8. Electronic Circuit Developments

Jean-Raymond Abrial

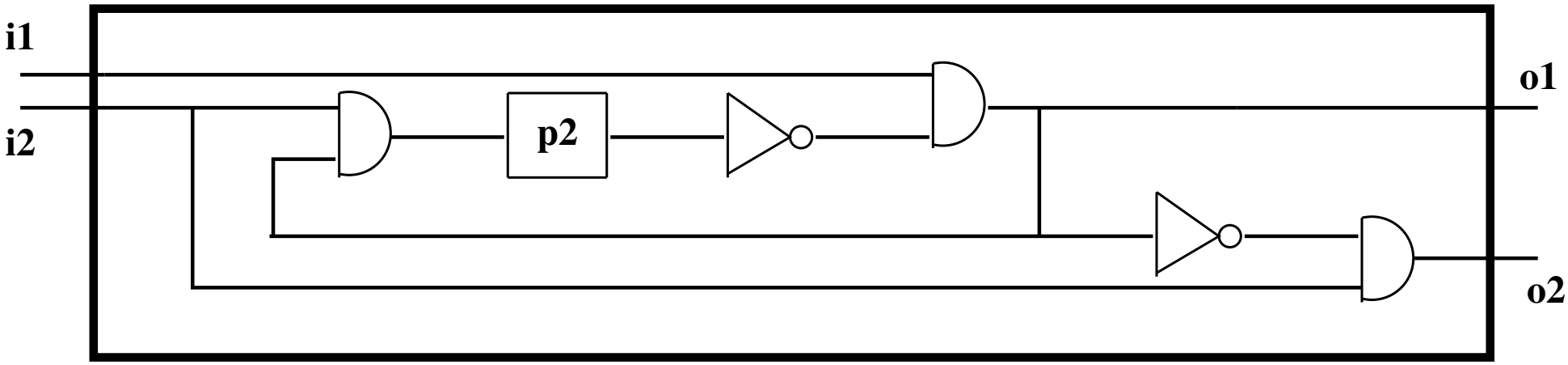
2009

Purpose of Lecture

- To present a methodology for developing electronic circuits

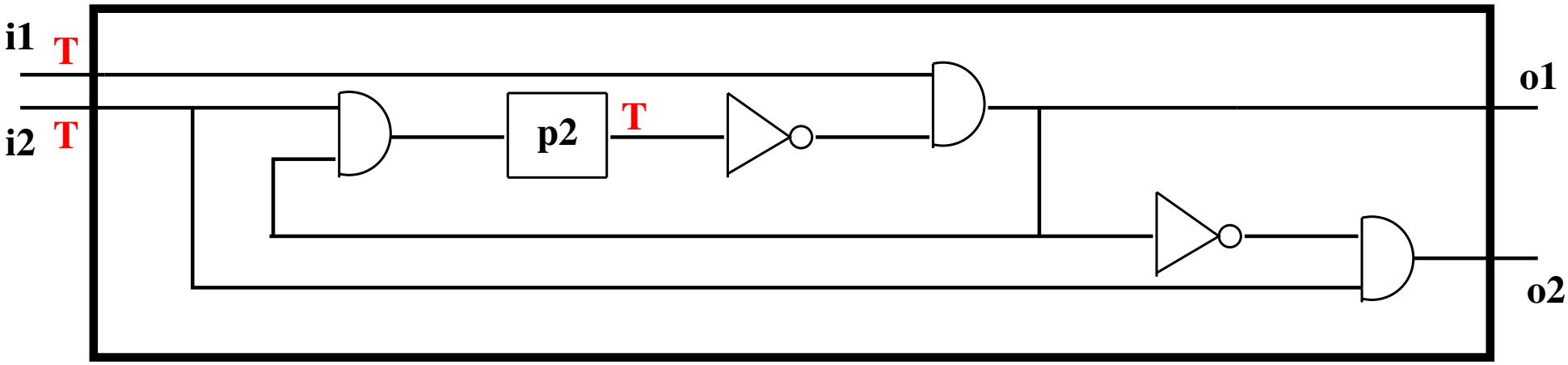
- To cover the development of three examples

A Typical Circuit



- It is made of the following components:
  - two input wires $i1$ and $i2$ carrying boolean values,
  - two output wires $o1$ and $o2$ carrying boolean values,
  - various gates (here three and-gates and two not-gates,
  - a register $p2$ containing a boolean value.

