : danger of metastability when sampling clock-domain crossing (CDC) signal
- Neglected by Esterel, which remains 0-delay
- Can be controlled by inter-clock relationship
  - harmonic (rational) relations : ex. 3/2
  - phase-shifted clocks, etc.

  $\Rightarrow$ count like musicians to ensure edge separation
- Asynchronous clocks => synchronizers

  Cf. Ran Ginosar, “Fourteen ways to fool your synchronizer”

Conclusion

- Weak suspension
  - makes clock-gating a full-fledged statement
  - can be implemented by enabling logic or actual clock-gating
  - behavior well-defined but can be intricate
- Multiclocking
  - fundamental for modern designs
  - easy language / semantics extension
  - multiclock synthesis or single-clock simulation
  - amenable to formal verification
  - but behavior can be very intricate!