Example of Computation: Initial Situation

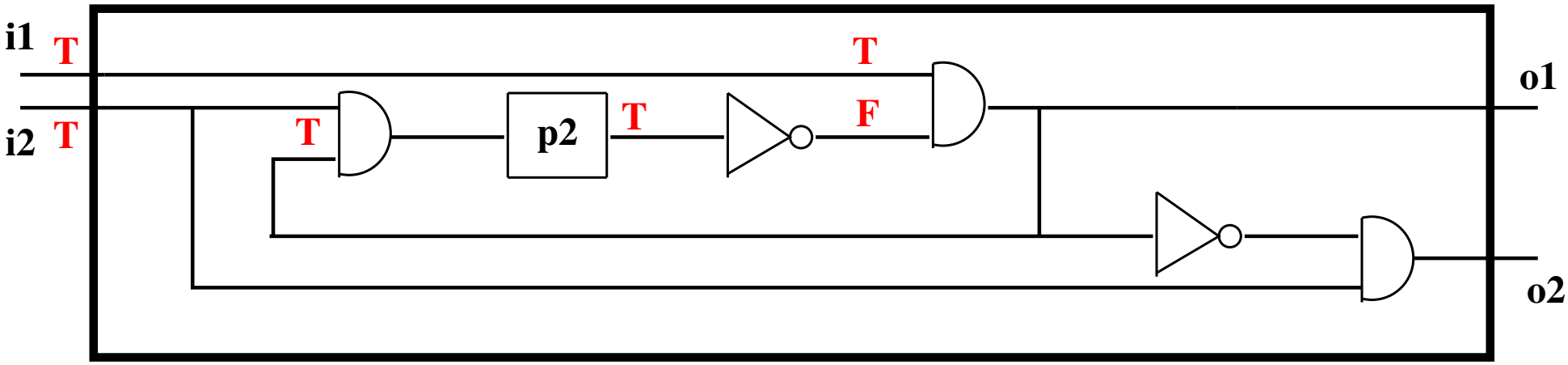


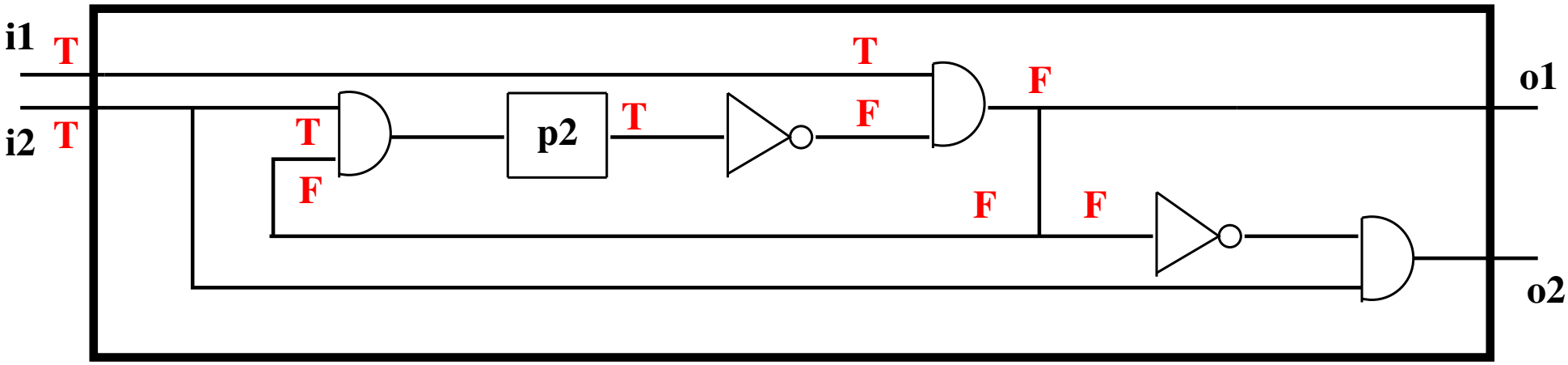
- Initially: $p2 = \mathrm{TRUE}$

$i1 = \mathrm{TRUE}$

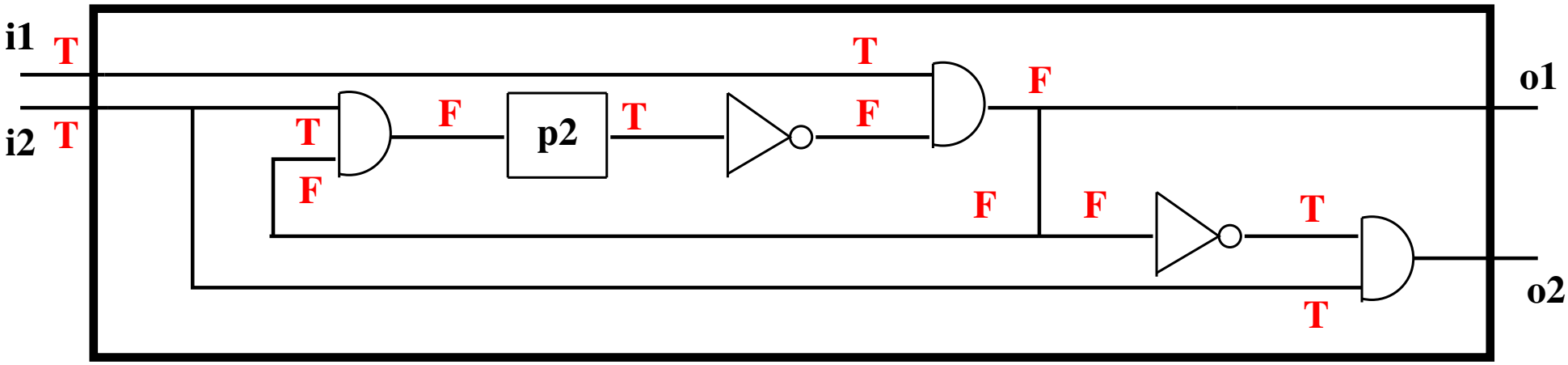
$i2 = \mathrm{TRUE}$

Computation

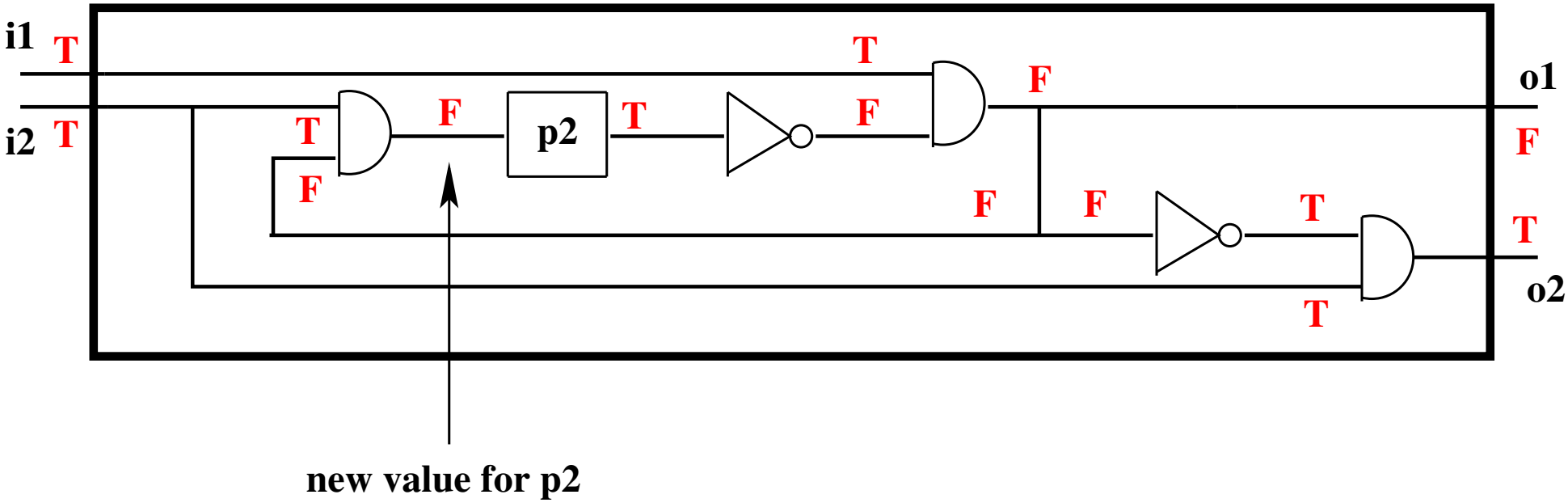



i1
T
i2
T
T
F
p2
T
T
F
T
F
F
F
o1
o2

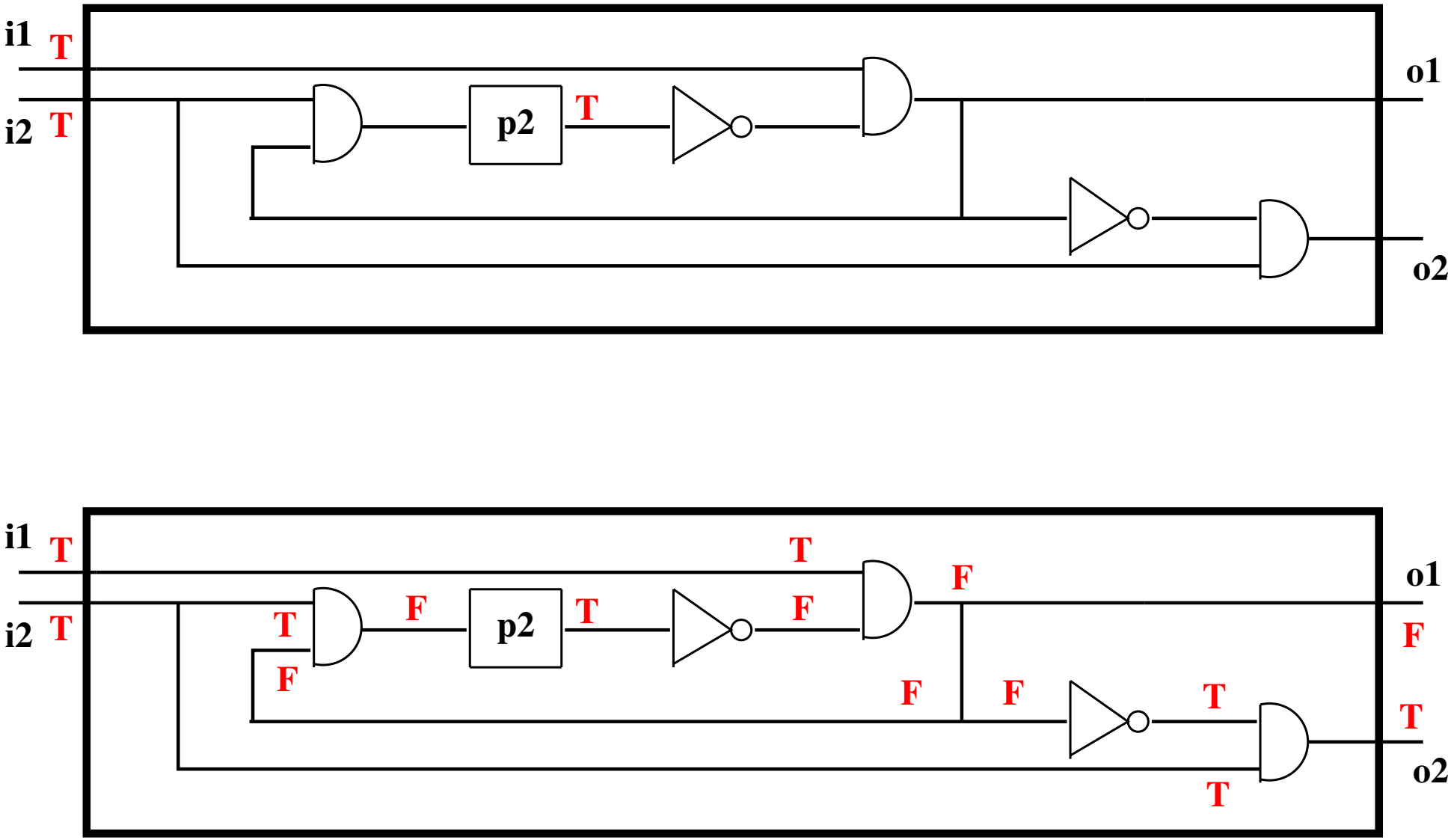
Computation



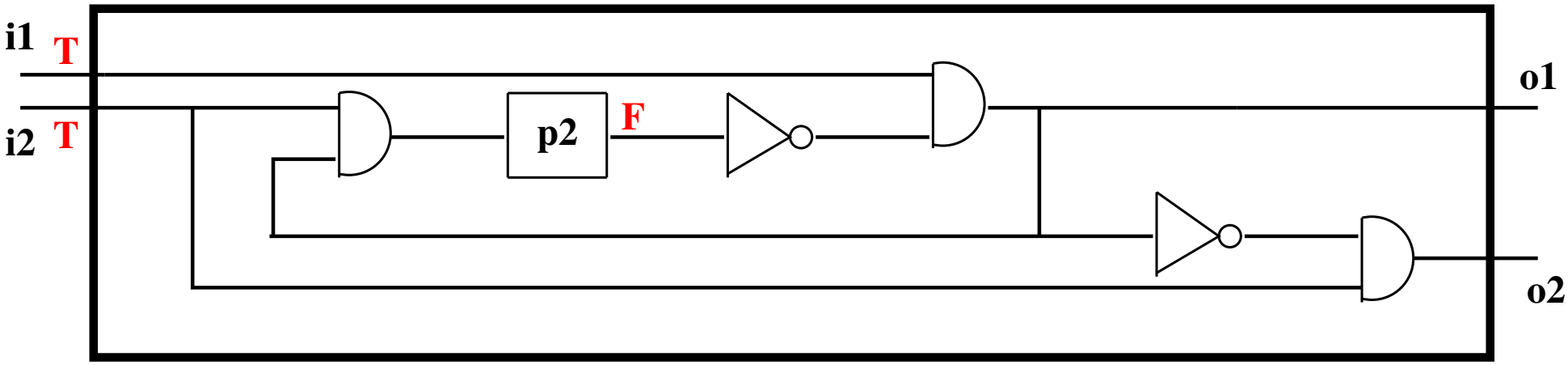
Computation



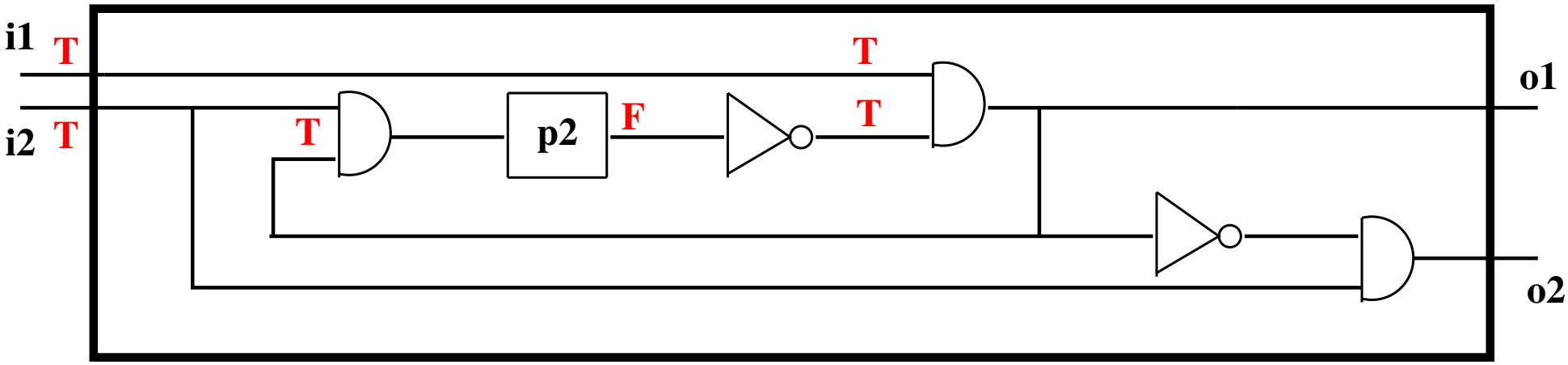
Computation Done in one Cycle



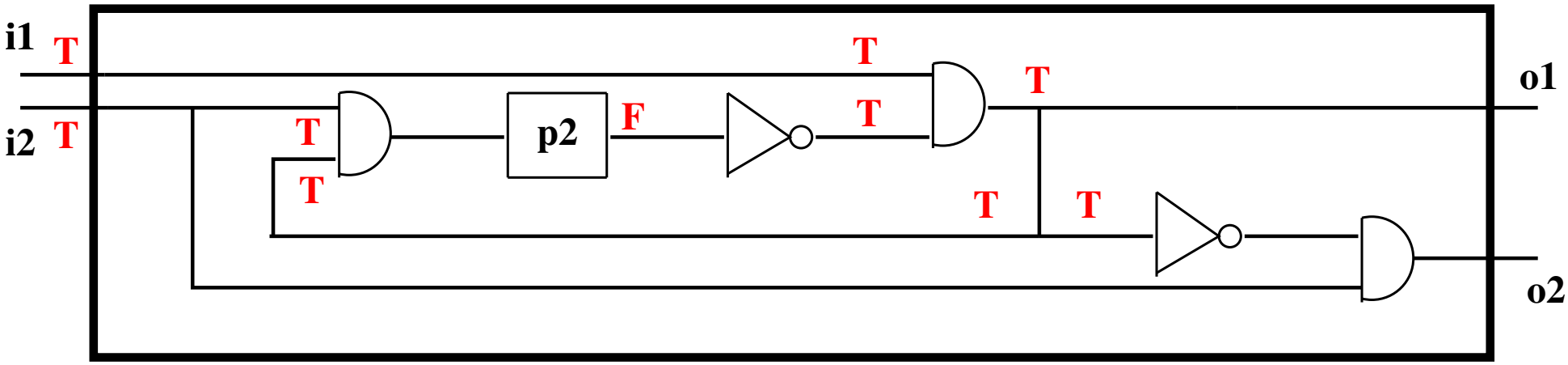
Another Computation



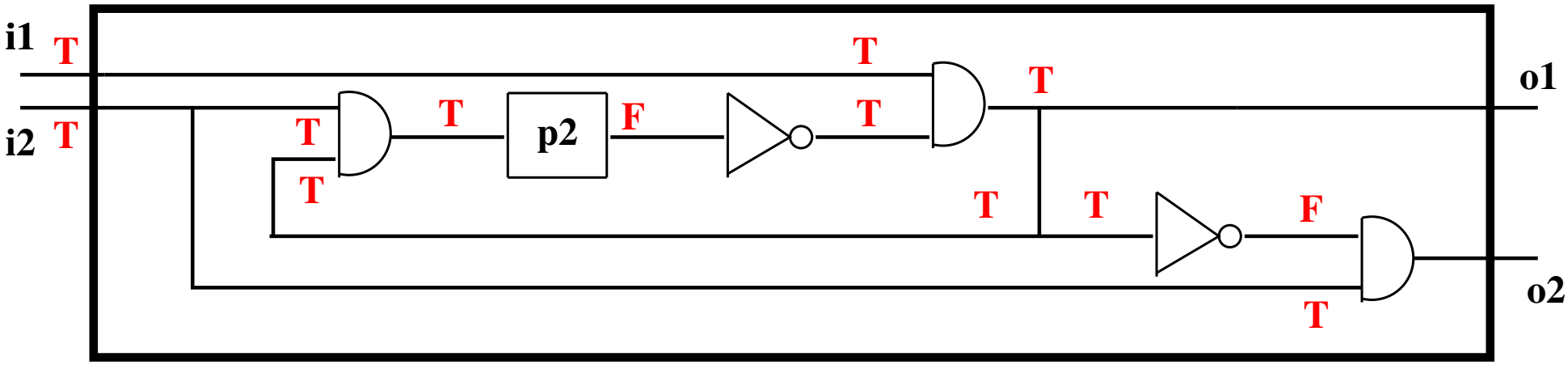
Computation



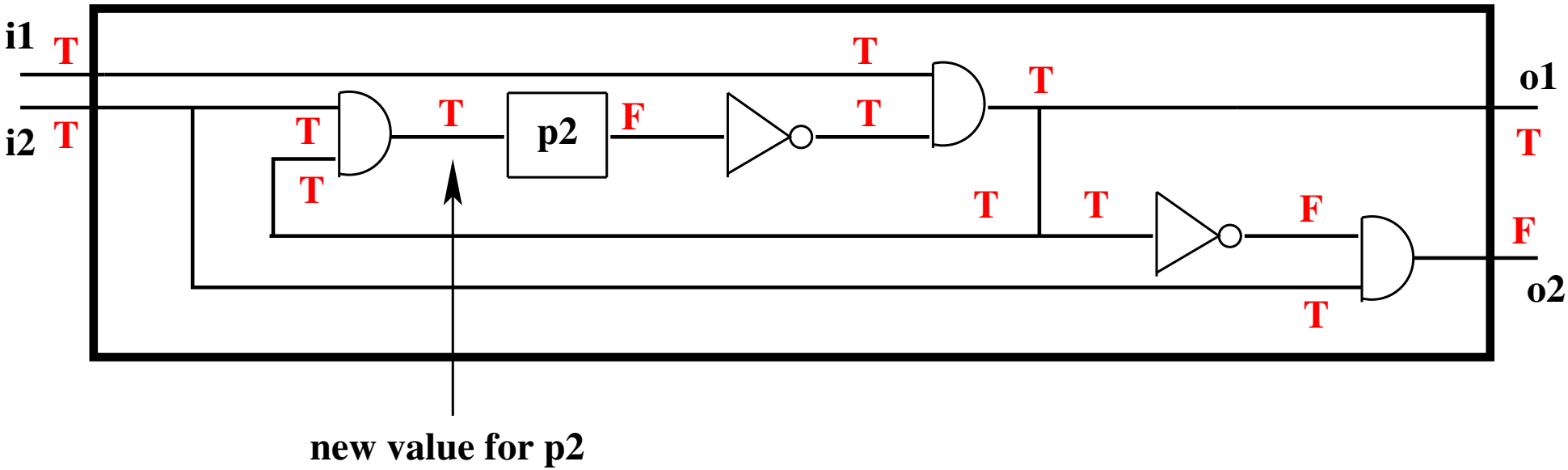
Computation



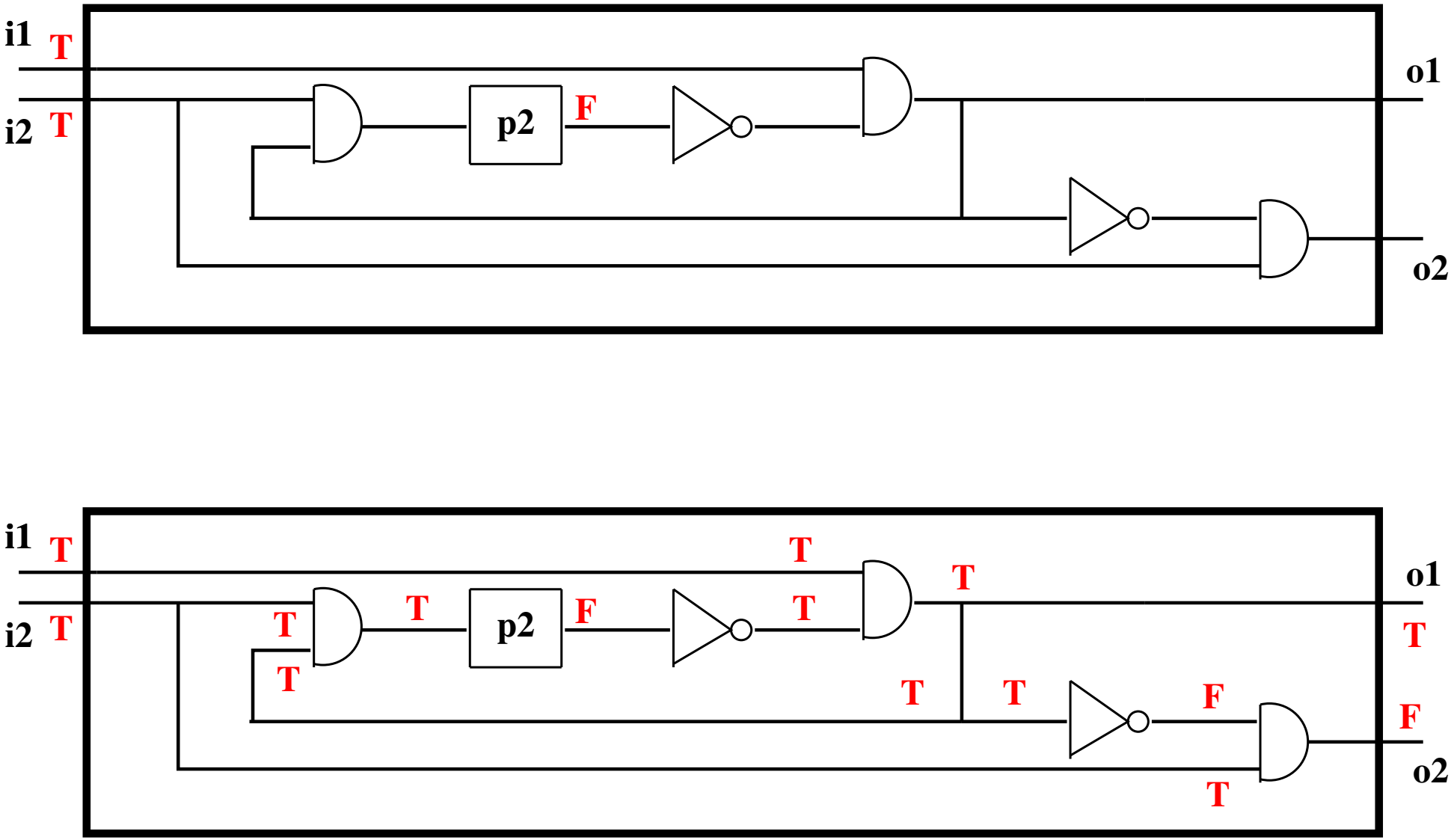
Computation



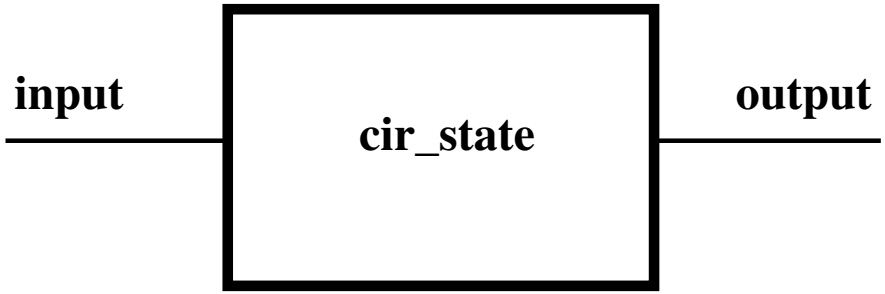
Computation



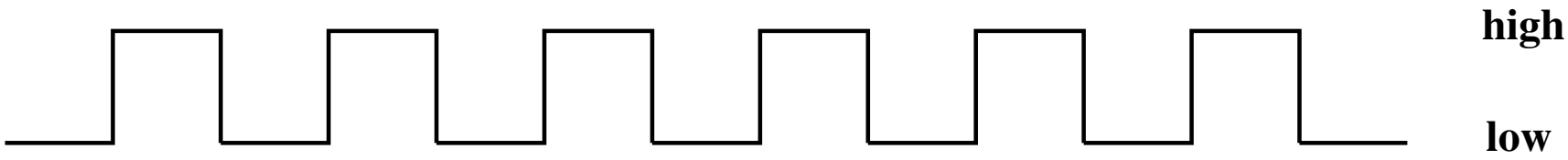
Computation Done in one Cycle



- A Circuit



- A Clock



Synchronous Circuits (cont'd)

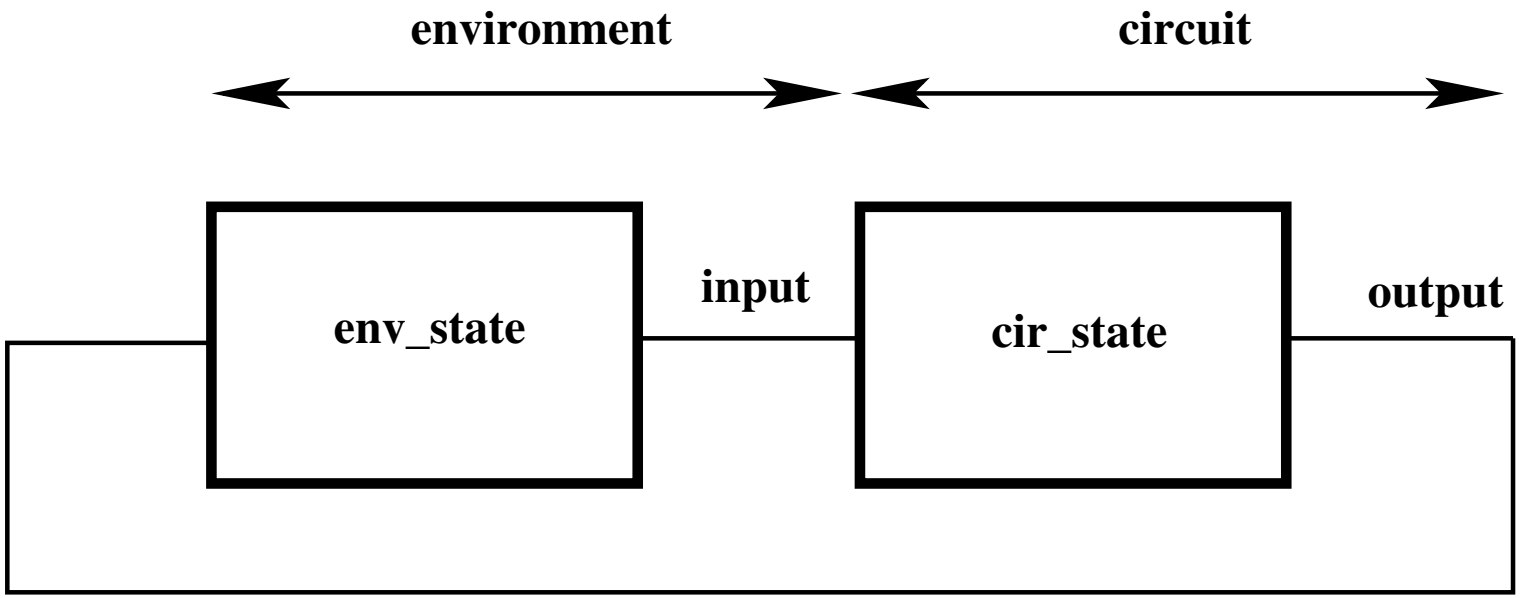
- When the clock is low:
  - cir_state and the $output$ wires are supposed to stay idle
  - only the $input$ wires may be modified,

- When the clock is $high$:
  - the $input$ wires are supposed to stay idle
  - cir_state and $output$ wires may be modified.

- cir_state and $output$ form the circuit

- $input$ and also outside state env_state form the environment

Coupling the Circuit and its Environment



The notion of clock can be made more abstract.

Modes of Observation

- Two alternative ways ofobserving the closed system.

- Here are these two modes of observation:

  -env, corresponds to observing the environment,

  -cir, consists in observing the circuit.

Such modes alternate for ever.

Dynamic View of the Coupling

- Let c be the circuit variables and e be the environment variables

- Circuit Events

cir_event_i
when
$mode = cir$
$GC\_i(input, c)$
then
$mode := env$
$c, output :| PC\_i(input, c, c', output')$
end

- Environment Events

env_event_j
when
$mode = env$
$GE\_j(output, e)$
then
$mode := cir$
$e, input :| PE\_j(output, e, e', input')$
end

Dynamic View of the Coupling (cont'd)

- The final circuit:
  - uses the $input$
  - modifies c and $output$
  - never accesses e

- The final environment:
  - uses the $output$
  - modifies e and $input$
  - never accesses c

- Warning: this is not necessary the case before last refinement

Static View of the Coupling

$$mode = env \;\Rightarrow\; C(e, input, c, output)$$

$$mode = cir \;\Rightarrow\; D(e, input, c, output)$$

- C states what the circuit establishes for the environment provided the circuit behaves in a situation where D holds.

- D states what the environment establishes for the circuit provided the environment behaves in a situation where C holds.

Consistency

$$C(e, input, c, output)$$
$$GE\_j(output, e)$$
$$PE\_j(output, e, e', input')$$
$$\Rightarrow$$
$$D(e', input', c, output))$$

$$D(e, input, c, output)$$
$$GC\_i(input, c)$$
$$PC\_i(input, c, c', output')$$
$$\Rightarrow$$
$$C(e, input, c', output')$$

- When C holds, then D must hold after any modifications e' and $input'$ made by the environment.

- When D holds, then C must hold after any modifications c' and $output'$ made by the circuit.

Final Conditions

- the circuit variables must all be boolean,

- the inputs must be boolean,

- the outputs must be boolean,

- the circuit must be deadlock free,

Final Conditions (cont'd)

- the circuit must be internally deterministic (variable assignments),

- the circuit must be externally deterministic (guards),

- the environment does not access the circuit variables,

- the circuit does not access the environment variables.

A Typical Final Circuit Event

cir_event_i

when

$mode = cir$

$GC\_i(input, c)$

then

$mode := env$

$c := C\_i(input, c)$

$output := O\_i(input, c)$

end

Final Conditions

- Deadlock Freeness: Disjunction of guards

$$mode = cir \;\Rightarrow\; GC\_1(input, c) \;\vee\; \ldots \;\vee\; GC\_n(input, c)$$

- External Determinacy: Mutual exclusion of guards

$$mode = cir \;\Rightarrow\; \neg\,(\,GC\_i(input, c) \;\wedge\; GC\_j(input, c)\,)$$

Merging of Circuit Events (implicit refinement)

cir_event_i

when

$mode = cir$

$GC\_i(input, c)$

then

$mode := env$

$$c := \text{bool}\left(\begin{array}{l} \ldots \\ GC\_i(input, c) \ \wedge \ C\_i(input, c) = \text{TRUE} \ \vee \\ \ldots \end{array} \right)$$

$$output := \text{bool}\left(\begin{array}{l} \ldots \\ GC\_i(input, c) \ \wedge \ O\_i(input, c) = \text{TRUE} \ \vee \\ \ldots \end{array} \right)$$

end

The bool operator: transforming a predicate into a boolean expression

$$E = \text{bool}(P) \quad \Leftrightarrow \quad \left(\begin{array}{lcl} P & \Rightarrow & E = \text{TRUE} \\ \neg\, P & \Rightarrow & E = \text{FALSE} \end{array} \right)$$

Proving the Implicit Refinement

$$GC\_i(input, c)$$
$$\Rightarrow$$
$$C\_i(input, c) \;=\; \mathbf{bool}\left(\begin{array}{l}\ldots \;\vee \\ GC\_i(input, c) \;\wedge\; C\_i(input, c) = \mathbf{TRUE} \;\vee \\ \ldots\end{array}\right)$$

- According to the definition of $\mathbf{bool}$, this reduces to the following two statements:

$$\left.\begin{array}{l} GC\_i(input, c) \\ \left(\begin{array}{l}\ldots \;\vee \\ GC\_i(input, c) \;\wedge\; C\_i(input, c) = \mathbf{TRUE} \;\vee \\ \ldots\end{array}\right) \\ \Rightarrow \\ C\_i(input, c) = \mathbf{TRUE}\end{array}\right.$$

$$\left.\begin{array}{l} GC\_i(input, c) \\ \neg\left(\begin{array}{l}\ldots \;\vee \\ GC\_i(input, c) \;\wedge\; C\_i(input, c) = \mathbf{TRUE} \;\vee \\ \ldots\end{array}\right) \\ \Rightarrow \\ C\_i(input, c) = \mathbf{FALSE}\end{array}\right.$$

Proving the First Statement

$$
\begin{array}{l}
GC\_i(input, c) \\
\left(\begin{array}{l}
\ldots \ \vee \\
GC\_i(input, c) \ \wedge \ C\_i(input, c) = \mathrm{TRUE} \ \vee \\
\ldots
\end{array} \right) \\
\Rightarrow \\
C\_i(input, c) = \mathrm{TRUE}
\end{array}
$$

- Since the guard are mutually exclusive, it can be reduced to:

$$
\begin{array}{l}
GC\_i(input, c) \\
C\_i(input, c) = \mathrm{TRUE} \\
\Rightarrow \\
C\_i(input, c) = \mathrm{TRUE}
\end{array}
$$

Proving the Second Statement

$$
\begin{array}{l}
GC\_i(input, c) \\
\neg \left(\begin{array}{l} \ldots \ \vee \\ GC\_i(input, c) \ \wedge \ C\_i(input, c) = \mathbf{TRUE} \ \vee \\ \ldots \end{array} \right) \\
\Rightarrow \\
C\_i(input, c) = \mathbf{FALSE}
\end{array}
$$

- Equivalently (and trivially discharged)

$$
\begin{array}{l}
GC\_i(input, c) \\
\ldots \\
(\neg\, GC\_i(input, c) \ \vee \ C\_i(input, c) = \mathbf{FALSE}) \\
\ldots \\
\Rightarrow \\
C\_i(input, c) = \mathbf{FALSE}
\end{array}
$$

Final Circuit

- The circuit events are deadlock free (disjunction of guards holds)

- They have identical actions,

- They can all be merged into a single event as follows:

circuit_event
when
$mode = cir$
then
$mode := env$

$$c := \text{bool}\left(\begin{array}{l} \ldots \ \vee \\ GC\_i(input, c) \ \wedge \ C\_i(input, c) = \text{TRUE} \ \vee \\ \ldots \end{array}\right)$$

$$output := \text{bool}\left(\begin{array}{l} \ldots \ \vee \\ GC\_i(input, c) \ \wedge \ O\_i(input, c) = \text{TRUE} \ \vee \\ \ldots \end{array}\right)$$

end

Further Simplifications

- When $C_i(input, c)$ is syntactically equal to $\mathbf{TRUE}$ then $C_i(input, c) = \mathbf{TRUE}$ can be removed,

circuit_event
when
$mode = cir$
then
$mode := env$

$$c := \text{bool}\left(\begin{array}{l}\ldots \ \vee \\ GC\_i(input, c) \wedge \mathbf{TRUE} = \mathbf{TRUE} \ \vee \\ \ldots\end{array}\right)$$

$$output := \text{bool}\left(\begin{array}{l}\ldots \ \vee \\ GC\_i(input, c) \ \wedge \ O\_i(input, c) = \mathbf{TRUE} \ \vee \\ \ldots\end{array}\right)$$

end

- Same for $O\_i(input, c)$

Further Simplifications (cont'd)

- When $C_i(input, c)$ is syntactically equal to $\mathbf{FALSE}$ then $GC\_i(input, c) \wedge C_i(input, c) = \mathbf{TRUE}$ can be removed.

circuit_event
  when
    $mode = cir$
  then
    $mode := env$
    $c := \text{bool}\left(\begin{array}{l}\ldots \vee \\ GC\_i(input, c) \wedge \mathbf{FALSE} = \mathbf{TRUE} \vee \\ \ldots\end{array}\right)$
    $output := \text{bool}\left(\begin{array}{l}\ldots \vee \\ GC\_i(input, c) \wedge O\_i(input, c) = \mathbf{TRUE} \vee \\ \ldots\end{array}\right)$
  end

- Same for $O\_i(input, c)$

First Example: The Arbiter

The circuit has two boolean inputs $i1$ and $i2$ and two boolean outputs $o1$ and $o2$	FUN-1

Requirements

A $\mathbf{TRUE}$ input means a user, associated with that input, is asking for a certain resource	FUN-2
The circuit reacts positively to a request by setting the corresponding output to $\mathbf{TRUE}$	FUN-3
The circuit can react positively to one request only at a time (mutual exclusion)	FUN-4

More Requirements

Each user frees the resource immediately	FUN-5

A requesting user cannot be postponed indefinitely	FUN-6

A user asking for a resource continues to ask for it as long as it is not served	FUN-7

The resource cannot be granted without a user asking for it	FUN-8

Initial Abstract Arbiter

- r_i denotes the number of requests made by user i

- a_i denotes the number of acknowledgements made by the circuit for user i

Initial Abstract Model

- ri number of requests made so far by user i

- ai number of acknowledgements made so far for user i

variables: $r1, r2, a1, a2$

inv0_1: $r1 \in \mathbb{N}$

inv0_2: $r2 \in \mathbb{N}$

inv0_3: $a1 \in \mathbb{N}$

inv0_4: $a2 \in \mathbb{N}$

More Invariants

inv0_5: $a1 \leq r1$

inv0_6: $r1 \leq a1 + 1$

inv0_7: $a2 \leq r2$

inv0_8: $r2 \leq a2 + 1$

- Number of requests greater or equal to the number of acks.
  (no spurious acks)

- The number of requests never exceeds the number of ack.
  by more than 1 (users never quit without being served)

More State Variables

- A requesting user never starves

variables: $p1, p2$

- $pi = \text{TRUE}$ means user i waits to be served

inv0_9: $p1 \in \text{BOOL}$

inv0_10: $p2 \in \text{BOOL}$

inv0_11: $p1 = \text{FALSE} \lor p2 = \text{FALSE}$

inv0_12: $mode = env \Rightarrow (r1 = a1 \Leftrightarrow p1 = \text{FALSE})$

inv0_13: $mode = env \Rightarrow (r2 = a2 \Leftrightarrow p2 = \text{FALSE})$

Environment Events (1)

- Asking for user 1 or for user 2

env1
when
$mode = env$
$r1 = a1$
then
$mode := cir$
$r1 := r1 + 1$
end

env2
when
$mode = env$
$r2 = a2$
then
$mode := cir$
$r2 := r2 + 1$
end

Environment Events (2)

- Asking for both users

env3
when
$mode = env$
$r1 = a1$
$r2 = a2$
then
$mode := cir$
$r1 := r1 + 1$
$r2 := r2 + 1$
end

env0
when
$mode = env$
then
$mode := cir$
end

- Note the last "do-nothing" event

Circuit Events (2)

cir1

when

$mode = cir$

$r1 \neq a1$

$p2 = \mathrm{FALSE}$

then

$mode := env$

$a1 := a1 + 1$

$p1 := \mathrm{FALSE}$

$p2 := \mathrm{bool}(r2 \neq a2)$

end

cir2

when

$mode = cir$

$r2 \neq a2$

$p1 = \mathrm{FALSE}$

then

$mode := env$

$a2 := a2 + 1$

$p2 := \mathrm{FALSE}$

$p1 := \mathrm{bool}(r1 \neq a1)$

end

- Notice the external non-determinacy

Circuit Events (2)

- A circuit "do-nothing" event

cir0

when

$mode = cir$

$r1 = a1$

$r2 = a2$

then

$mode := env$

$p1 := \mathrm{FALSE}$

$p2 := \mathrm{FALSE}$

end

$$mode = cir \Rightarrow \begin{pmatrix} r1 \neq a1 \ \wedge \ p2 = \mathbf{FALSE} \ \vee \\ r2 \neq a2 \ \wedge \ p1 = \mathbf{FALSE} \ \vee \\ r1 = a1 \ \wedge \ r2 = a2 \end{pmatrix}$$

We have to add the following invariants:

inv0_14: $mode = cir \Rightarrow (r1 = a1 \Rightarrow p1 = \mathbf{FALSE})$

inv0_15: $mode = cir \Rightarrow (r2 = a2 \Rightarrow p2 = \mathbf{FALSE})$

Proofs of Initial Model

- 53 proofs

- All automatic

Problems of Initial Model

- Inputs $r1$ and $r2$ are not boolean

- Outputs $a1$ and $a2$ are not boolean

- Circuit reads its outputs $a1$ and $a2$

- Circuit is externally non-deterministic

First Refinement

- Delivering boolean outputs $o1$ and $o2$ (the offsets to $a1$ and $a2$)
- Pushing $a1$ and $a2$ outside of the circuit: $b1$ and $b2$
- $b1$ and $b2$ are slightly shifted

variables: $b1, o1, b2, o2$

inv1_1: $b1 \in \mathbb{N}$

inv1_2: $o1 \in \mathrm{BOOL}$

inv1_3: $b2 \in \mathbb{N}$

inv1_4: $o2 \in \mathrm{BOOL}$

- Transforming a boolean into a number (0 or 1)

constants: b_2_01

prp1_1: $b\_2\_01 \in \mathrm{BOOL} \rightarrow \{0, 1\}$

prp1_2: $b\_2\_01(\mathrm{TRUE}) = 1$

prp1_3: $b\_2\_01(\mathrm{FALSE}) = 0$

inv1_5: $mode = cir \Rightarrow a1 = b1$

inv1_6: $mode = cir \Rightarrow a2 = b2$

inv1_7: $mode = env \Rightarrow a1 = b1 + b\_2\_01(o1)$

inv1_8: $mode = env \Rightarrow a2 = b2 + b\_2\_01(o2)$

First Refinement: Environment Events

env1

when

$mode = env$

$r1 = b1 + b\_2\_01(o1)$

then

$mode := cir$

$r1 := r1 + 1$

$b1 := b1 + b\_2\_01(o1)$

$b2 := b2 + b\_2\_01(o2)$

end

env2

when

$mode = env$

$r2 = b2 + b\_2\_01(o2)$

then

$mode := cir$

$r2 := r2 + 1$

$b1 := b1 + b\_2\_01(o1)$

$b2 := b2 + b\_2\_01(o2)$

end

First Refinement: Environment Events (cont'd)

env3
when
$mode = env$
$r1 = b1 + b\_2\_01(o1)$
$r2 = b2 + b\_2\_01(o2)$
then
$mode := cir$
$r1 := r1 + 1$
$r2 := r2 + 1$
$b1 := b1 + b\_2\_01(o1)$
$b2 := b2 + b\_2\_01(o2)$
end

(abstract-)env3
when
$mode = env$
$r1 = a1$
$r2 = a2$
then
$mode := cir$
$r1 := r1 + 1$
$r2 := r2 + 1$
end

inv1_7: $mode = env \Rightarrow a1 = b1 + b\_2\_01(o1)$

inv1_8: $mode = env \Rightarrow a2 = b2 + b\_2\_01(o2)$

First Refinement: Circuit Events

cir1
when
$mode = cir$
$r1 \neq b1$
$p2 = \text{FALSE}$
then
$mode := env$
$o1 := \text{TRUE}$
$o2 := \text{FALSE}$
$p1 := \text{FALSE}$
$p2 := \text{bool}(r2 \neq b2)$
end

(abstract-)cir1
when
$mode = cir$
$r1 \neq a1$
$p2 = \text{FALSE}$
then
$mode := env$
$a1 := a1 + 1$
$p1 := \text{FALSE}$
$p2 := \text{bool}(r2 \neq a2)$
end

inv1_5: $mode = cir \Rightarrow a1 = b1$

inv1_6: $mode = cir \Rightarrow a2 = b2$

cir2

when

$mode = cir$

$r2 \neq b2$

$p1 = \mathrm{FALSE}$

then

$mode := env$

$o1 := \mathrm{FALSE}$

$o2 := \mathrm{TRUE}$

$p1 := \mathrm{bool}(r1 \neq b1)$

$p2 := \mathrm{FALSE}$

end

cir0

when

$mode = cir$

$r1 = b1$

$r2 = b2$

then

$mode := env$

$o1 := \mathrm{FALSE}$

$o2 := \mathrm{FALSE}$

$p1 := \mathrm{FALSE}$

$p2 := \mathrm{FALSE}$

end

First Refinement: Proofs

- 57 Proofs

- all automatic

First Refinement: Problems

- Inputs are not boolean: $r1$ and $r2$

- Circuit is accessing environment variables: $b1$ and $b2$

- Circuit still externally non-deterministic

Second Refinement: the State

- Introducing boolean inputs $i1$ and $i2$.

variables: $i1, i2$

inv2_1: $i1 \in \mathrm{BOOL}$

inv2_2: $i2 \in \mathrm{BOOL}$

- Gluing Invariant:

inv2_2: $mode = cir \Rightarrow (i1 = \mathrm{FALSE} \Leftrightarrow r1 = b1)$

inv2_3: $mode = cir \Rightarrow (i2 = \mathrm{FALSE} \Leftrightarrow r2 = b2)$

Second Refinement: the Circuit Events

cir1
when
$mode = cir$
$i1 = \text{TRUE}$
$p2 = \text{FALSE}$
then
$mode := env$
$o1 := \text{TRUE}$
$o2 := \text{FALSE}$
$p1 := \text{FALSE}$
$p2 := i2$
end

(abstract-)cir1
when
$mode = cir$
$r1 \neq b1$
$p2 = \text{FALSE}$
then
$mode := env$
$o1 := \text{TRUE}$
$o2 := \text{FALSE}$
$p1 := \text{FALSE}$
$p2 := \text{bool}(r2 \neq b2)$
end

inv2_2: $mode = cir \Rightarrow (i1 = \text{FALSE} \Leftrightarrow r1 = b1)$

inv2_3: $mode = cir \Rightarrow (i2 = \text{FALSE} \Leftrightarrow r2 = b2)$

Second Refinement: the Circuit Events

cir2

when

$mode = cir$
$i2 = \text{TRUE}$
$p1 = \text{FALSE}$

then

$mode := env$
$o1 := \text{FALSE}$
$o2 := \text{TRUE}$
$p1 := i1$
$p2 := \text{FALSE}$

end

cir0

when

$mode = cir$
$i1 = \text{FALSE}$
$i2 = \text{FALSE}$

then

$mode := env$
$o1 := \text{FALSE}$
$o2 := \text{FALSE}$
$p1 := \text{FALSE}$
$p2 := \text{FALSE}$

end

Second Refinement: Proofs

- 26 Proofs

- All automatic

Second Refinement: Pending Problem

- Circuit still non-deterministic

Third Refinement: the Circuit Events

- Removing variables $p1$

cir1
when
$mode = cir$
$i1 = \text{TRUE}$
$p2 = \text{FALSE}$
then
$mode := env$
$o1 := \text{TRUE}$
$o2 := \text{FALSE}$
$p2 := i2$
end

cir2
when
$mode = cir$
$i2 = \text{TRUE}$
$\neg (i1 = \text{TRUE} \wedge p2 = \text{FALSE})$
then
$mode := env$
$o1 := \text{FALSE}$
$o2 := \text{TRUE}$
$p2 := \text{FALSE}$
end

Third Refinement: the Circuit Events (cont'd)

cir0

when

$mode = cir$

$i1 = \mathrm{FALSE}$

$i2 = \mathrm{FALSE}$

then

$mode := env$

$o1 := \mathrm{FALSE}$

$o2 := \mathrm{FALSE}$

$p2 := \mathrm{FALSE}$

end

Third Refinement: Proving Deadlock Freeness

thm3_1:

$$mode = cir \Rightarrow \left(\begin{array}{l} i1 = \mathrm{TRUE} \ \wedge \ p2 = \mathrm{FALSE}) \ \vee \\ i2 = \mathrm{TRUE} \ \wedge \ \neg (i1 = \mathrm{TRUE} \ \wedge \ p2 = \mathrm{FALSE}) \ \vee \\ i1 = \mathrm{FALSE} \ \wedge \ i2 = \mathrm{FALSE} \end{array} \right)$$

- Note that the circuit is now clearly deterministic

Third Refinement: Proofs

- 5 proofs

- All automatic

- Total: 141 proofs

Building the Final Circuit

cir1
when
$mode = cir$
$i1 = \text{TRUE}$
$p2 = \text{FALSE}$
then
$mode := env$
$o1 := \text{TRUE}$
$o2 := \text{FALSE}$
$p2 := i2$
end

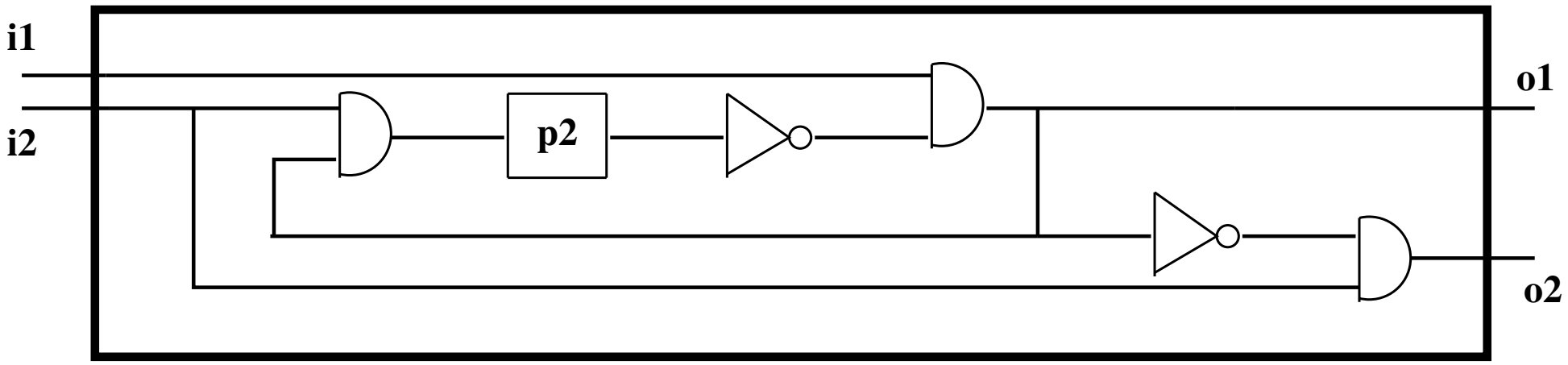
cir2
when
$mode = cir$
$i2 = \text{TRUE}$
$\neg \left(\begin{array}{l} i1 = \text{TRUE} \\ p2 = \text{FALSE} \end{array} \right)$
then
$mode := env$
$o1 := \text{FALSE}$
$o2 := \text{TRUE}$
$p2 := \text{FALSE}$
end

cir0
when
$mode = cir$
$i1 = \text{FALSE}$
$i2 = \text{FALSE}$
then
$mode := env$
$o1 := \text{FALSE}$
$o2 := \text{FALSE}$
$p2 := \text{FALSE}$
end

arbiter
when
$mode = cir$
then
$mode := env$
$o1 := \text{bool}\,(\, i1 = \text{TRUE} \ \wedge \ p2 = \text{FALSE}\,)$
$o2 := \text{bool}\,(\, i2 = \text{TRUE} \ \wedge \ \neg\,(i1 = \text{TRUE} \ \wedge \ p2 = \text{FALSE})\,)$
$p2 := \text{bool}\,(\, i1 = \text{TRUE} \ \wedge \ p2 = \text{FALSE} \ \wedge \ i2 = \text{TRUE}\,)$
end

Building the Final Circuit (cont'd)

arbiter
  when
    $mode = cir$
  then
    $mode := env$
    $o1 := \mathrm{bool}\,(\,i1 = \mathrm{TRUE} \;\wedge\; p2 = \mathrm{FALSE}\,)$
    $o2 := \mathrm{bool}\,(\,i2 = \mathrm{TRUE} \;\wedge\; \neg\,(i1 = \mathrm{TRUE} \;\wedge\; p2 = \mathrm{FALSE})\,)$
    $p2 := \mathrm{bool}\,(\,i1 = \mathrm{TRUE} \;\wedge\; p2 = \mathrm{FALSE} \;\wedge\; i2 = \mathrm{TRUE}\,)$
  end



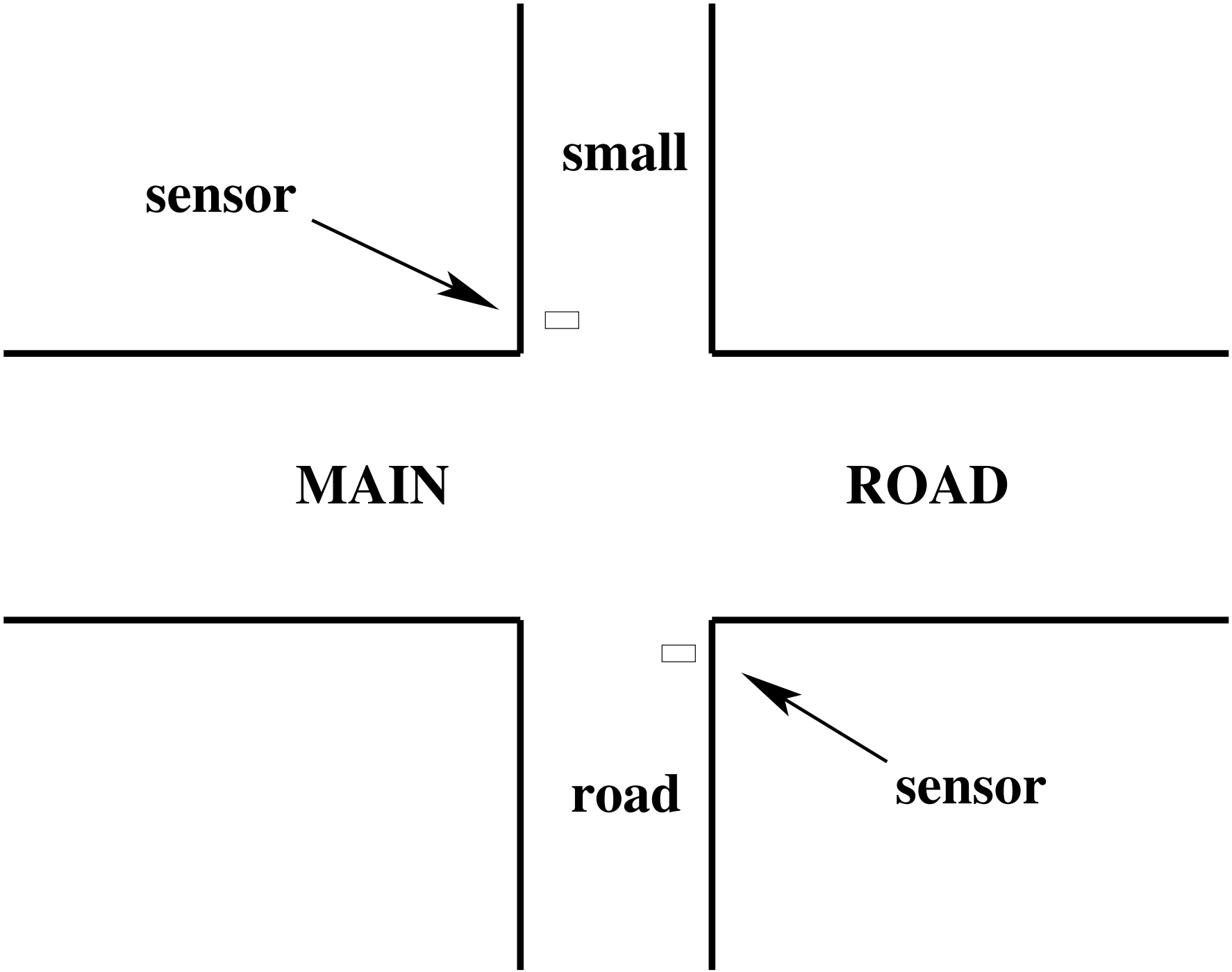
Second Example: A Traffic Light Circuit

- This example is due to Mead and Conway

- Various traffic lights are situated at the crossing between
    - a small road
    - a main road

- Purpose: to give some priority to the traffic on the main road

- Cars on the small road are detected by appropriate sensors


small
sensor
MAIN
ROAD
road
sensor

Informal Specification: The Priority Rules

- **P1**: Main rd. looses priority when cars are present on the small rd.

- **P2**: The loss of priority by main rd. occurs only after a LONG delay
  (even if some cars are present on small road)

- **P3**: Small road looses priority when there are no cars on it

- **P4**: Small road also looses priority after the same LONG delay
  (even if some cars are present on small road)

Informal Specification: The Light Rules

- **L1**: Traffic lights transitions are as usual:
  - green $\rightsquigarrow$ orange
  - orange $\rightsquigarrow$ red
  - red $\rightsquigarrow$ green

- **L2**: An orange light turns red automatically after a SMALL delay

- **L3**: When one light is green or orange, then the other is red

Separation of Concern

- Rules **P1** to **P4** are dealing with traffic priorities

- Rules **L1** to **L3** are dealing with traffic lights

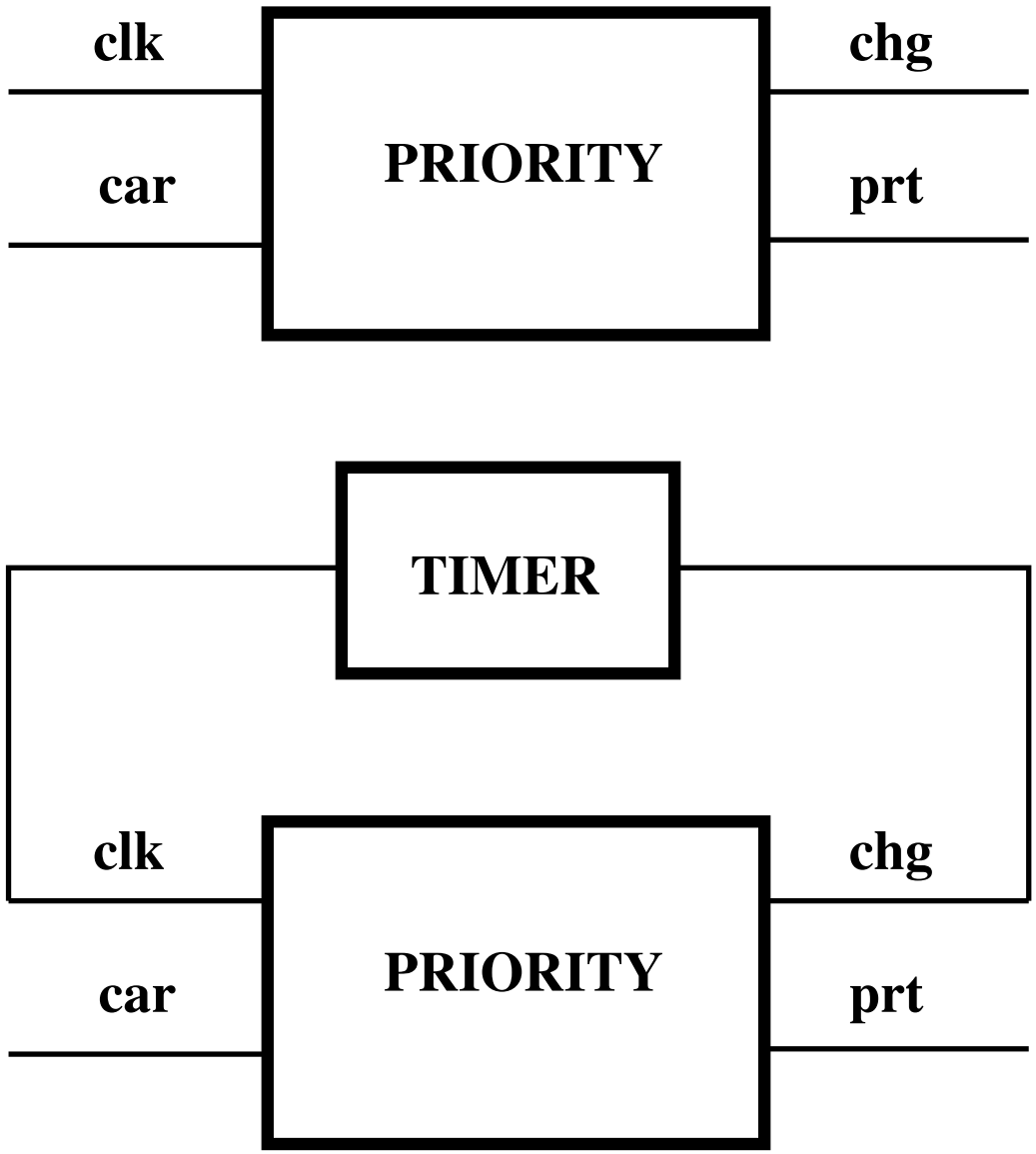
- These questions seem to be independent

- Making two distinct circuits
    - a Priority circuit
    - a Light circuit

- Connecting the two circuits

The Priority Circuit

The Priority Circuit



The Priority Circuit: the Initial State

variables: car, clk, chg, prt

inv0_1: $car \in \mathrm{BOOL}$

inv0_2: $clk \in \mathrm{BOOL}$

inv0_3: $chg \in \mathrm{BOOL}$

inv0_4: $prt \in \mathrm{BOOL}$

- $prt = \mathrm{FALSE}$ means priority on the main road

- $prt = \mathrm{TRUE}$ means priority on the small road

- $clk = \mathrm{TRUE}$ means long delay is over

The Priority Circuit: the Initial Circuit Events

main_to_small

when

$mode = cir$

$prt = \mathrm{FALSE}$

$car = \mathrm{TRUE}$

$clk = \mathrm{TRUE}$

then

$mode := env$

$prt := \mathrm{TRUE}$

$chg := \mathrm{TRUE}$

end

- Priority on main road
- Some cars on small road
- Long delay is over

small_to_main

when

$mode = cir$

$prt = \mathrm{TRUE}$

$car = \mathrm{FALSE} \ \vee \ clk = \mathrm{TRUE}$

then

$mode := env$

$prt := \mathrm{FALSE}$

$chg := \mathrm{TRUE}$

end

- Priority on small road
- No cars on small road or long delay is over

The Priority Circuit: the Initial Circuit Events

do_nothing_1
when
$mode = cir$
$prt = \text{FALSE}$
$car = \text{FALSE} \ \vee \ clk = \text{FALSE}$
then
$mode := env$
$chg := \text{FALSE}$
end

do_nothing_2
when
$mode = cir$
$prt = \text{TRUE}$
$car = \text{TRUE}$
$clk = \text{FALSE}$
then
$mode := env$
$chg := \text{FALSE}$
end

- Priority on main road
- No car on small road or delay is not over

- Priority on small road
- Car on small road
- Delay not ever

The Priority Circuit: the Environment Event

env1

when

$mode = env$

then

$mode := cir$

$car :\in \mathrm{BOOL}$

$clk :\in \mathrm{BOOL}$

end

Circuit Deadlock Freeness

thm0_1:

$$
\begin{array}{l}
mode = cir \\
\Rightarrow \\
\left(\begin{array}{l}
prt = \mathrm{FALSE} \ \wedge \ car = \mathrm{TRUE} \ \wedge \ clk = \mathrm{TRUE} \ \vee \\
prt = \mathrm{TRUE} \ \wedge \ (car = \mathrm{FALSE} \ \vee \ clk = \mathrm{TRUE}) \ \vee \\
prt = \mathrm{FALSE} \ \wedge \ (car = \mathrm{FALSE} \ \vee \ clk = \mathrm{FALSE}) \ \vee \\
prt = \mathrm{TRUE} \ \wedge \ car = \mathrm{TRUE} \ \wedge \ clk = \mathrm{FALSE}
\end{array} \right)
\end{array}
$$

- The circuit is deterministic

Building the Final Priority Circuit

main_to_small
when
$mode = cir$
$prt = \text{FALSE}$
$car = \text{TRUE}$
$clk = \text{TRUE}$
then
$mode := env$
$prt := \text{TRUE}$
$chg := \text{TRUE}$
end

small_to_main
when
$mode = cir$
$prt = \text{TRUE}$
$car = \text{FALSE} \vee$
$clk = \text{TRUE}$
then
$mode := env$
$prt := \text{FALSE}$
$chg := \text{TRUE}$
end

do_nothing_1
when
$mode = cir$
$prt = \text{FALSE}$
$car = \text{FALSE} \vee$
$clk = \text{FALSE}$
then
$mode := env$
$chg := \text{FALSE}$
$prt := \text{FALSE}$
end

do_nothing_2
when
$mode = cir$
$prt = \text{TRUE}$
$car = \text{TRUE}$
$clk = \text{FALSE}$
then
$mode := env$
$chg := \text{FALSE}$
$prt := \text{TRUE}$
end

priority
when
$mode = cir$
then
$mode := env$

$$prt := \text{bool}\left(\begin{array}{l}(prt = \text{FALSE} \wedge car = \text{TRUE} \wedge clk = \text{TRUE}) \vee \\ (prt = \text{TRUE} \wedge car = \text{TRUE} \wedge clk = \text{FALSE})\end{array}\right)$$

$$chg := \text{bool}\left(\begin{array}{l}(prt = \text{FALSE} \wedge car = \text{TRUE} \wedge clk = \text{TRUE}) \vee \\ (prt = \text{TRUE} \wedge (car = \text{FALSE} \vee clk = \text{TRUE}))\end{array}\right)$$

end

Two Identical Circuits (Proof by cases)

priority
  when
    $mode = cir$
  then
    $mode := env$
    $prt := \text{bool} \left(\begin{array}{l} (prt = \text{FALSE} \ \wedge \ car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (prt = \text{TRUE} \ \wedge \ car = \text{TRUE} \ \wedge \ clk = \text{FALSE}) \end{array} \right)$
    $chg := \text{bool} \left(\begin{array}{l} (prt = \text{FALSE} \ \wedge \ car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (prt = \text{TRUE} \ \wedge \ (car = \text{FALSE} \ \vee \ clk = \text{TRUE})) \end{array} \right)$
  end

priority
  when
    $mode = cir$
  then
    $mode := env$
    $prt := \text{bool} \left(\begin{array}{l} (prt = \text{TRUE} \ \wedge \ \neg \left(\begin{array}{l} (car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE}) \end{array} \right) \ \vee \\ (prt = \text{FALSE} \ \wedge \ \left(\begin{array}{l} (car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE}) \end{array} \right) \end{array} \right)$
    $chg := \text{bool} \left(\begin{array}{l} (car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE}) \end{array} \right)$
  end

The Second Circuit

priority
when
$mode = cir$
then
$mode := env$

$$prt := \text{bool}\left(\begin{array}{l}(prt = \text{TRUE} \ \wedge \ \neg \left(\begin{array}{l}(car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE})\end{array}\right) \ \vee \\ (prt = \text{FALSE} \ \wedge \ \left(\begin{array}{l}(car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE})\end{array}\right)\end{array}\right)$$

$$chg := \text{bool}\left(\begin{array}{l}(car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE})\end{array}\right)$$

end

This circuit is interesting because it contains three times the same fragment:

$$\begin{array}{l}(car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE})\end{array}$$

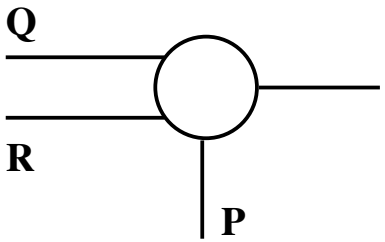
Introducing an IF-gate

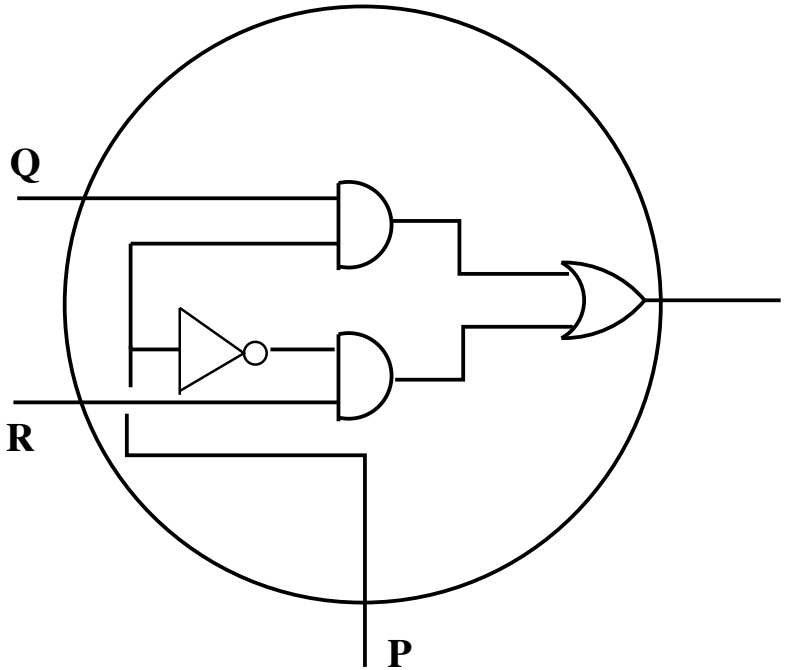
- The previous circuit contains several occurrences of the following fragment:

$$(P \wedge Q) \ \vee \ (\neg P \wedge R)$$

- This will be economically represented by a special "IF-gate".

- Such a gate is pictorially represented and defined as follows:





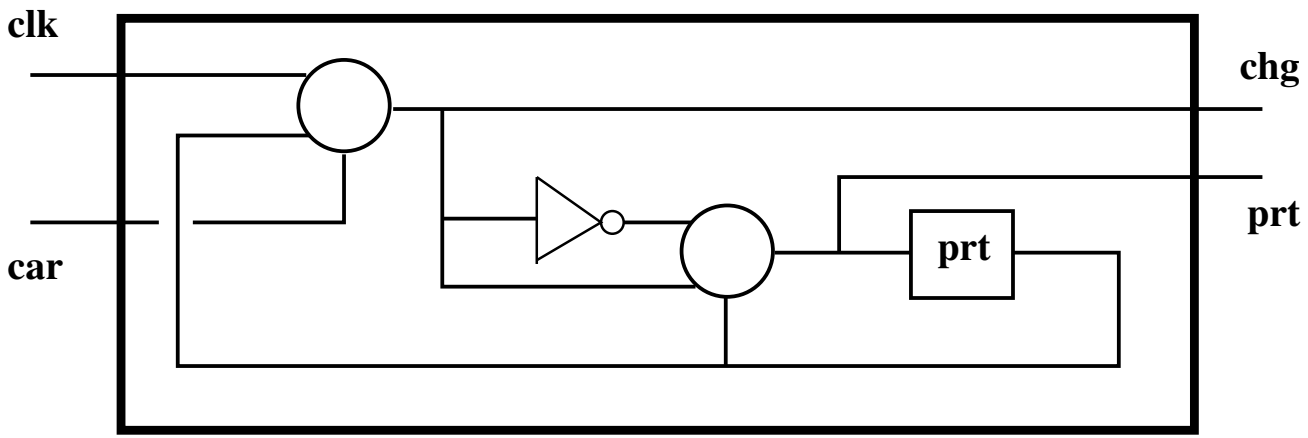
The Priority Circuit

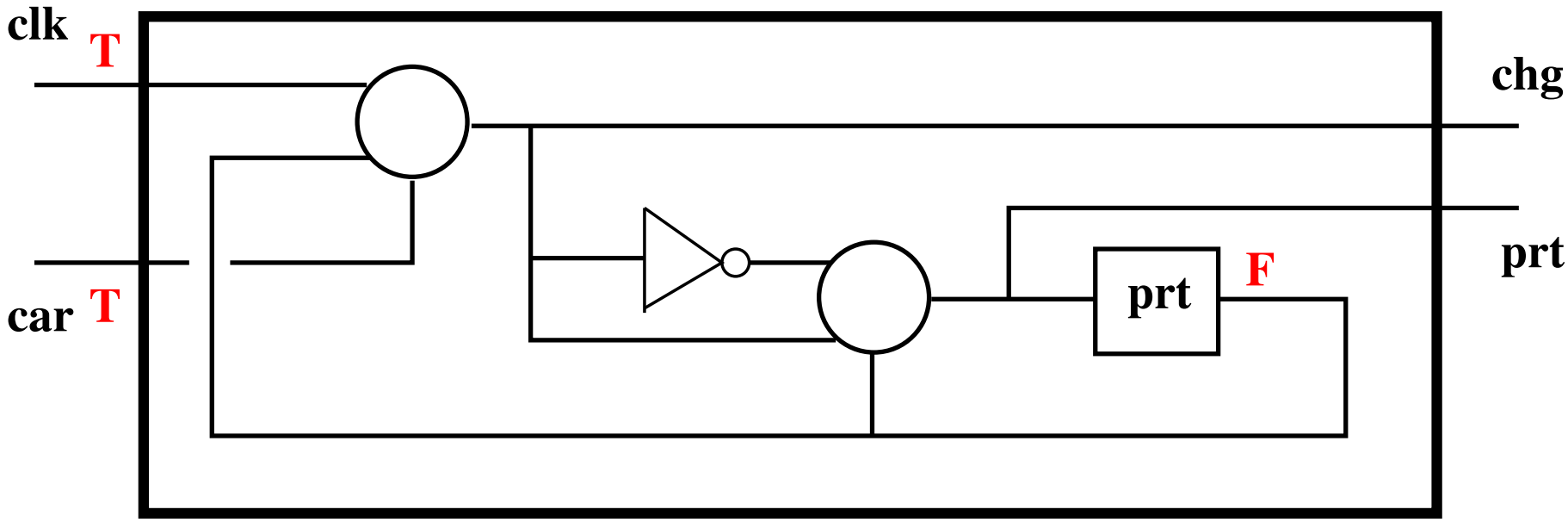
priority
when
$mode = cir$
then
$mode := env$

$$prt := \text{bool}\left(\begin{array}{l}(prt = \text{TRUE} \ \wedge \ \neg \left(\begin{array}{l}(car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE})\end{array}\right) \ \vee \\ (prt = \text{FALSE} \ \wedge \ \left(\begin{array}{l}(car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE})\end{array}\right)\end{array}\right)$$

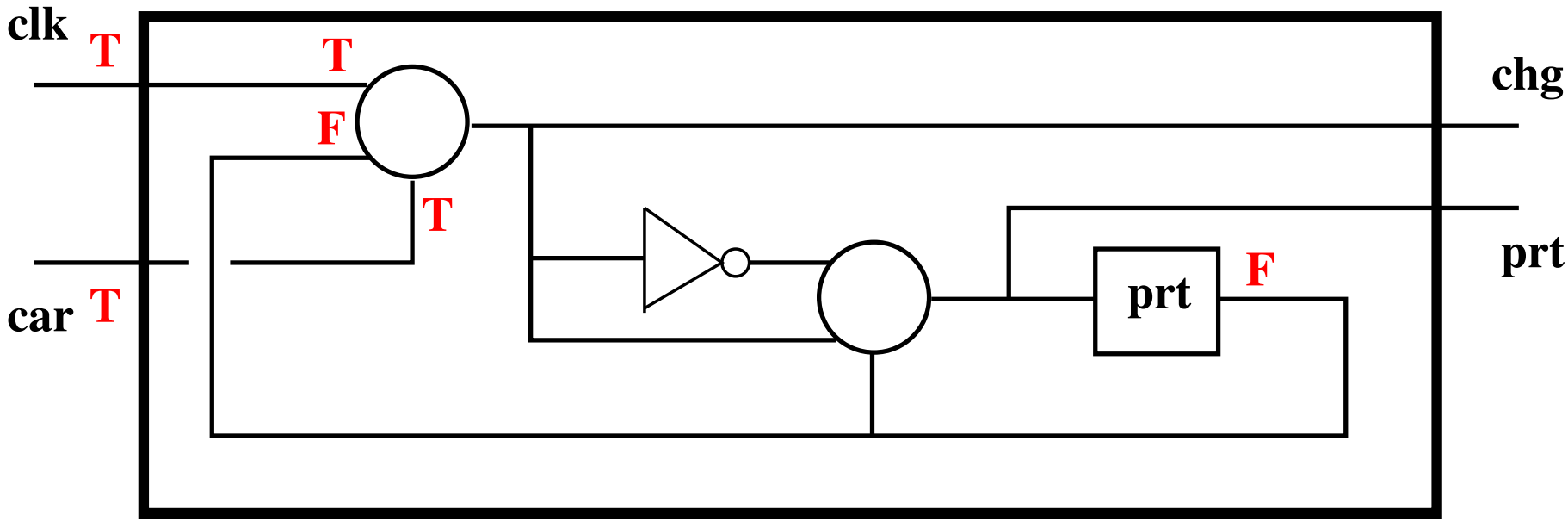
$$chg := \text{bool}\left(\begin{array}{l}(car = \text{TRUE} \ \wedge \ clk = \text{TRUE}) \ \vee \\ (car = \text{FALSE} \ \wedge \ prt = \text{TRUE})\end{array}\right)$$

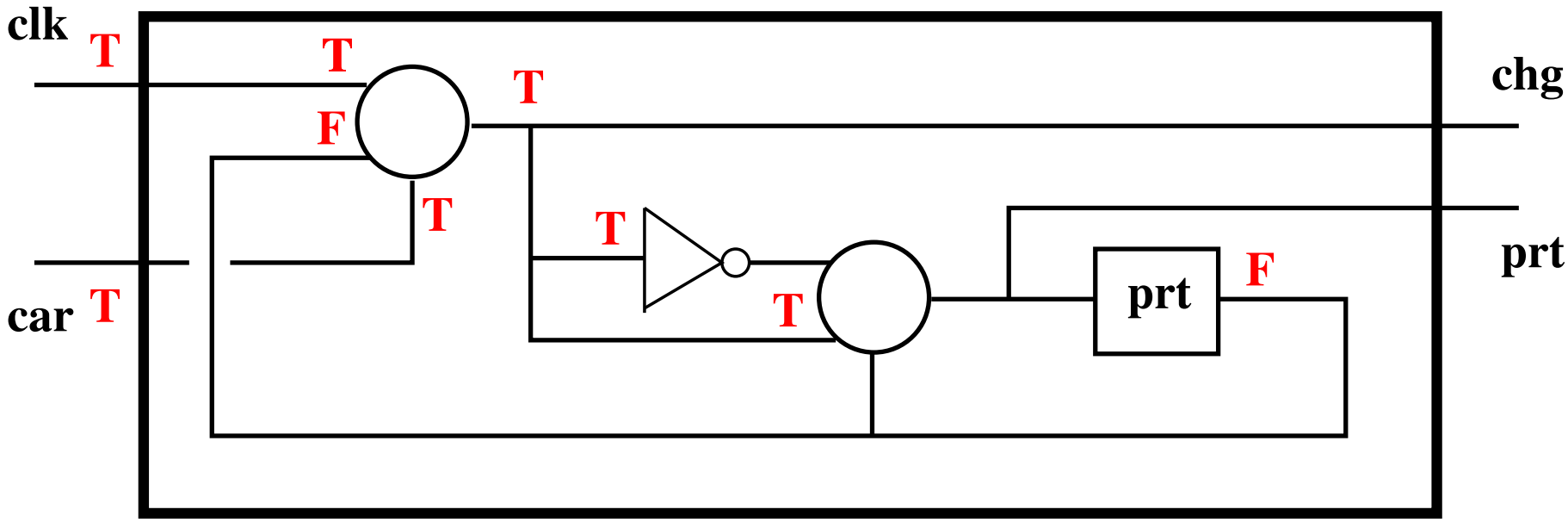
end

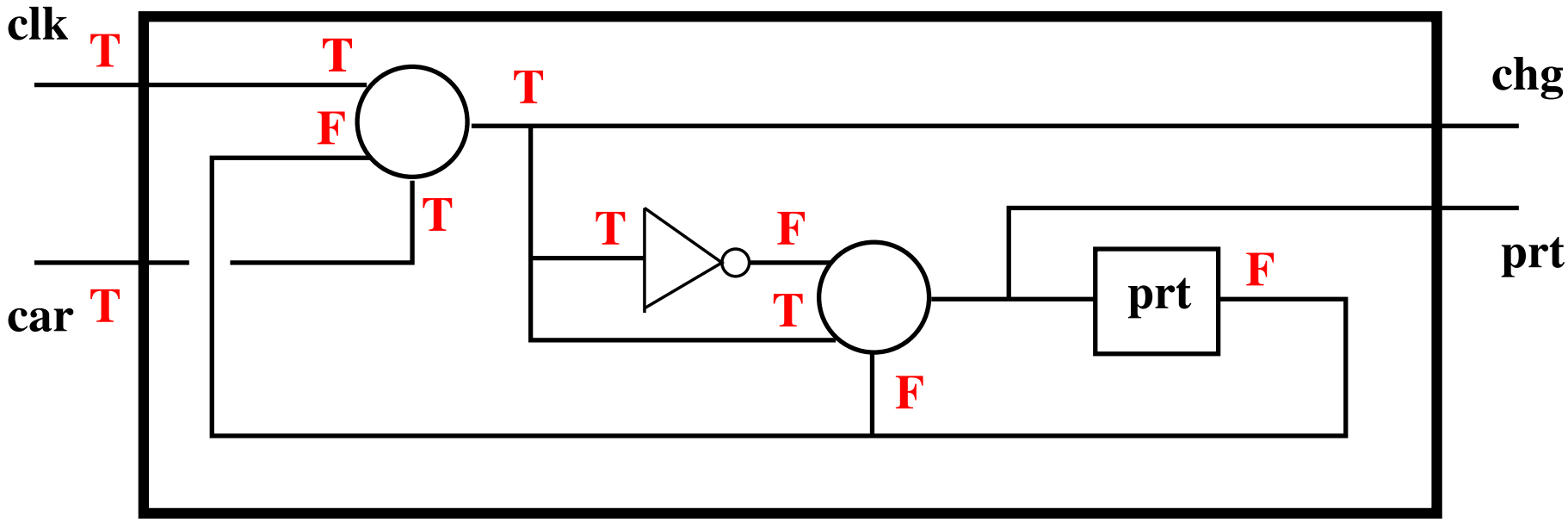


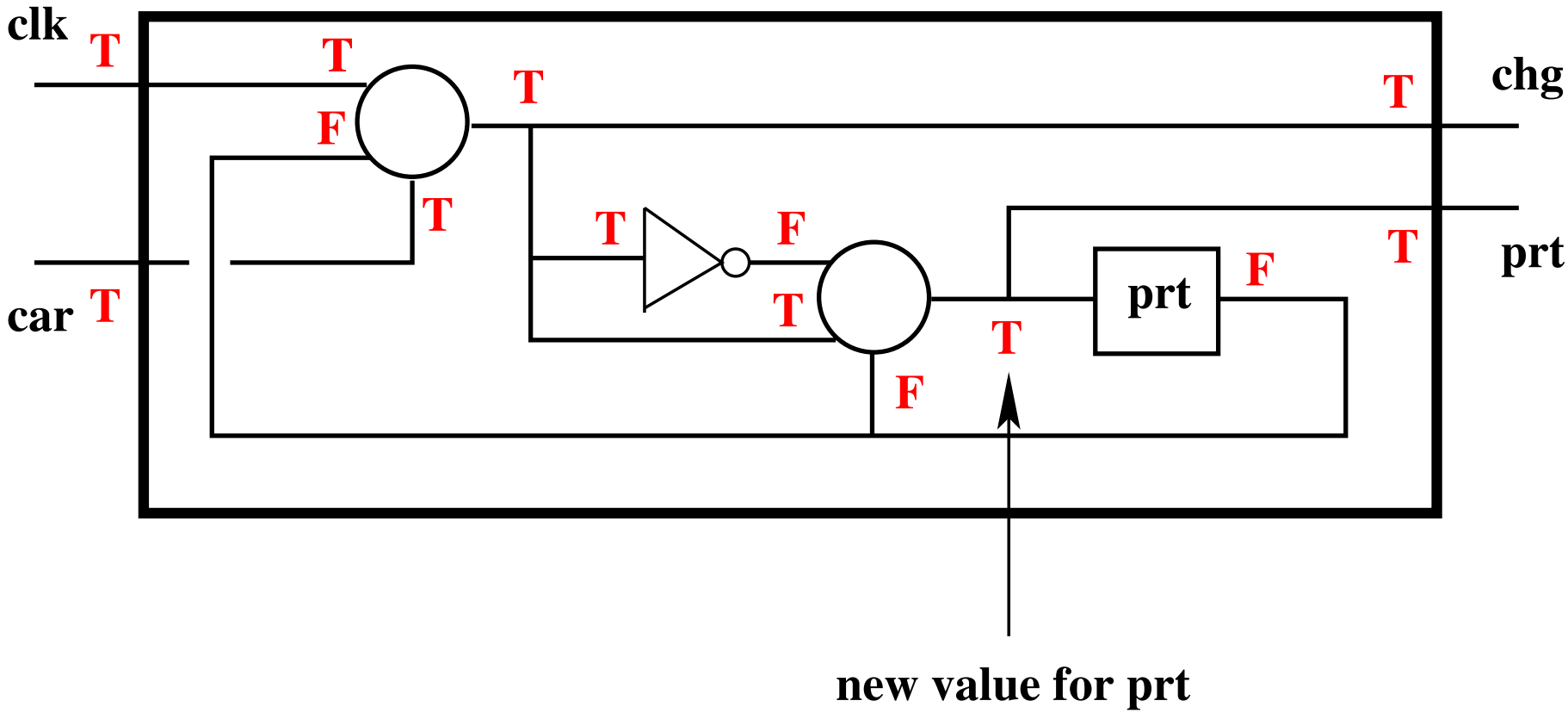


- priority on main road
- car on small road
- long delay is over

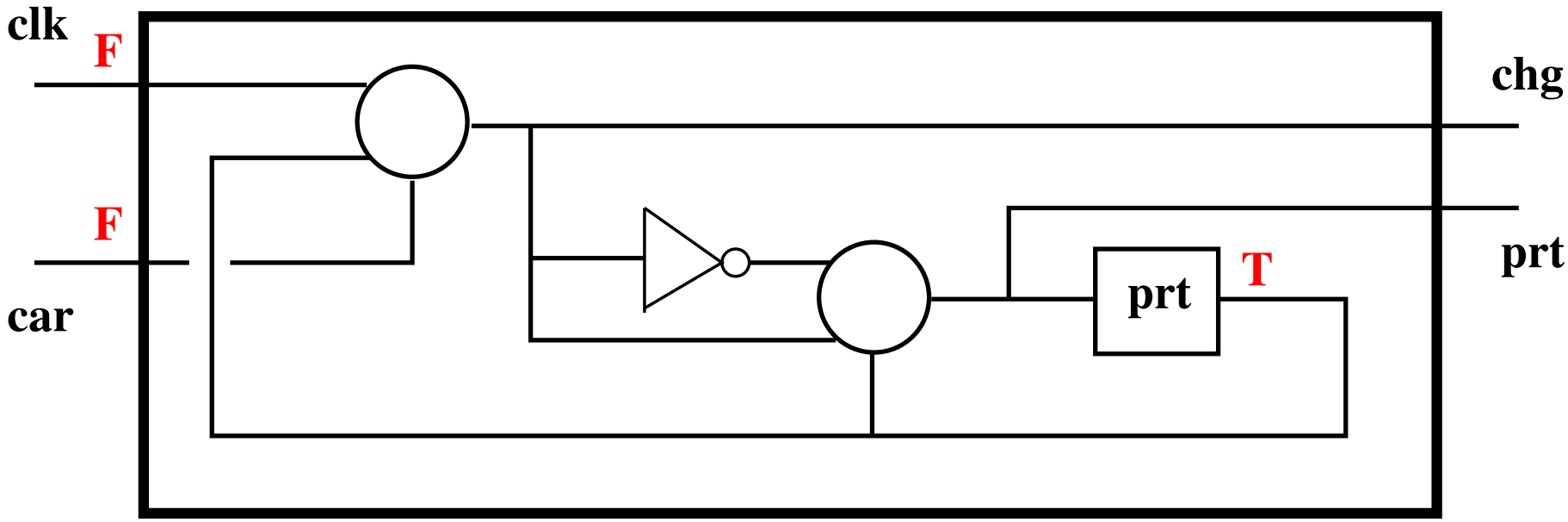

clk
T
T
F
T
car
T
prt
F
prt
chg
prt


clk
T
T
T
F
T
T
T
car
T
T
prt
F
prt
chg

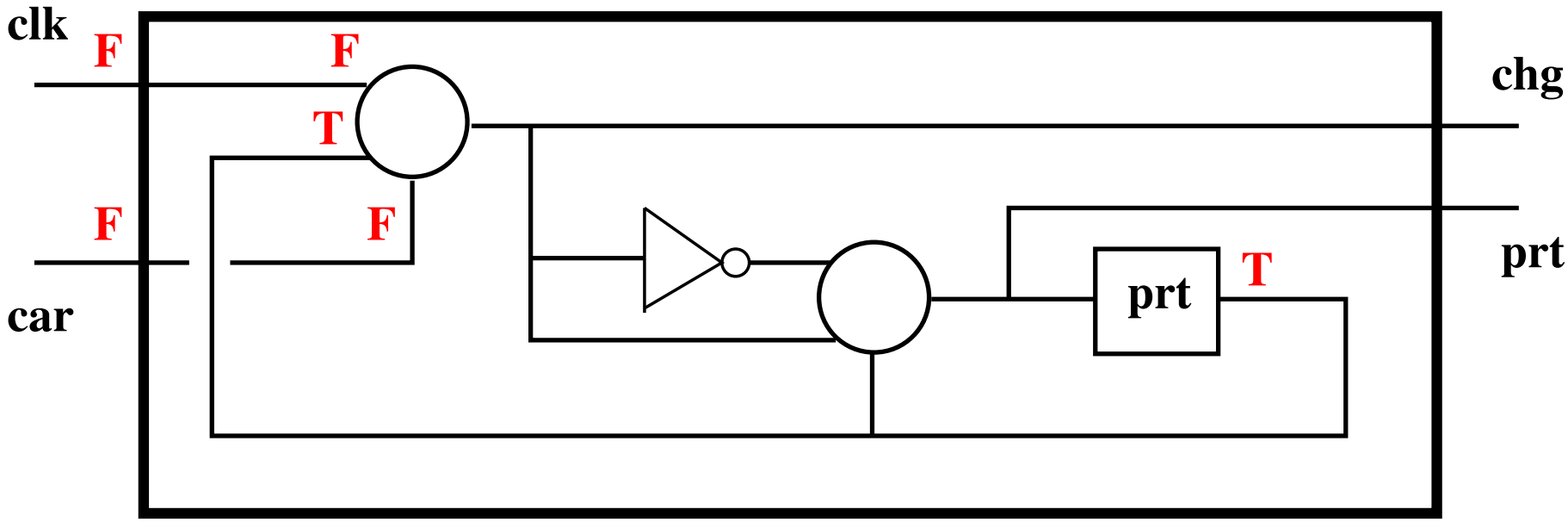

clk
T
T
F
T
T
T
T
F
car
T
T
F
prt
F
prt
chg

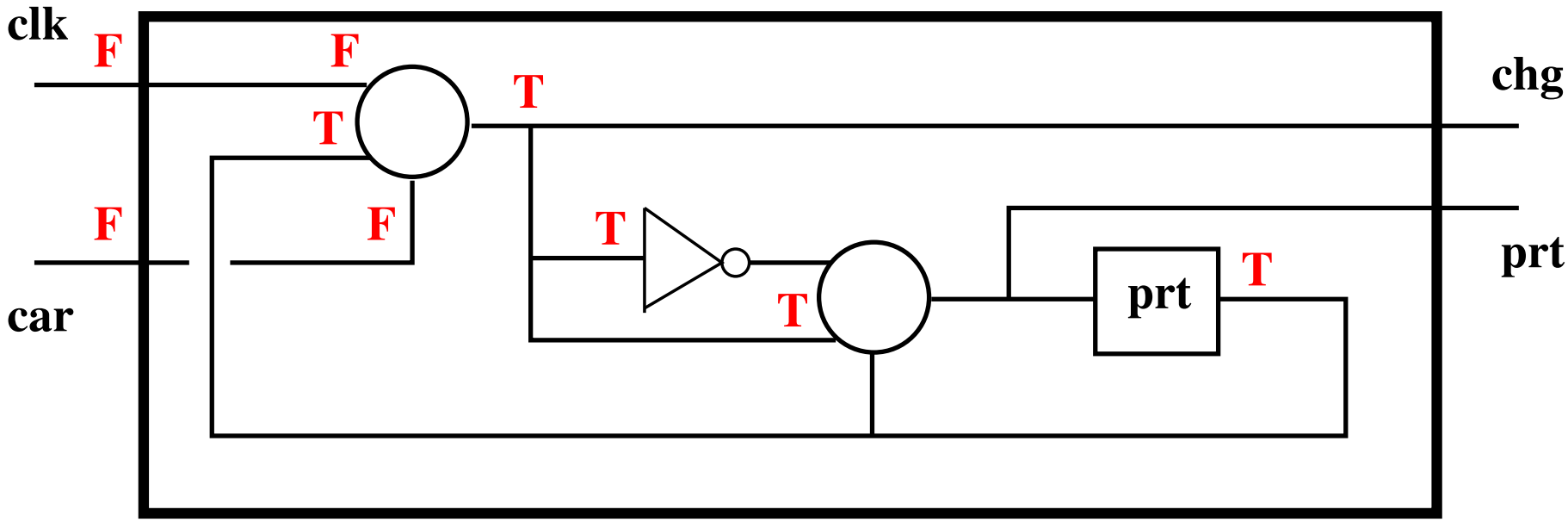

clk
T
T
F
T
T
T
chg
T
T
F
T
T
car
T
T
F
prt
T
prt
T
F
new value for prt

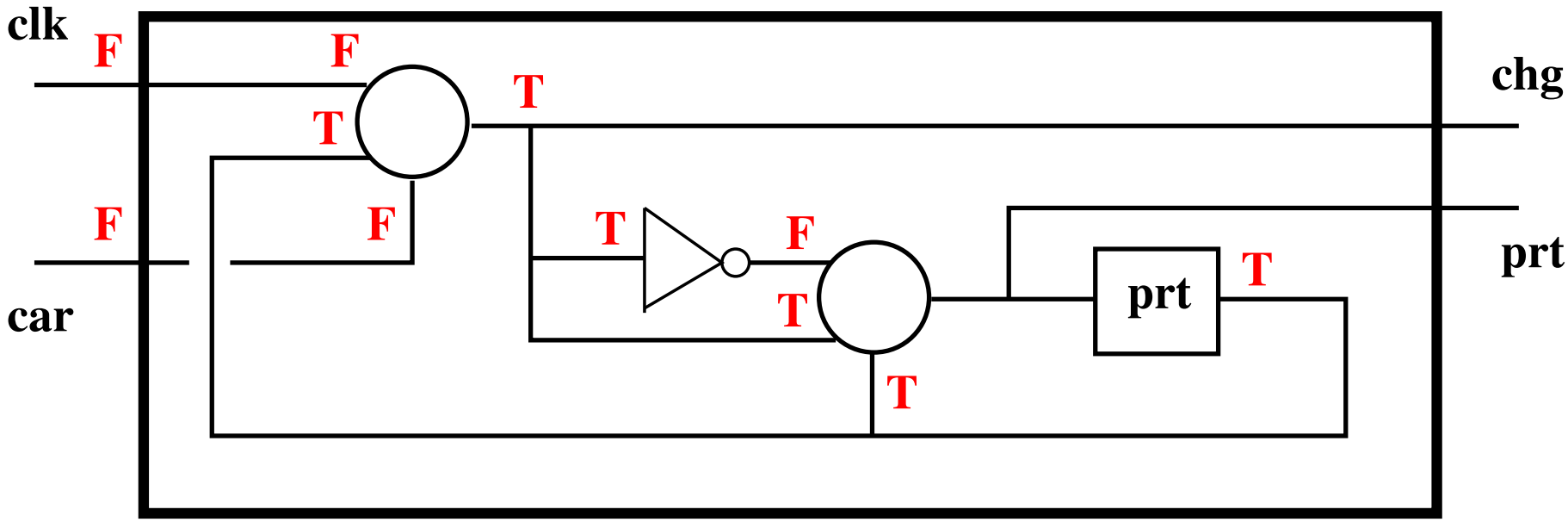
Another Computation

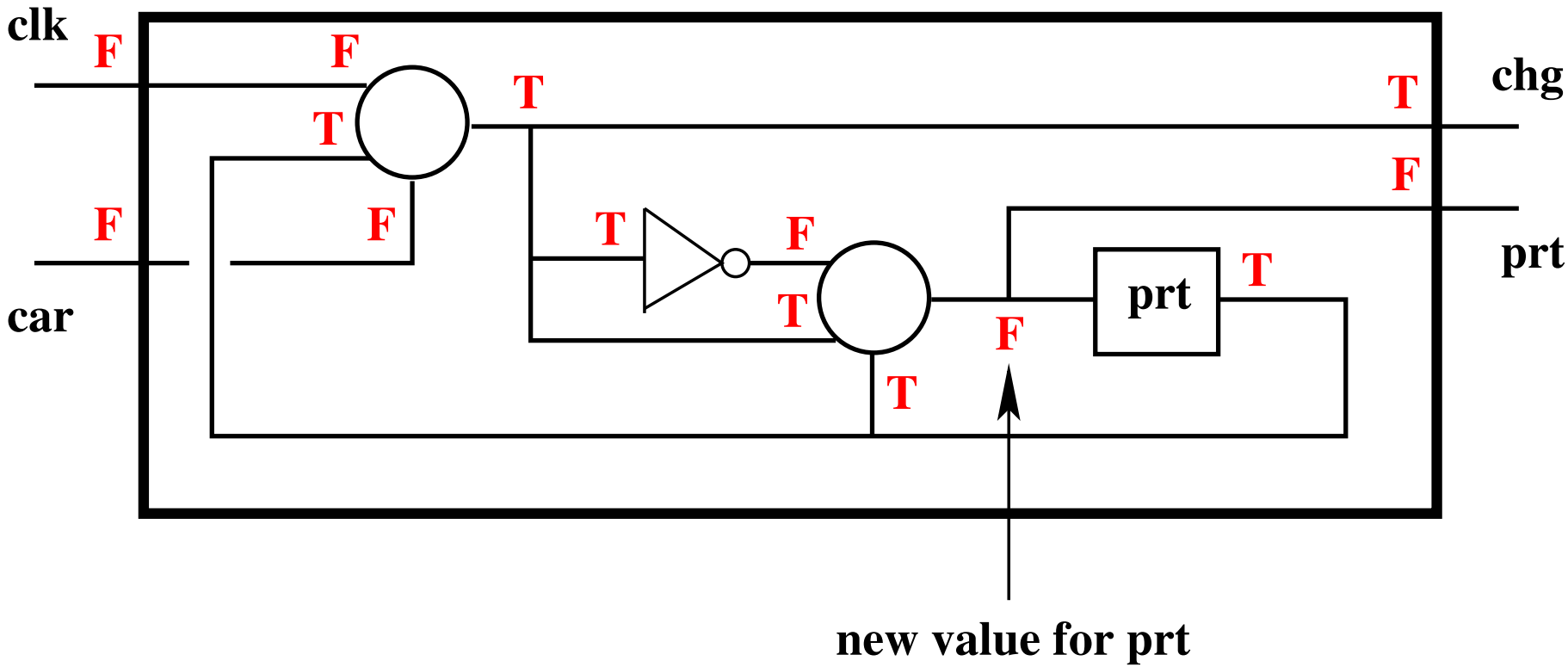


- Priority on small road
- No cars on small road
- Long delay is not over


clk
F
F
T
F
F
car
chg
prt
prt
T

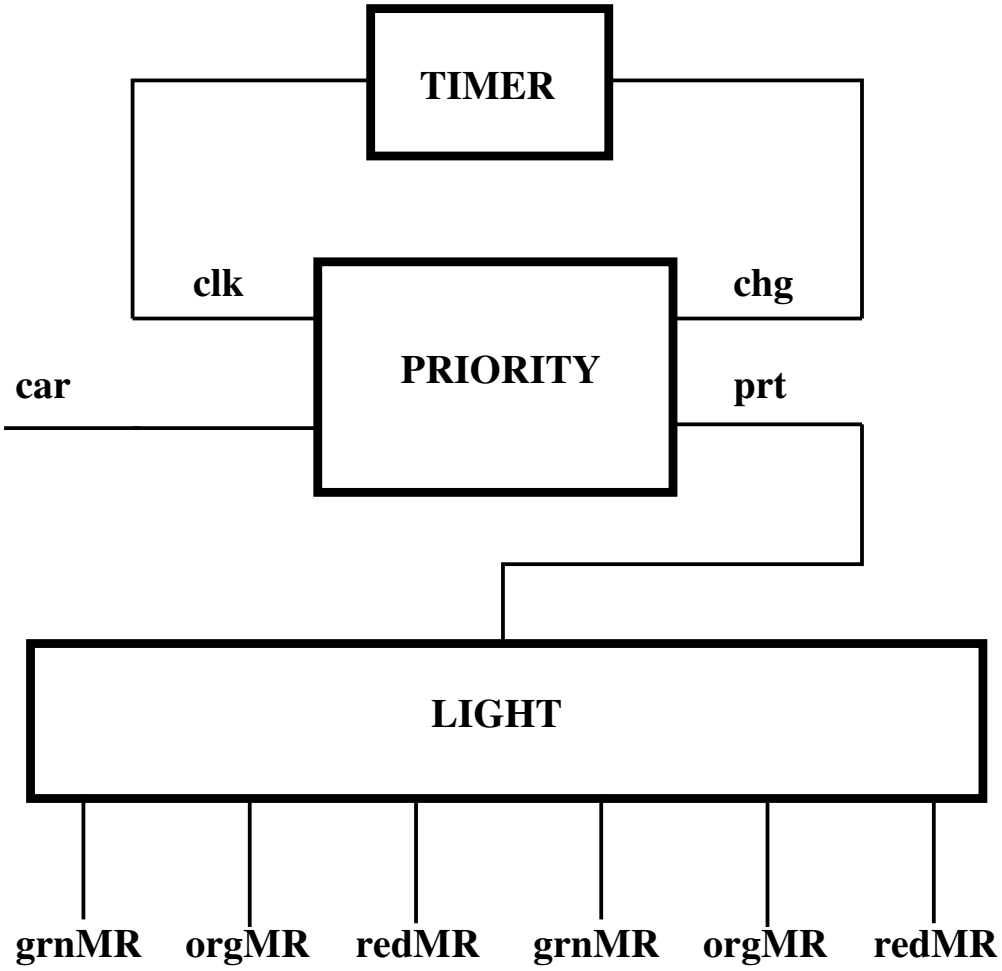

clk
F
F
T
T
chg
F
F
T
prt
car
T
prt
T


clk
F
F
T
T
chg
F
F
T
F
T
prt
car
T
prt
T
T


clk
F
F
T
T
chg
T
F
F
F
T
F
prt
car
T
prt
T
F
T
new value for prt

The "Light" Circuit

- We connect the PRIORITY circuit to the LIGHT circuit



The "Upper" Circuit

- We start with an abstraction of the LIGHT ciruit
- It yields the colors for the main road
- The two red colors $rd1$ and $rd2$ are there for symmetry with grn and org

variables: $prt, grn, org, rd1, rd2$

inv0_1: $prt \in \mathrm{BOOL}$

inv0_2: $grn \in \mathrm{BOOL}$

inv0_3: $org \in \mathrm{BOOL}$

inv0_4: $rd1 \in \mathrm{BOOL}$

inv0_5: $rd2 \in \mathrm{BOOL}$

The "Upper" Circuit: More Invariants

- Always one color
- At most one color

inv0_6: $grn = \text{TRUE} \ \vee \ org = \text{TRUE} \ \vee \ rd1 = \text{TRUE} \ \vee \ rd2 = \text{TRUE}$

inv0_7: $grn = \text{TRUE} \Rightarrow org = \text{FALSE} \ \wedge \ rd1 = \text{FALSE} \ \wedge \ rd2 = \text{FALSE}$

inv0_8: $org = \text{TRUE} \Rightarrow grn = \text{FALSE} \ \wedge \ rd1 = \text{FALSE} \ \wedge \ rd2 = \text{FALSE}$

inv0_9: $rd1 = \text{TRUE} \Rightarrow grn = \text{FALSE} \ \wedge \ org = \text{FALSE} \ \wedge \ rd2 = \text{FALSE}$

inv0_10: $rd2 = \text{TRUE} \Rightarrow grn = \text{FALSE} \ \wedge \ org = \text{FALSE} \ \wedge \ rd1 = \text{FALSE}$

The "Upper" Circuit: the Events

grn_to_org

when

$mode = cir$

$prt = \text{TRUE}$

$grn = \text{TRUE}$

then

$mode := env$

$grn := \text{FALSE}$

$org := \text{TRUE}$

end

org_to_rd1

when

$mode = cir$

$org = \text{TRUE}$

then

$mode := env$

$org := \text{FALSE}$

$rd1 := \text{TRUE}$

end

The "Upper" Circuit: the Events

rd1_to_rd2

when

$mode = cir$

$prt = \text{FALSE}$

$rd1 = \text{TRUE}$

then

$mode := env$

$rd1 := \text{FALSE}$

$rd2 := \text{TRUE}$

end

rd2_to_grn

when

$mode = cir$

$rd2 = \text{TRUE}$

then

$mode := env$

$grn := \text{TRUE}$

$rd2 := \text{FALSE}$

end

The "Upper" Circuit: the Events

grn_to_nth

when

$mode = cir$

$grn = \mathrm{TRUE}$

$prt = \mathrm{FALSE}$

then

$mode := env$

end

rd1_to_nth

when

$mode = cir$

$rd1 = \mathrm{TRUE}$

$prt = \mathrm{TRUE}$

then

$mode := env$

end

The "Upper" Circuit: the Environment Event

env_evt
when
$mode = env$
then
$mode := cir$
$prt :\in BOOL$
end

Refinement: Adding the "Lower" Circuit

variables: $prt, grn, org, rd1, rd2$

grn_MR, org_MR, red_MR

grn_SR, org_SR, red_SR

inv1_1: $grn\_MR \in \text{BOOL}$

inv1_2: $org\_MR \in \text{BOOL}$

inv1_3: $red\_MR \in \text{BOOL}$

inv1_4: $grn\_SR \in \text{BOOL}$

inv1_5: $org\_SR \in \text{BOOL}$

inv1_6: $red\_SR \in \text{BOOL}$

The "Lower" Circuit: More Invariants

inv1_7: $grn\_MR = grn$

inv1_8: $org\_MR = org$

inv1_9: $red\_MR = \text{TRUE} \Leftrightarrow (rd1 = \text{TRUE} \vee rd2 = \text{TRUE})$

inv1_10: $grn\_SR = rd1$

inv1_11: $org\_SR = rd2$

inv1_12: $red\_SR = \text{TRUE} \Leftrightarrow (grn = \text{TRUE} \vee org = \text{TRUE})$

The "Lower" Circuit: Two Important Theorems

- Lights are mutually exclusive

thm1_1: $red\_MR = \text{TRUE} \Leftrightarrow (grn\_SR = TRUE \vee org\_SR = \text{TRUE})$

thm1_2: $red\_SR = \text{TRUE} \Leftrightarrow (grn\_MR = TRUE \vee org\_MR = \text{TRUE})$

The "Lower" Circuit: Events

grn_to_org

when

$mode = cir$

$prt = \text{TRUE}$

$grn = \text{TRUE}$

then

$mode := env$

$grn := \text{FALSE}$

$org := \text{TRUE}$

$grn\_MR := \text{FALSE}$

$org\_MR := \text{TRUE}$

end

org_to_rd1

when

$mode = cir$

$org = \text{TRUE}$

then

$mode := env$

$org := \text{FALSE}$

$rd1 := \text{TRUE}$

$org\_MR := \text{FALSE}$

$red\_MR := \text{TRUE}$

$grn\_SR := \text{TRUE}$

$red\_SR := \text{FALSE}$

end

The "Lower" Circuit: Events

rd1_to_rd2

when

$mode = cir$
$prt = \text{FALSE}$
$rd1 = \text{TRUE}$

then

$mode := env$
$rd1 := \text{FALSE}$
$rd2 := \text{TRUE}$
$org\_SR := \text{TRUE}$
$grn\_SR := \text{FALSE}$

end

rd2_to_grn

when

$mode = cir$
$rd2 = \text{TRUE}$

then

$mode := env$
$grn := \text{TRUE}$
$rd2 := \text{FALSE}$
$grn\_MR := \text{TRUE}$
$red\_MR := \text{FALSE}$
$org\_SR := \text{FALSE}$
$red\_SR := \text{TRUE}$

end

The Final "Light" Circuit

light

when

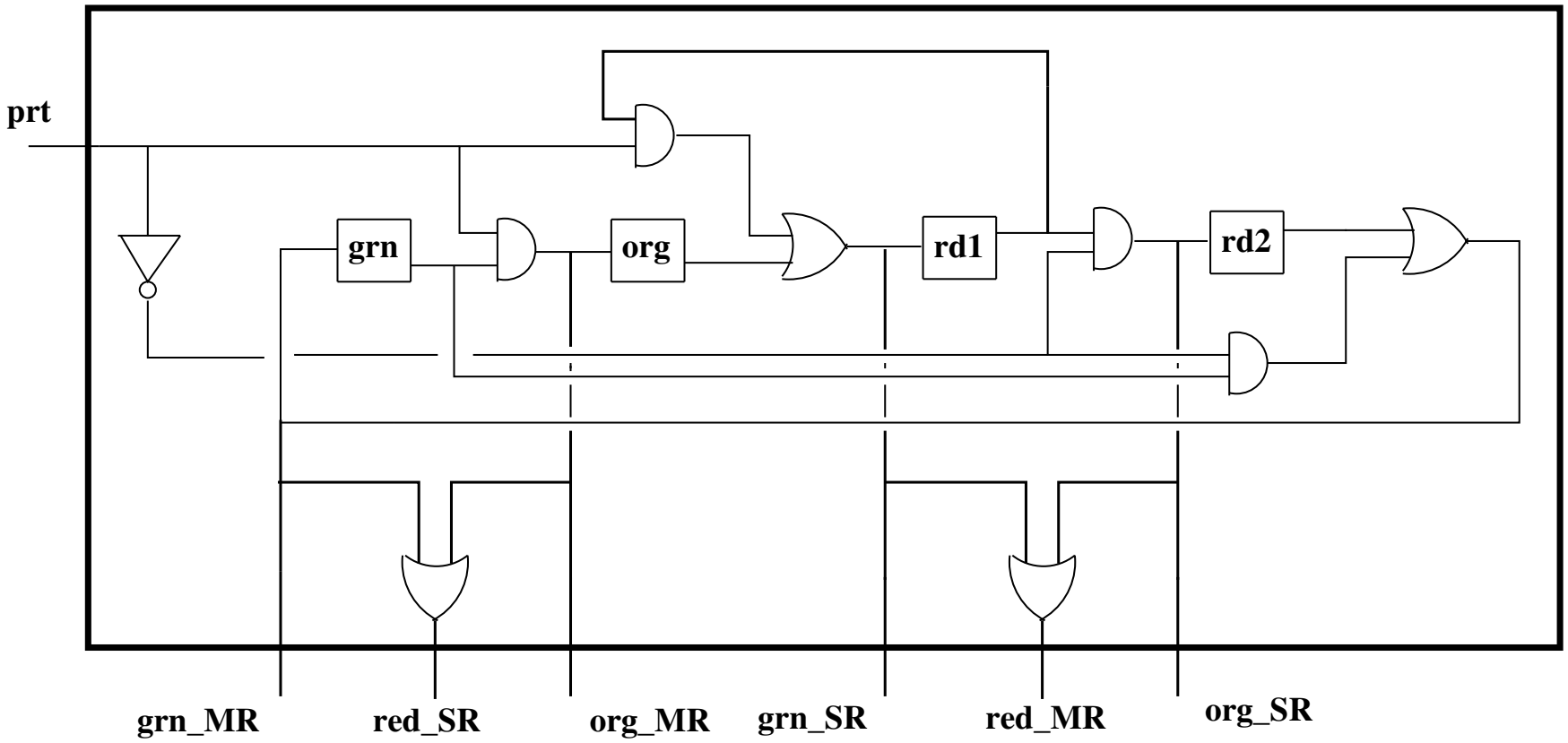
$mode = cir$

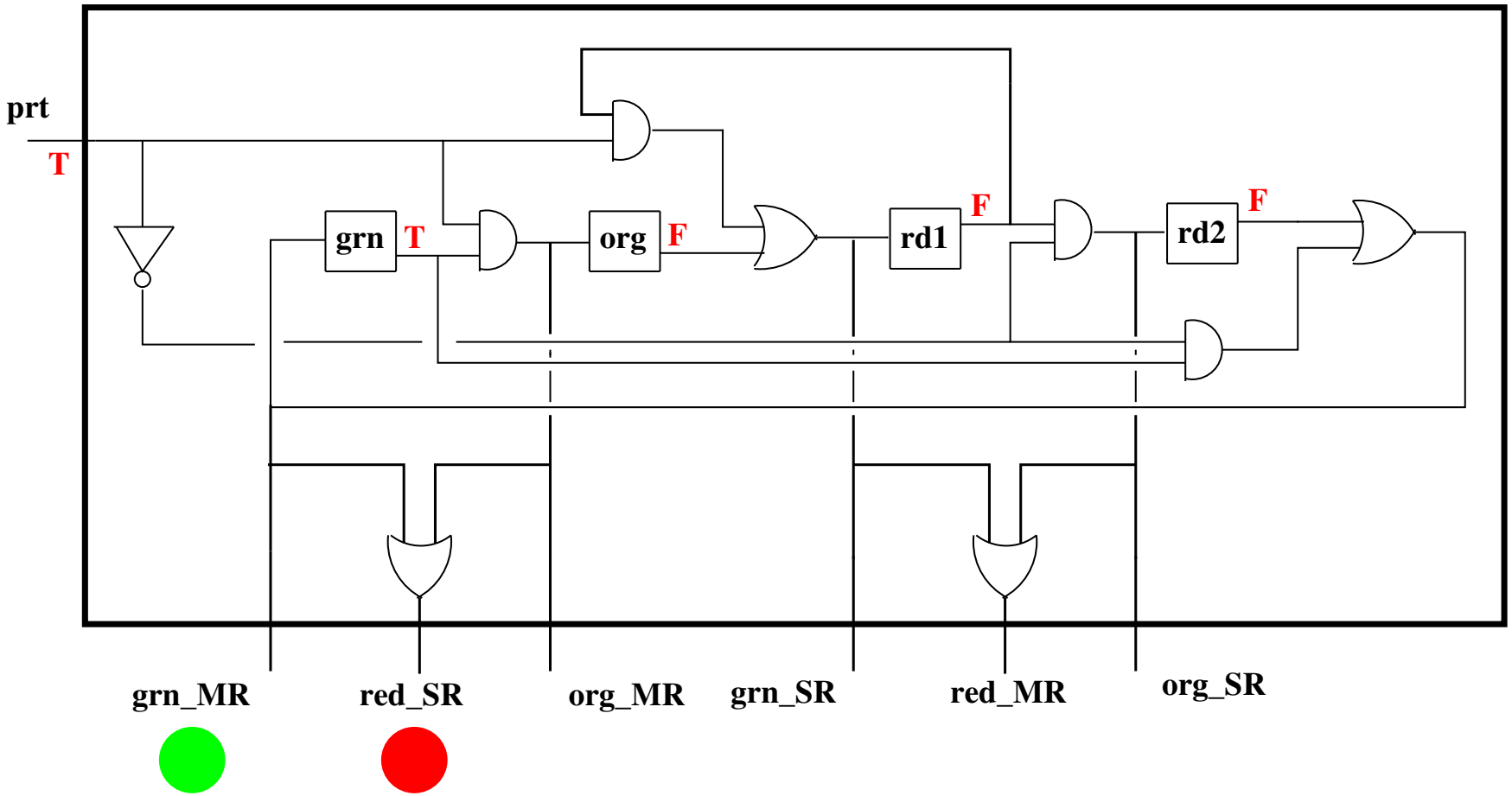
then

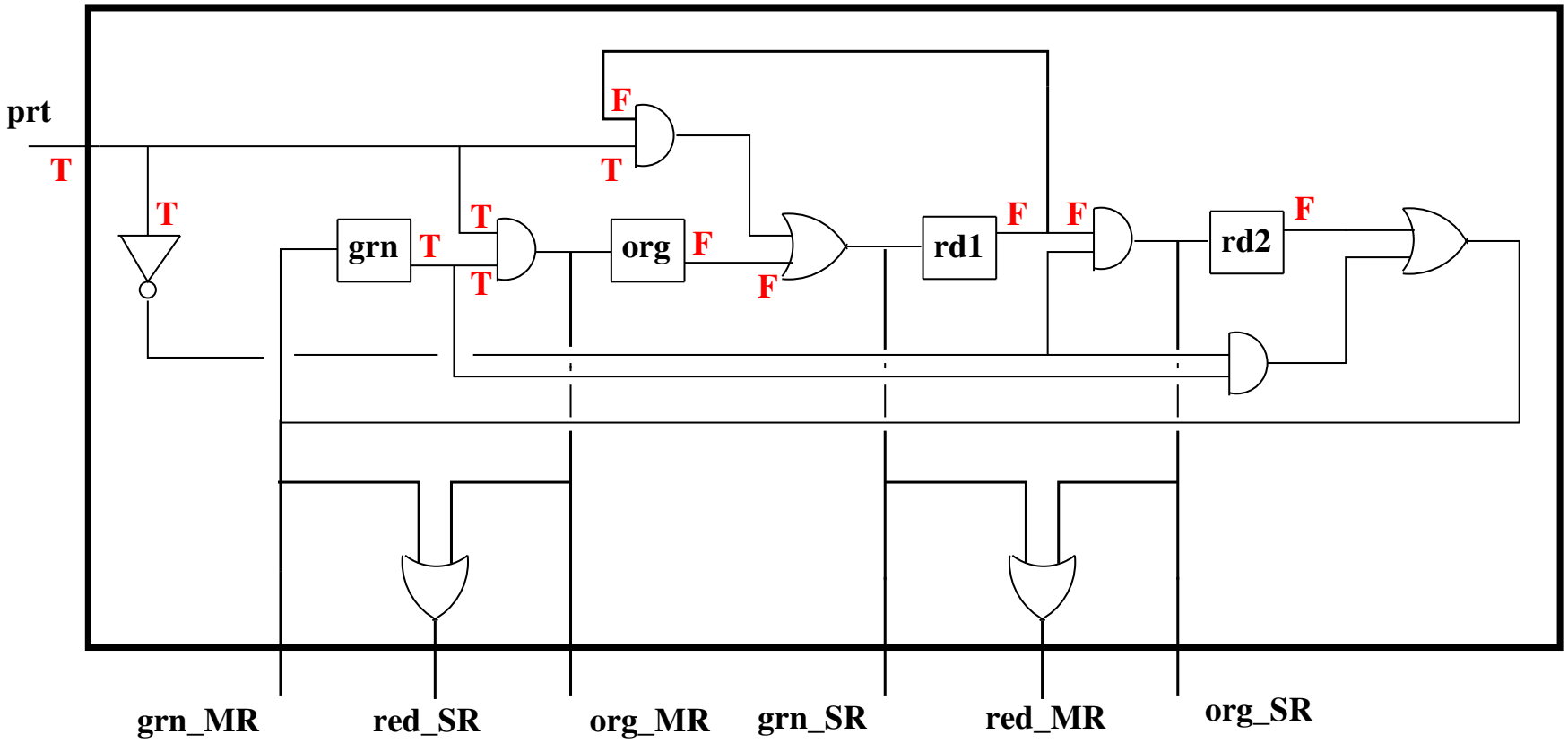
$$
\begin{aligned}
&mode := env \\
&grn := \text{bool}(\, rd2 = \text{TRUE} \ \vee \ (prt = \text{FALSE} \ \wedge \ grn = \text{TRUE})\,) \\
&org := \text{bool}(\, prt = \text{TRUE} \ \wedge \ grn = \text{TRUE}\,) \\
&rd1 := \text{bool}(\, org = \text{TRUE} \ \vee \ (prt = \text{TRUE} \ \wedge \ rd1 = \text{TRUE})\,) \\
&rd2 := \text{bool}(\, prt = \text{FALSE} \ \wedge \ rd1 = \text{TRUE}\,) \\
&grn\_MR := \text{bool}(\, rd2 = \text{TRUE} \ \vee \ (prt = \text{FALSE} \ \wedge \ grn = \text{TRUE})\,) \\
&org\_MR := \text{bool}(\, prt = \text{TRUE} \ \wedge \ grn = \text{TRUE}\,) \\
&red\_MR := \text{bool}(\, org = \text{TRUE} \ \vee \ (prt = \text{TRUE} \ \wedge \ rd1 = \text{TRUE}) \ \vee \\
&\qquad\qquad (prt = \text{FALSE} \ \wedge \ rd1 = \text{TRUE})\,) \\
&grn\_SR := \text{bool}(\, org = \text{TRUE} \ \vee \ (prt = \text{TRUE} \ \wedge \ rd1 = \text{TRUE})\,) \\
&org\_MR := \text{bool}(\, prt = \text{FALSE} \ \wedge \ rd1 = \text{TRUE}\,) \\
&red\_MR := \text{bool}(\, rd2 = \text{TRUE} \ \vee \ (prt = \text{FALSE} \ \wedge \ grn = \text{TRUE}) \ \vee \\
&\qquad\qquad (prt = \text{TRUE} \ \wedge \ grn = \text{TRUE})\,)
\end{aligned}
$$

end

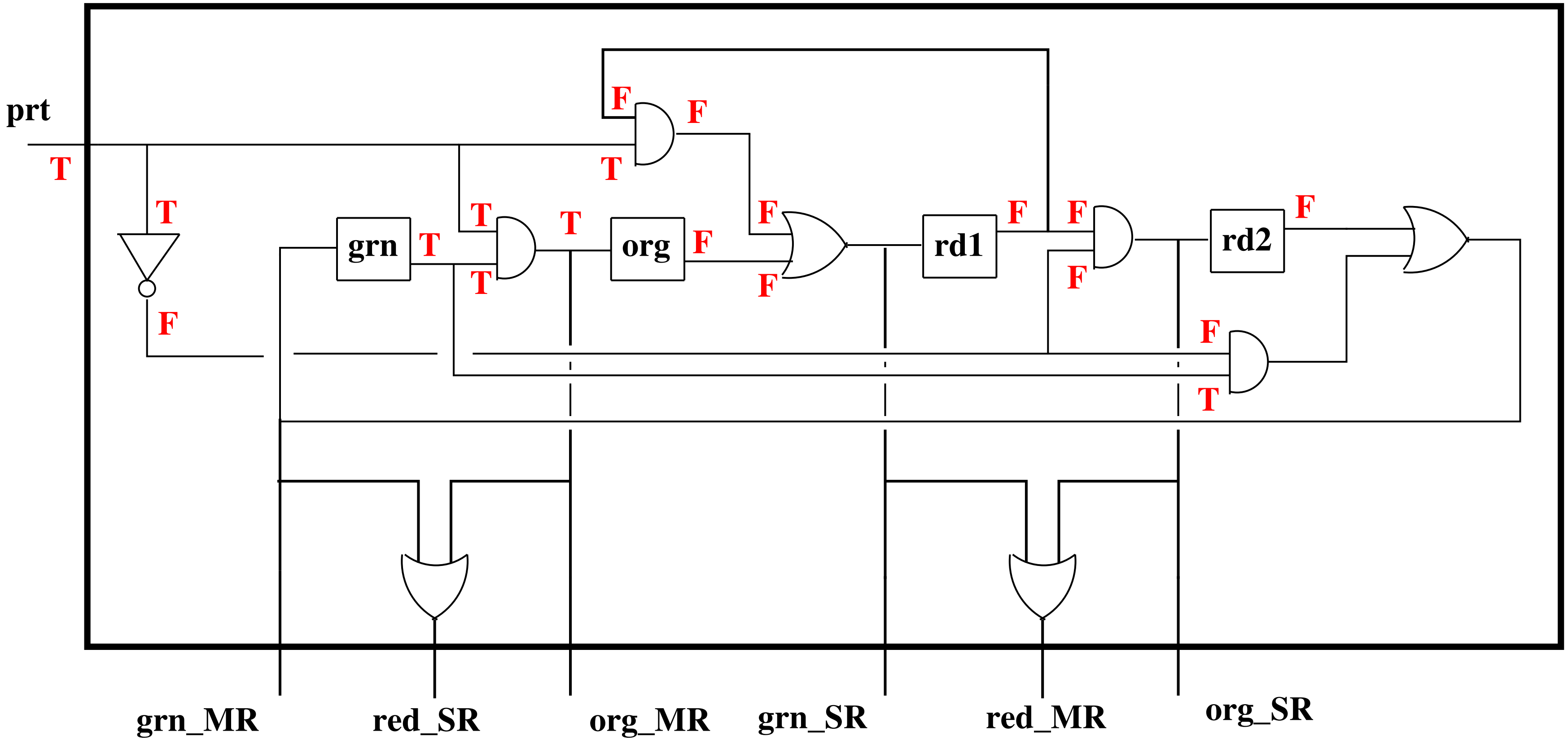
The Final "Light" Circuit

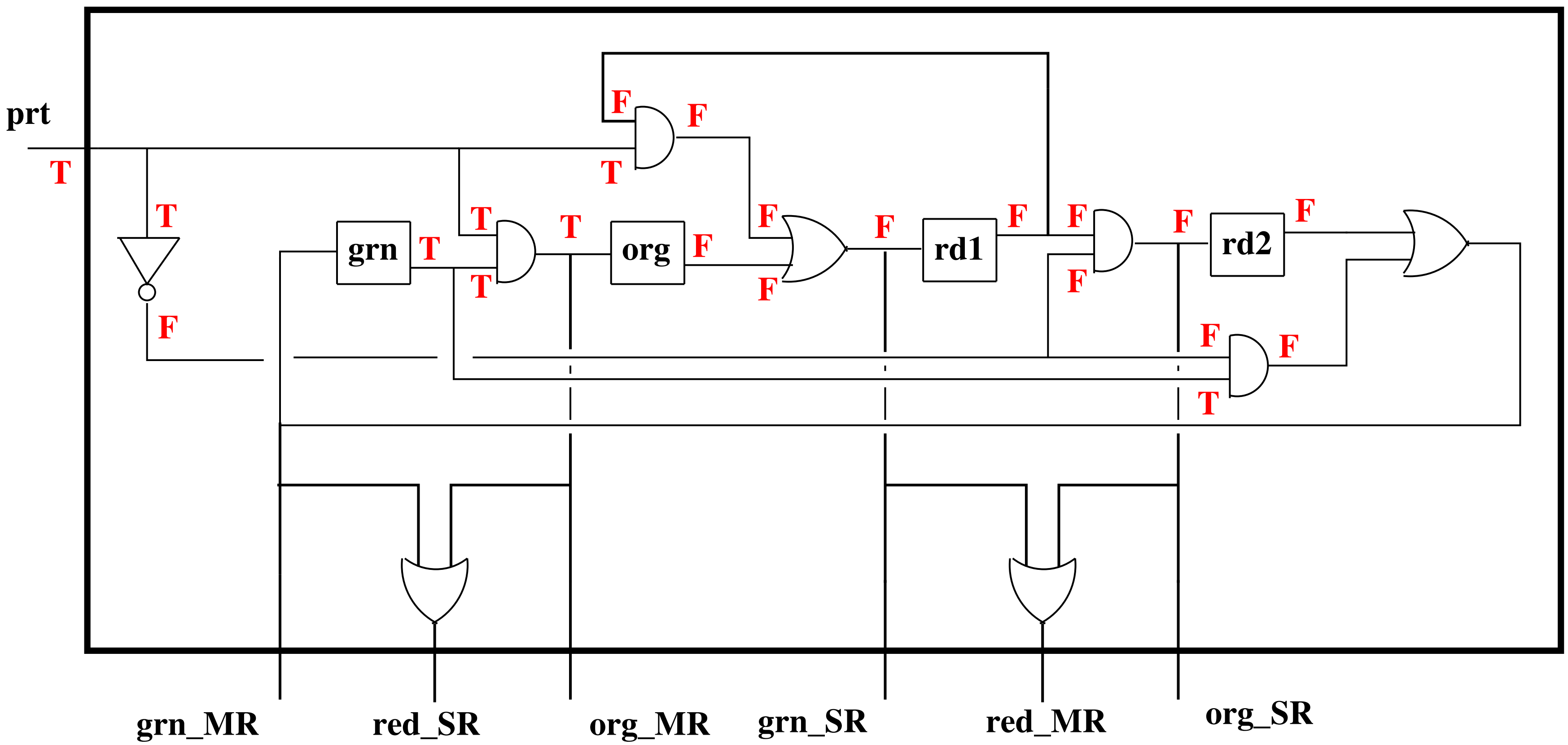


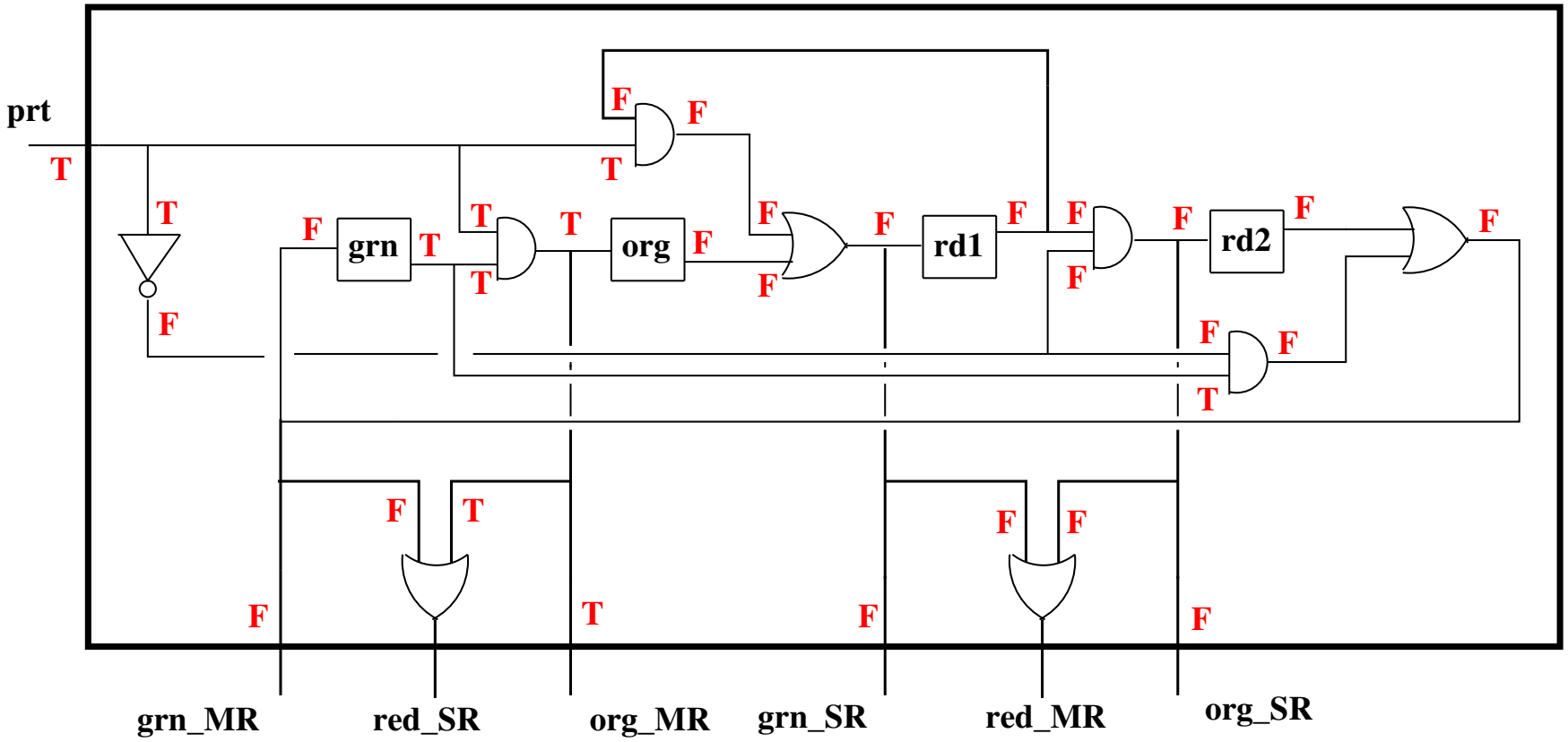

prt
T
grn
T
org
F
rd1
F
rd2
F
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR

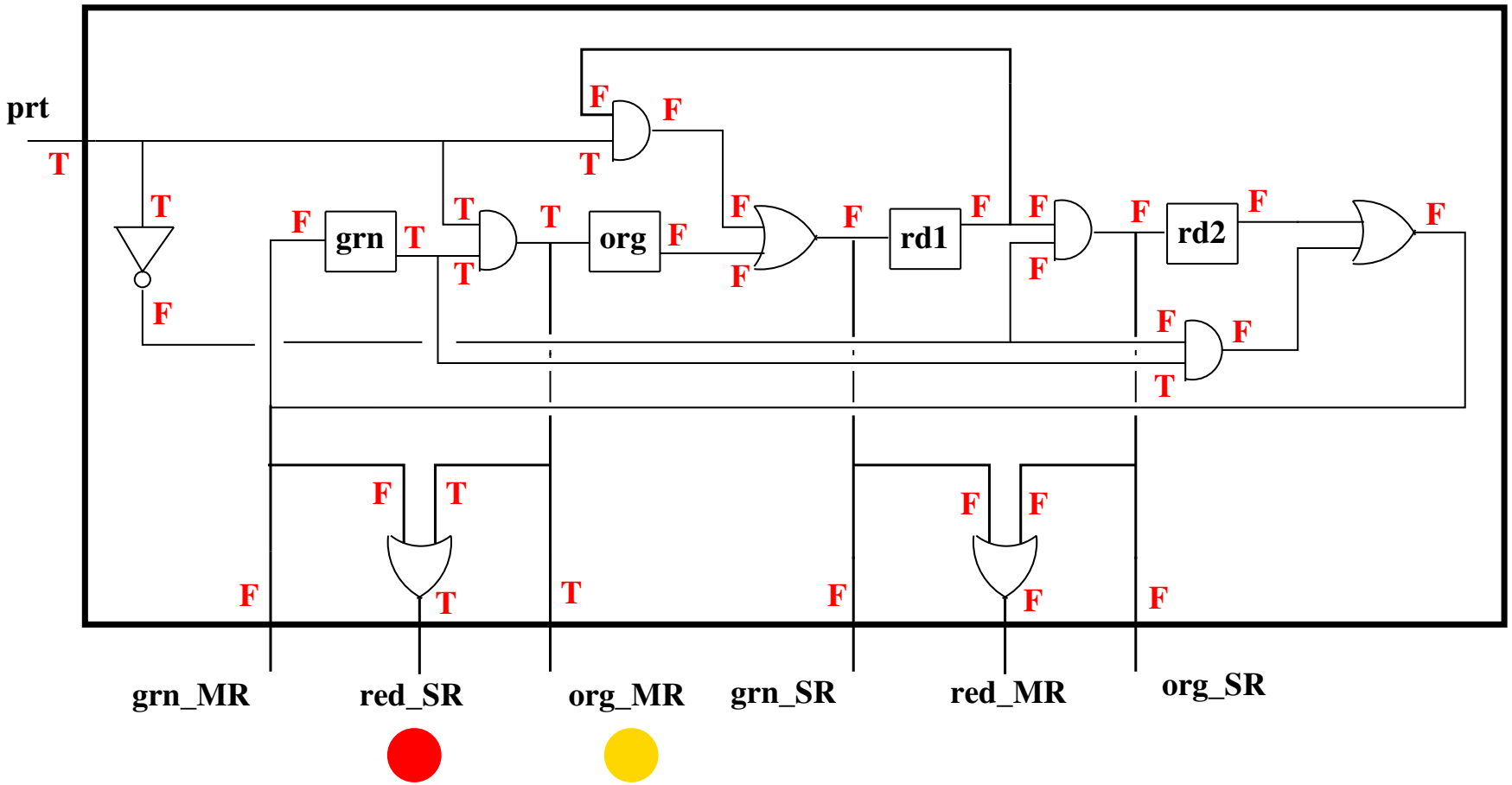

prt
T
T
grn
T
T
T
org
F
F
F
T
F
rd1
F
F
rd2
F
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR

Computation

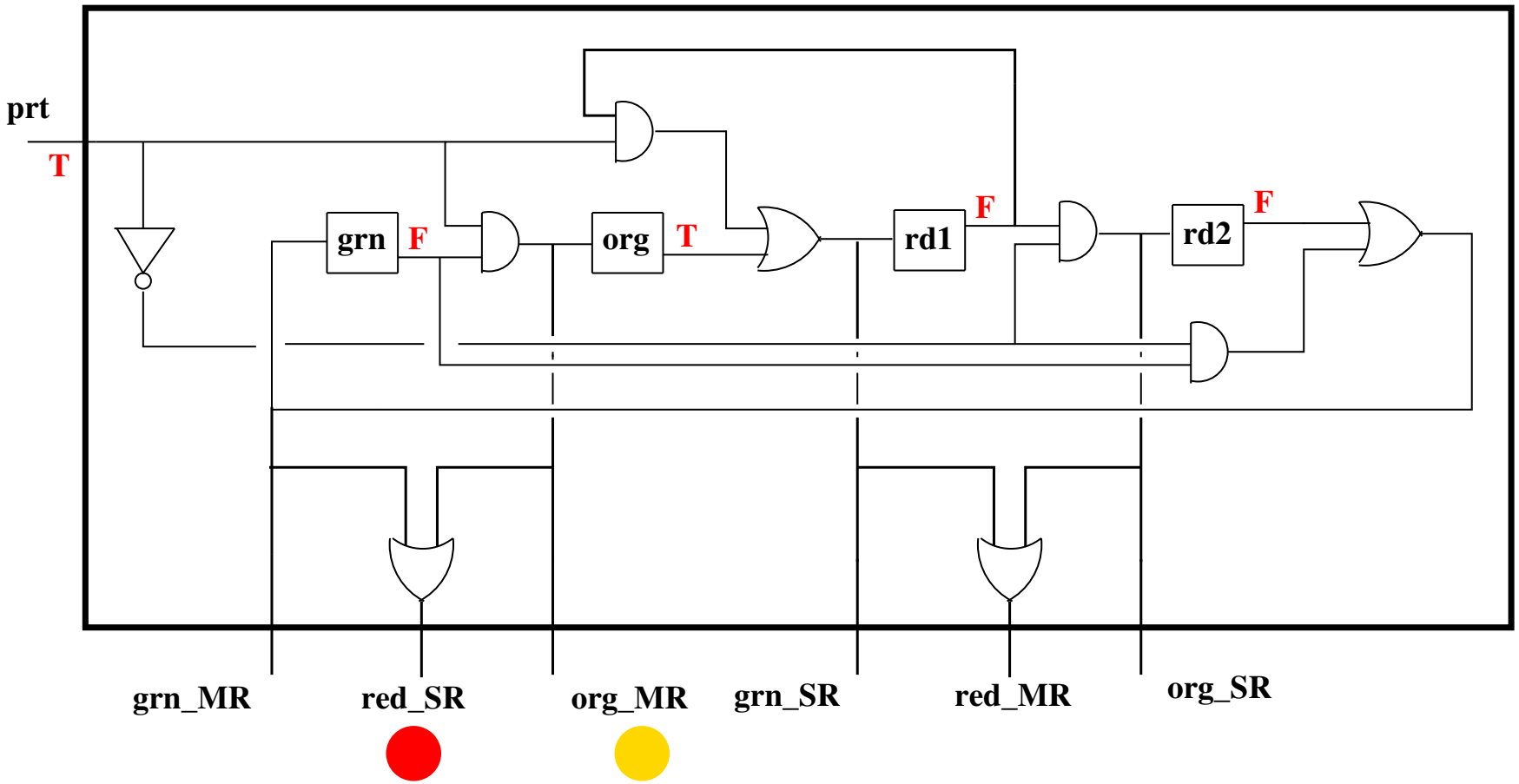


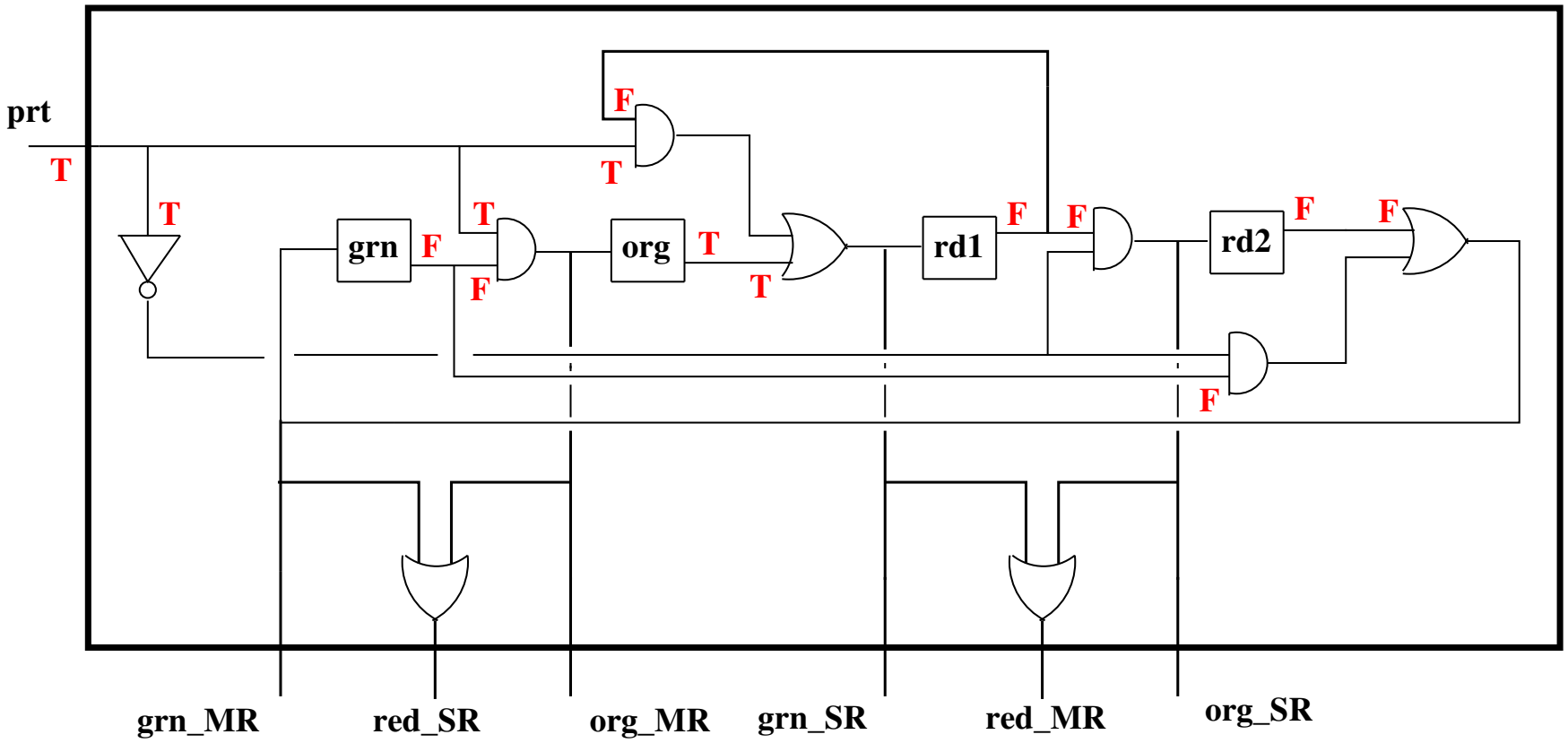

prt
T
T
F
grn
T
T
T
T
org
F
F
T
F
F
F
F
rd1
F
F
F
F
rd2
F
F
F
T
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR

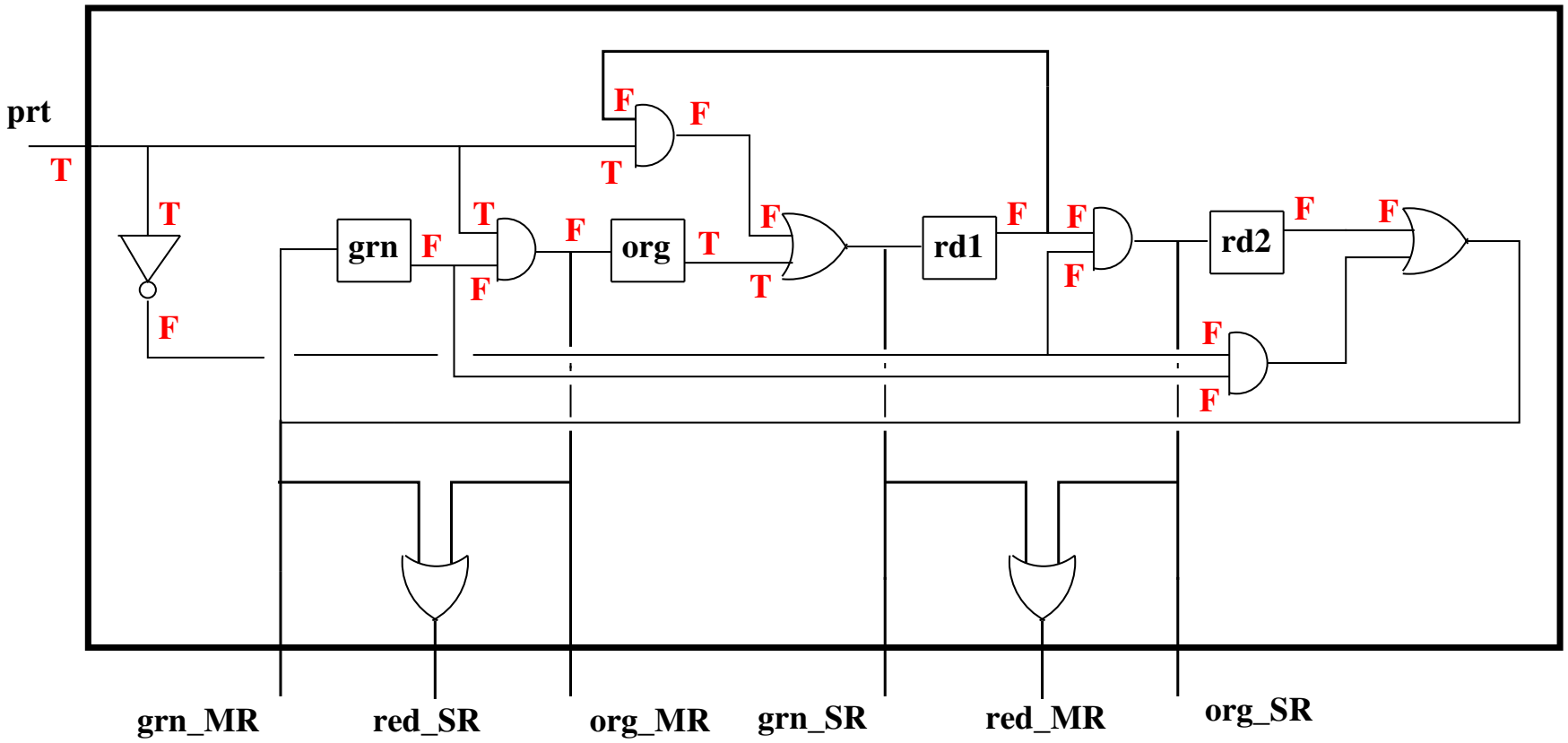

prt
T
F
T
F
grn
T
T
T
T
org
F
F
F
rd1
F
F
F
F
rd2
F
F
F
F
F
T
F
T
F
T
F
F
F
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR

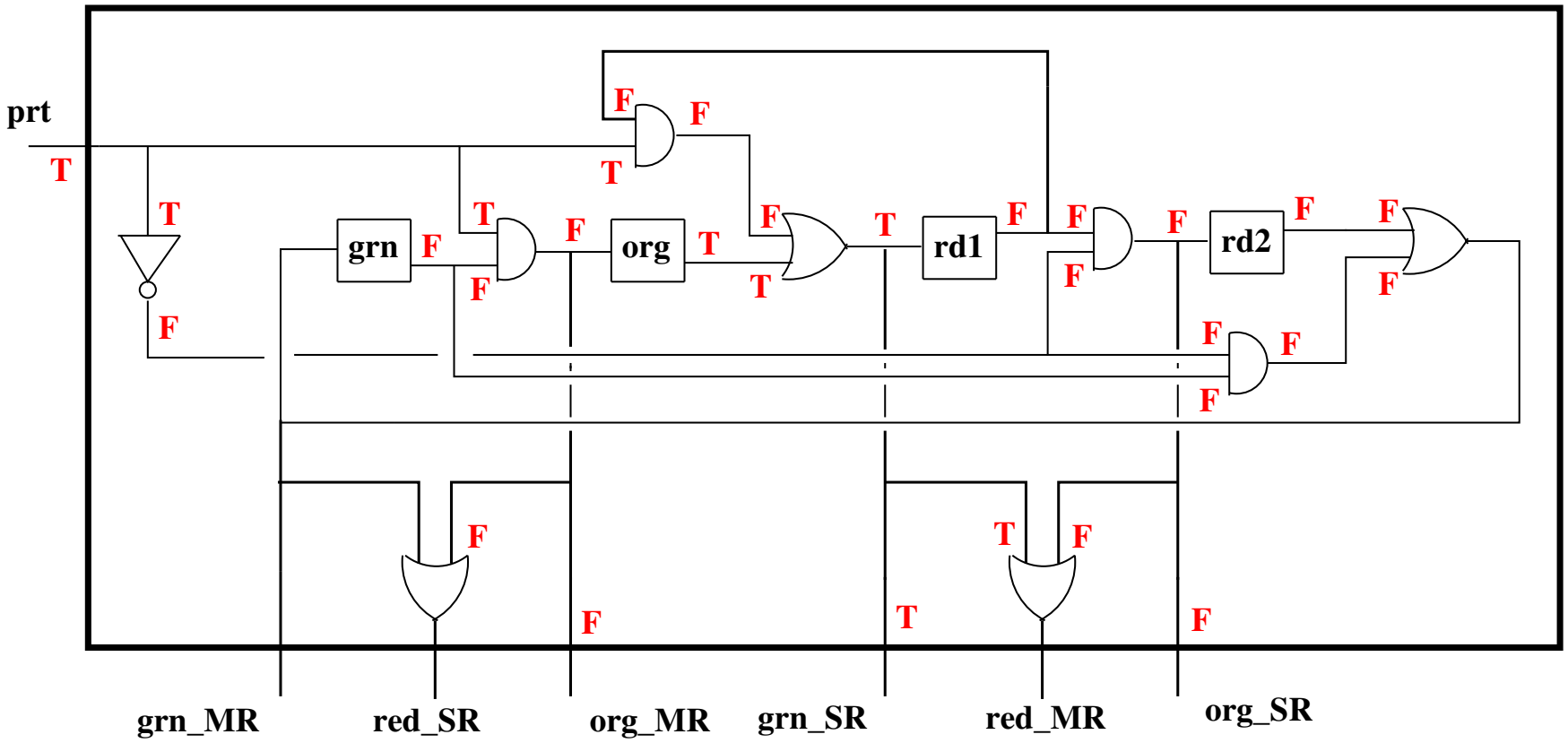

prt
grn
org
rd1
rd2
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR

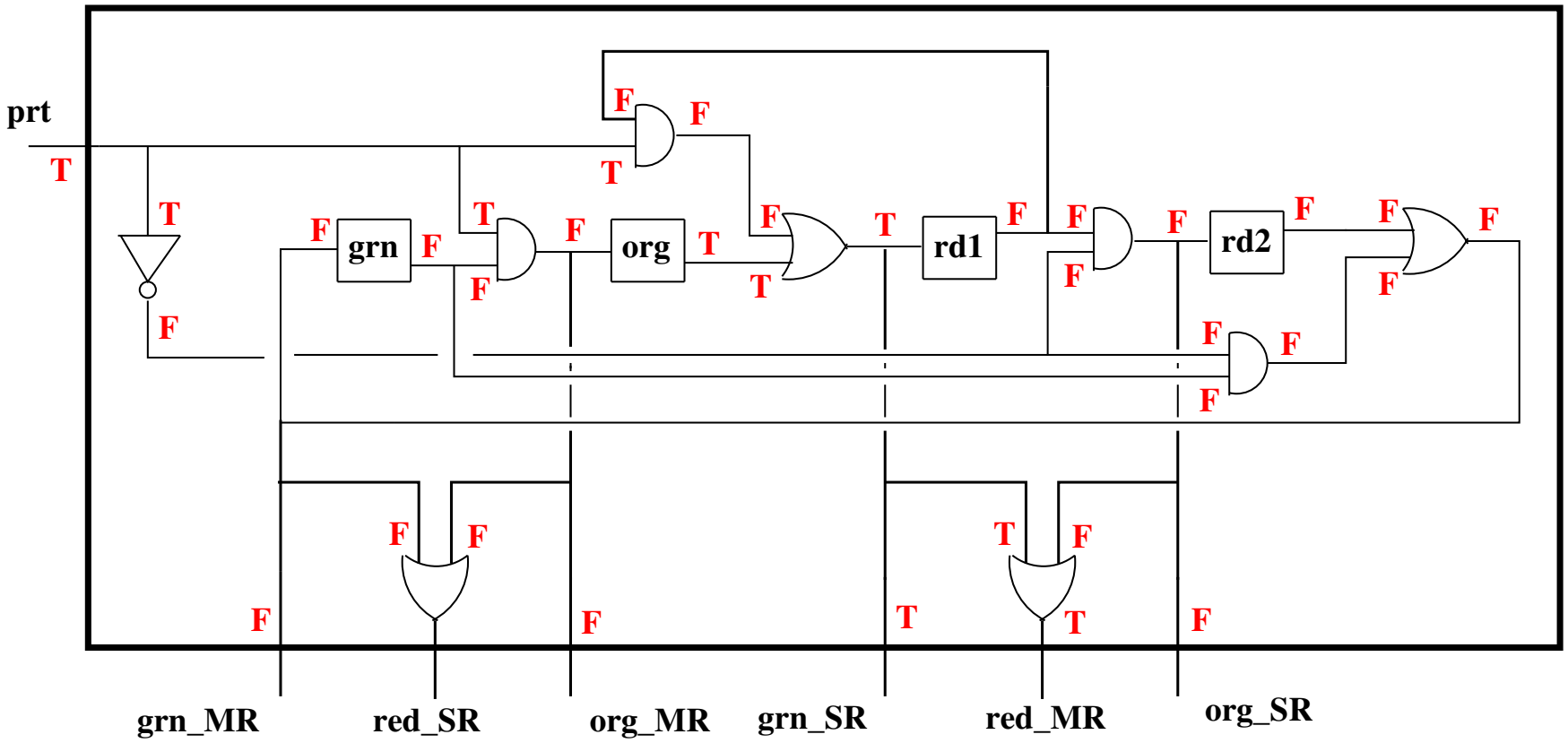
Another Computation



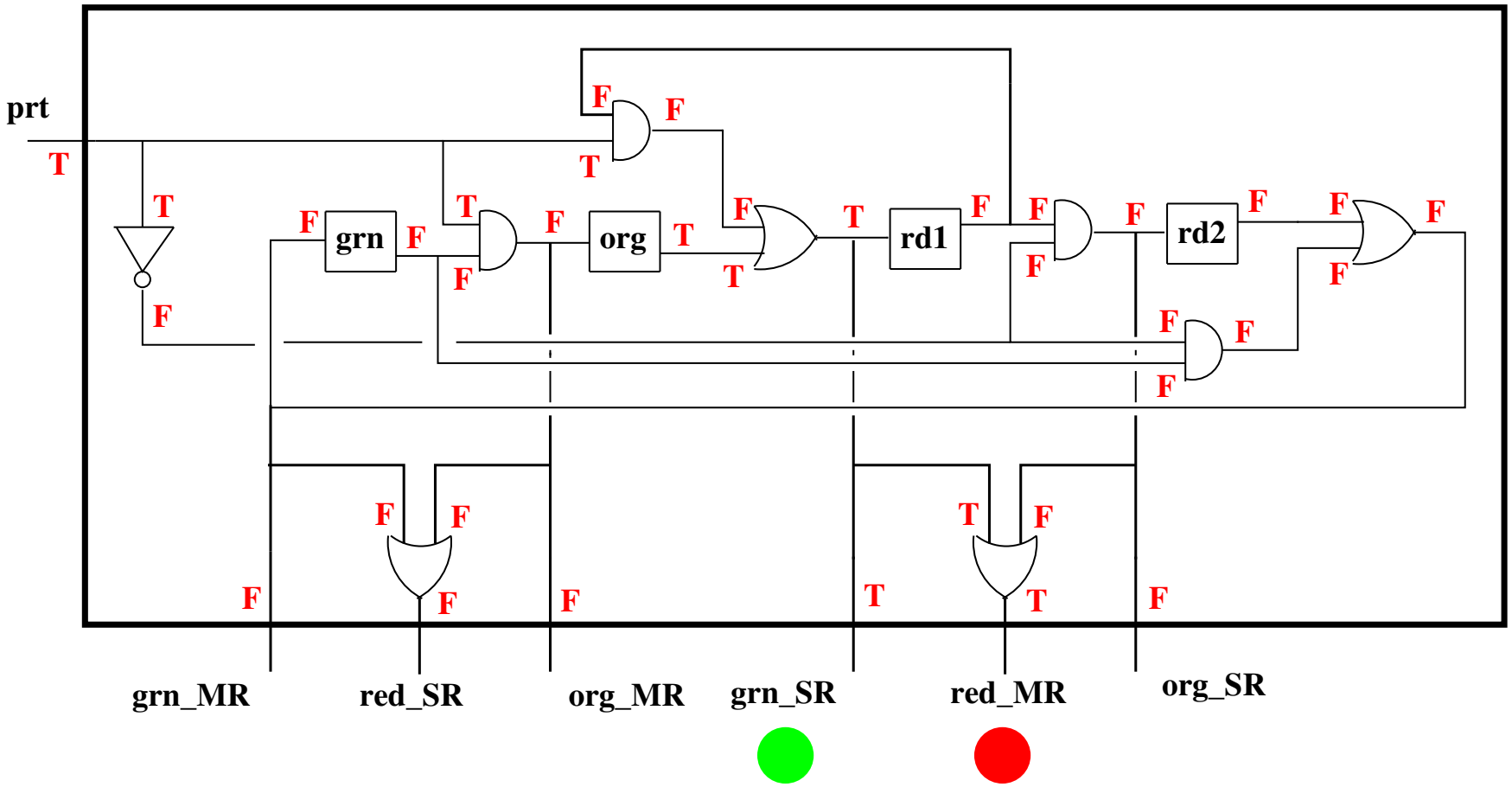

prt
T
T
grn
F
F
T
F
org
T
T
F
T
rd1
F
F
rd2
F
F
F
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR


prt
T
T
F
grn
F
T
F
F
org
T
T
F
F
T
F
rd1
F
F
F
rd2
F
F
F
F
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR

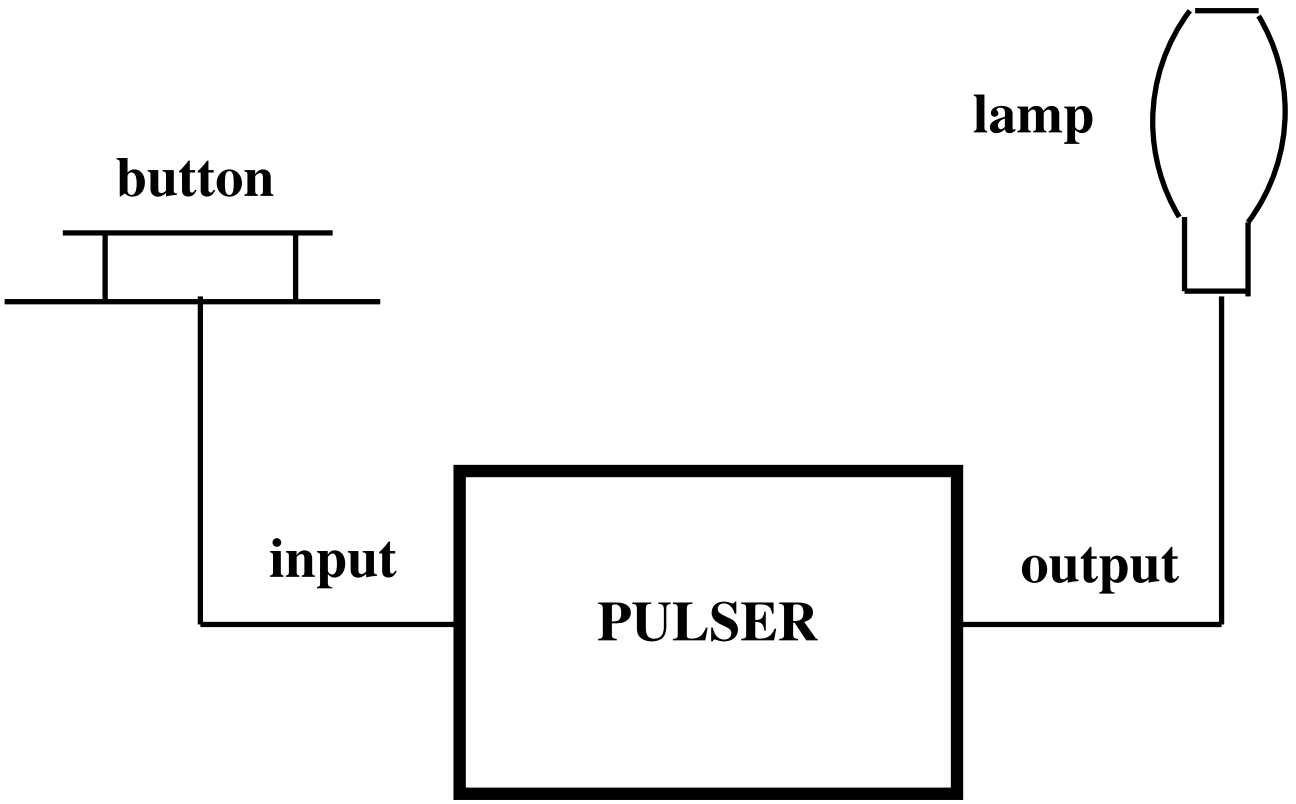

prt
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
T
F
grn
org
rd1
rd2
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR


prt
T
F
T
F
grn
F
T
F
F
org
T
F
T
T
rd1
F
F
F
F
rd2
F
F
F
F
F
F
F
F
F
F
F
T
F
T
T
F
F
F
grn_MR
red_SR
org_MR
grn_SR
red_MR
org_SR

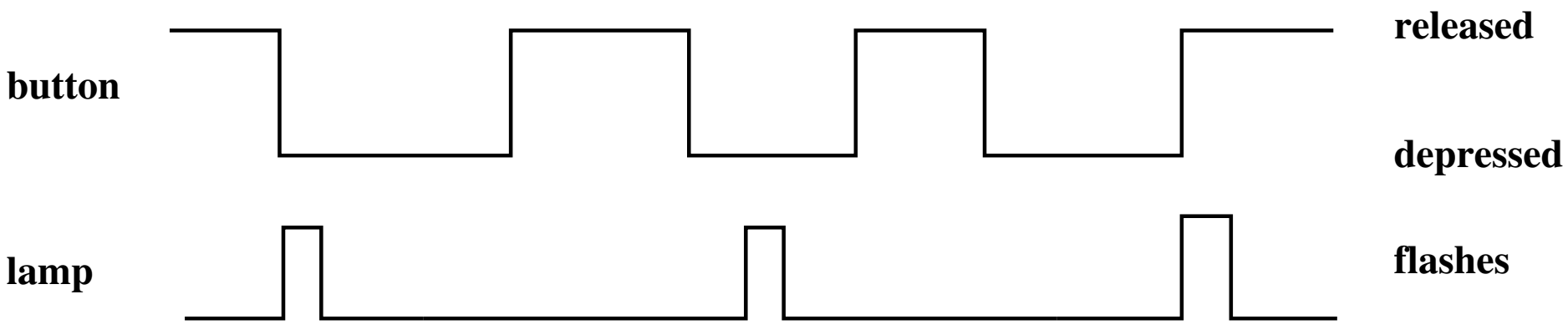
Computation



Yet Another Example: the Pulser



- The lamp must flash once between button depression and subsequent release.



- The flash may occur very early: just after button depression

- or in between button depression and subsequent release

- or very late: just after button release

- We have a certain non-determinacy

Initial Model: Carrier Set and Constants

carrier set: $MODE$

constants: env, cir

prp0_1: $env \in MODE$

prp0_2: $cir \in MODE$

prp0_3: $env \neq cir$

prp0_4: $MODE = \{env, cir\}$

Initial Model: the State

- $push$: number of times the button is depressed

- pop: number of times the button is released

variables: $mode, push, pop$

inv0_1: $mode \in MODE$

inv0_2: $push \in \mathbb{N}$

inv0_3: $pop \in \mathbb{N}$

inv0_4: $pop \leq push$

inv0_5: $push \leq pop + 1$

In other words:

$$push \in \{pop, pop + 1\}$$

Initial Model: the State (cont'd)

- $flash$: number of times the lamp flashes

variables: $\ldots, flash$

inv0_6: $flash \in \mathbb{N}$

inv0_7: $flash \leq push$

inv0_8: $push \leq flash + 1$

- In other words:

$$push \in \{flash, flash + 1\}$$

Initial Model Events: Environment

env1
when
$mode = env$
$pop = push$
then
$mode := cir$
$push := push + 1$
end

env2
when
$mode = env$
$pop \neq push$
then
$mode := cir$
$pop := pop + 1$
end

env3
when
$mode = env$
then
$mode := cir$
end

init
$mode := env$
$push := 0$
$pop := 0$
$flash := 0$

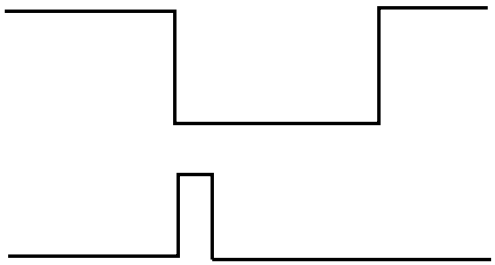
Initial Model Events: the Circuit

cir1
when
$mode = cir$
$push \neq flash$
then
$mode := env$
$flash := flash + 1$
end

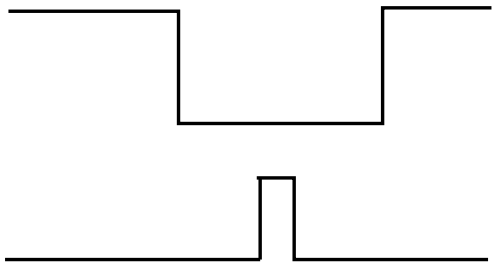
- Button has been depressed and flash not done yet

- Notice that button might have been just released ($push = pop$)

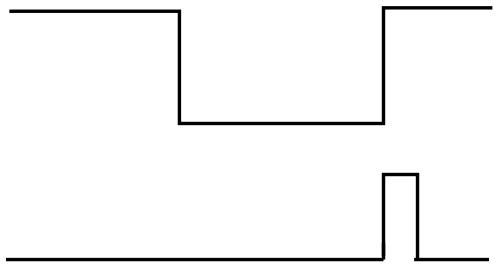
Analysing the "do-nothing" Event of the circuit



Case 1

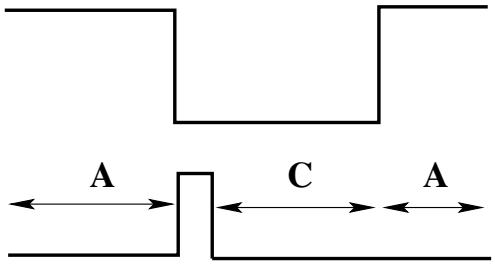


Case 2

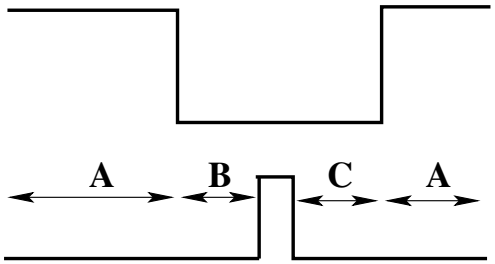


Case 1

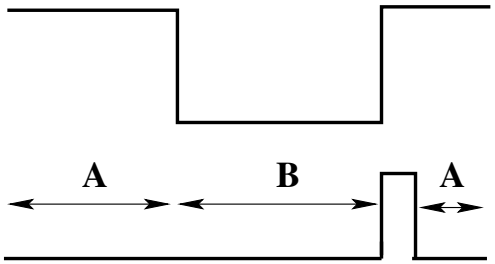
Analysing the "do-nothing" Event of the circuit



Case 1

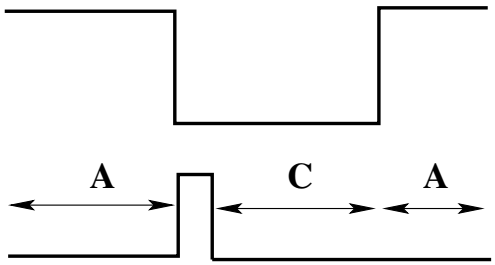


Case 2

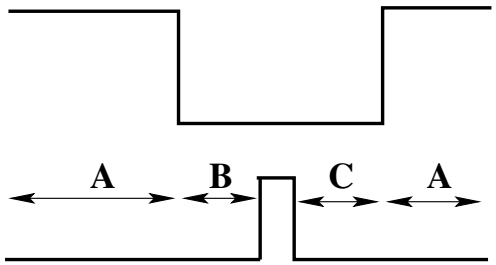


Case 3

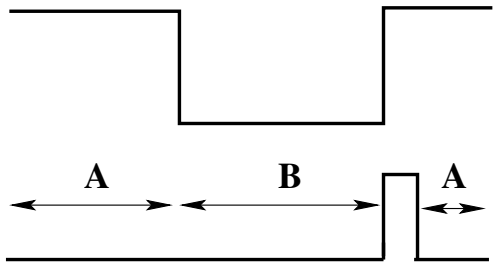
Analysing the "do-nothing" Event of the circuit



Case 1



Case 2



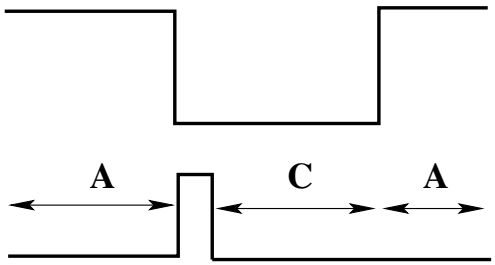
Case 3

Condition A: $push = pop \quad \wedge \quad push = flash$

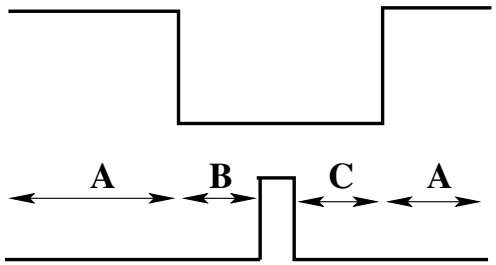
Condition B: $push \neq pop \quad \wedge \quad push \neq flash$

Condition C: $push \neq pop \quad \wedge \quad push = flash$

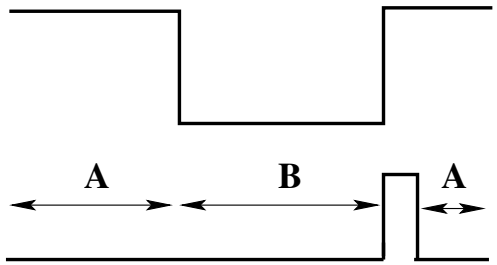
Analysing the "do-nothing" Event of the circuit



Case 1



Case 2



Case 3

Condition A: $push = pop \ \wedge \ push = flash$

Condition B: $push \neq pop \ \wedge \ push \neq flash$

Condition C: $push \neq pop \ \wedge \ push = flash$

- Guard of "do-nothing":

$$A \ \vee \ B \ \vee \ C \ \Leftrightarrow \ push \neq pop \ \vee \ push = flash$$

The "do-nothing" Event of the circuit

cir2

when

$mode = cir$

$push \neq pop \ \vee \ push = flash$

then

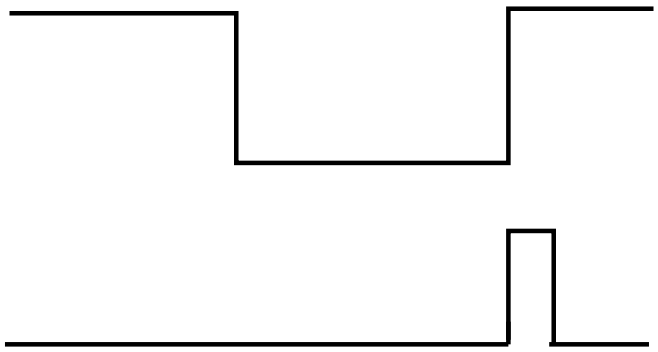
$mode := env$

end

- The following invariant has to be added:

inv0_9: $mode = env \Rightarrow flash = push \vee flash = pop$

- Notice that $flash = push \vee flash = pop$ does not hold always:



- Here we have: $pop = push = flash + 1$

First Refinement: Removing Non-determinacy

cir1
when
$mode = cir$
$push \neq flash$
then
$mode := env$
$flash := flash + 1$
end

cir2
when
$mode = cir$
$push \neq pop \ \vee \ push = flash$
then
$mode := env$
end

- The guard are both true when $push \neq flash$ and $push \neq pop$ both hold
- When the flash has not yet occured ($push \neq flash$)
- When we are in between a depression and release ($push \neq pop$)
- We can either flash (cir1) or do nothing (cir2)

Removing Non-determinacy

cir1
when
$mode = cir$
$push \neq flash$
then
$mode := env$
$flash := flash + 1$
end

cir2
when
$mode = cir$
$push \neq pop \ \vee \ push = flash$
then
$mode := env$
end

- Two solutions:
  - removing $push \neq pop$ in cir2
  - adding $push = pop$ in cir1
- First solution: flashing as early as possible
- Second solution: flashing as late as possible

cir1_PULSER1
when
$mode = cir$
$push \neq flash$
then
$mode := env$
$flash := flash + 1$
end

cir2_PULSER1
when
$mode = cir$
$push = flash$
then
$mode := env$
end

- Early flash: as soon as $push$ is different from $flash$

(just after depressing the button)

cir1_PULSER2
when
$mode = cir$
$push = pop \ \wedge \ push \neq flash$
then
$mode := env$
$flash := flash + 1$
end

cir2_PULSER2
when
$mode = cir$
$push \neq pop \ \vee \ push = flash$
then
$mode := env$
end

- Late flash: both $push$ and pop are the same (button release has occurred) but the $flash$ has not yet occured

- Introducing boolean Input and Output

variables: $mode, input, output$

inv2_1: $input \in \mathrm{BOOL}$

inv2_2: $output \in \mathrm{BOOL}$

Second Refinement (cont'd)

(abstract-)env1
when
$mode = env$
$pop = push$
then
$mode := cir$
$push := push + 1$
end

(abstract-)env2
when
$mode = env$
$pop \neq push$
then
$mode := cir$
$pop := pop + 1$
end

(abstract-)env3
when
$mode = env$
then
$mode := cir$
end

env1
when
$mode = env$
$input = \text{FALSE}$
then
$mode := cir$
$input := \text{TRUE}$
end

env2
when
$mode = env$
$input = \text{TRUE}$
then
$mode := cir$
$input := \text{FALSE}$
end

env3
when
$mode = env$
then
$mode := cir$
end

inv2_3: $input = \text{TRUE} \Leftrightarrow pop \neq push$

Implementing PULSER1

(abstract-)cir1_PULSER1
when
$mode = cir$
$push \neq flash$
then
$mode := env$
$flash := flash + 1$
end

(abstract-)cir2_PULSER1
when
$mode = cir$
$push = flash$
then
$mode := env$
end

variables: $\ldots, reg$

inv2_4: $reg \in \mathrm{BOOL}$

invi2_5: $mode = env \Rightarrow reg = input$

cir1_PULSER1
when
$mode = cir$
$input = \mathrm{TRUE} \wedge reg = \mathrm{FALSE}$
/* push just changed */
then
$mode := env$
$output := \mathrm{TRUE}$
$reg := input$
end

cir2_PULSER1
when
$mode = cir$
$input = \mathrm{FALSE} \vee reg = \mathrm{TRUE}$
then
$mode := env$
$output := \mathrm{FALSE}$
$reg := input$
end

Implementing PULSER1

(abstract-)cir1_PULSER1
when
$mode = cir$
$push \neq flash$
then
$mode := env$
$flash := flash + 1$
end

(abstract-)cir2_PULSER1
when
$mode = cir$
$push = flash$
then
$mode := env$
end

cir1_PULSER1
when
$mode = cir$
$input = \mathrm{TRUE} \ \wedge \ reg = \mathrm{FALSE}$
then
$mode := env$
$output := \mathrm{TRUE}$
$reg := input$
end

cir2_PULSER1
when
$mode = cir$
$input = \mathrm{FALSE} \ \vee \ reg = \mathrm{TRUE}$
then
$mode := env$
$output := \mathrm{FALSE}$
$reg := input$
end

inv2_PULSER1_6: $mode = cir \ \Rightarrow \left(\begin{array}{l} input = \mathrm{TRUE} \ \wedge \ reg = \mathrm{FALSE} \\ \Leftrightarrow \\ push \neq flash \end{array} \right)$

Implementing PULSER2

(abstract-)cir1_PULSER2
when
$mode = cir$
$push = pop \ \wedge \ push \neq flash$
then
$mode := env$
$flash := flash + 1$
end

(abstract-)cir2_PULSER2
when
$mode = cir$
$push \neq pop \ \vee \ push = flash$
then
$mode := env$
end

cir1_PULSER2
when
$mode = cir$
$input = \mathrm{FALSE} \ \wedge \ reg = \mathrm{TRUE}$
/* pop just changed */
then
$mode := env$
$output := \mathrm{TRUE}$
$reg := input$
end

cir2_PULSER2
when
$mode = cir$
$input = \mathrm{TRUE} \ \vee \ reg = \mathrm{FALSE}$
then
$mode := env$
$output := \mathrm{FALSE}$
$reg := input$
end

inv2_PULSER2_6: $mode = cir \ \Rightarrow \left(\begin{array}{l} input = \mathrm{FALSE} \ \wedge \ reg = \mathrm{TRUE} \\ \Leftrightarrow \\ push \neq flash \ \wedge push = pop \end{array} \right)$

Final Conditions

- the circuit variables must all be boolean,

- the inputs must be boolean,

- the outputs must be boolean,

- the circuit must be deadlock free,

Final Conditions (cont'd)

- the circuit must be internally deterministic (variable assignments),

- the circuit must be externally deterministic (guards),

- the environment does not access the circuit variables,

- the circuit does not access the environment variables.

A Typical Final Circuit Event

cir_event_i

when

$mode = cir$

$GC\_i(input, c)$

then

$mode := env$

$c := C\_i(input, c)$

$output := O\_i(input, c)$

end

Final Conditions

- Deadlock Freeness: Disjunction of guards

$$mode = cir \;\Rightarrow\; GC\_1(input, c) \;\vee\; \ldots \;\vee\; GC\_n(input, c)$$

- External Determinacy: Mutual exclusion of guards

$$mode = cir \;\Rightarrow\; \neg\,(\,GC\_i(input, c) \;\wedge\; GC\_j(input, c)\,)$$

Merging of Circuit Events (implicit refinement)

cir_event_i
when
$mode = cir$
$GC\_i(input, c)$
then
$mode := env$

$$c := \text{bool}\left(\begin{array}{l} \ldots \\ GC\_i(input, c) \ \wedge \ C\_i(input, c) = \text{TRUE} \ \vee \\ \ldots \end{array}\right)$$

$$output := \text{bool}\left(\begin{array}{l} \ldots \\ GC\_i(input, c) \ \wedge \ O\_i(input, c) = \text{TRUE} \ \vee \\ \ldots \end{array}\right)$$

end

The bool operator: transforming a predicate into a boolean expression

$$E = \text{bool}(P) \quad \Leftrightarrow \quad \left(\begin{array}{lcl} P & \Rightarrow & E = \text{TRUE} \\ \neg\, P & \Rightarrow & E = \text{FALSE} \end{array}\right)$$

$$
\begin{array}{l}
GC\_i(input, c) \\
\Rightarrow \\
C\_i(input, c) \;=\; \mathbf{bool}\left(\begin{array}{l} \ldots \;\vee \\ GC\_i(input, c) \;\wedge\; C\_i(input, c) = \mathbf{TRUE} \;\vee \\ \ldots \end{array}\right)
\end{array}
$$

- According to the definition of $\mathbf{bool}$, this reduces to the following two statements:

$$
\begin{array}{l}
GC\_i(input, c) \\
\left(\begin{array}{l} \ldots \;\vee \\ GC\_i(input, c) \;\wedge\; C\_i(input, c) = \mathbf{TRUE} \;\vee \\ \ldots \end{array}\right) \\
\Rightarrow \\
C\_i(input, c) = \mathbf{TRUE}
\end{array}
$$

$$
\begin{array}{l}
GC\_i(input, c) \\
\neg\left(\begin{array}{l} \ldots \;\vee \\ GC\_i(input, c) \;\wedge\; C\_i(input, c) = \mathbf{TRUE} \;\vee \\ \ldots \end{array}\right) \\
\Rightarrow \\
C\_i(input, c) = \mathbf{FALSE}
\end{array}
$$

Proving the First Statement

$$
\begin{array}{l}
GC\_i(input, c) \\
\left(\begin{array}{l}
\ldots \ \vee \\
GC\_i(input, c) \ \wedge \ C\_i(input, c) = \mathbf{TRUE} \ \vee \\
\ldots
\end{array} \right) \\
\Rightarrow \\
C\_i(input, c) = \mathbf{TRUE}
\end{array}
$$

- Since the guard are mutually exclusive, it can be reduced to:

$$
\begin{array}{l}
GC\_i(input, c) \\
C\_i(input, c) = \mathbf{TRUE} \\
\Rightarrow \\
C\_i(input, c) = \mathbf{TRUE}
\end{array}
$$

Proving the Second Statement

$$
\begin{array}{l}
GC\_i(input, c) \\
\neg \left(\begin{array}{l} \ldots \ \vee \\ GC\_i(input, c) \ \wedge \ C\_i(input, c) = \mathbf{TRUE} \ \vee \\ \ldots \end{array} \right) \\
\Rightarrow \\
C\_i(input, c) = \mathbf{FALSE}
\end{array}
$$

- Equivalently (and trivially discharged)

$$
\begin{array}{l}
GC\_i(input, c) \\
\ldots \\
(\neg GC\_i(input, c) \ \vee \ C\_i(input, c) = \mathbf{FALSE}) \\
\ldots \\
\Rightarrow \\
C\_i(input, c) = \mathbf{FALSE}
\end{array}
$$

Final Circuit

- The circuit events are deadlock free (disjunction of guards holds)

- They have identical actions,

- They can all be merged into a single event as follows:

circuit_event
  when
    $mode = cir$
  then
    $mode := env$

$$c := \mathrm{bool}\left(\begin{array}{l}\ldots \ \vee \\ GC\_i(input, c) \ \wedge \ C\_i(input, c) = \mathrm{TRUE} \ \vee \\ \ldots\end{array}\right)$$

$$output := \mathrm{bool}\left(\begin{array}{l}\ldots \ \vee \\ GC\_i(input, c) \ \wedge \ O\_i(input, c) = \mathrm{TRUE} \ \vee \\ \ldots\end{array}\right)$$

  end

- When $C_i(input, c)$ is syntactically equal to $\mathbf{TRUE}$ then $C_i(input, c) = \mathbf{TRUE}$ can be removed,

circuit_event
when
$mode = cir$
then
$mode := env$

$$c := \text{bool}\left(\begin{array}{l}\ldots \ \vee \\ GC\_i(input, c) \wedge \mathbf{TRUE} = \mathbf{TRUE} \ \vee \\ \ldots\end{array}\right)$$

$$output := \text{bool}\left(\begin{array}{l}\ldots \ \vee \\ GC\_i(input, c) \ \wedge \ O\_i(input, c) = \mathbf{TRUE} \ \vee \\ \ldots\end{array}\right)$$

end

- Same for $O\_i(input, c)$

Further Simplifications (cont'd)

- When $C_i(input, c)$ is syntactically equal to $\mathbf{FALSE}$ then $GC\_i(input, c) \wedge C_i(input, c) = \mathbf{TRUE}$ can be removed.

circuit_event
  when
    $mode = cir$
  then
    $mode := env$
    $c := \text{bool}\left(\begin{array}{l}\ldots \vee \\ GC\_i(input, c) \wedge \mathbf{FALSE} = \mathbf{TRUE} \vee \\ \ldots\end{array}\right)$
    $output := \text{bool}\left(\begin{array}{l}\ldots \vee \\ GC\_i(input, c) \wedge O\_i(input, c) = \mathbf{TRUE} \vee \\ \ldots\end{array}\right)$
  end

- Same for $O\_i(input, c)$

Final Circuit: PULSER1

cir1_PULSER1
when
$mode = cir$
$input = \text{TRUE} \ \wedge \ reg = \text{FALSE}$
then
$mode := env$
$output := \text{TRUE}$
$reg := input$
end

cir2_PULSER1
when
$mode = cir$
$input = \text{FALSE} \ \vee \ reg = \text{TRUE}$
then
$mode := env$
$output := \text{FALSE}$
$reg := input$
end

PULSER1
when
$mode = cir$
then
$mode := env$
$output := \text{bool}(input = \text{TRUE} \ \wedge \ reg = \text{FALSE})$
$reg := \text{bool}(input = \text{TRUE} \ \wedge \ (input = \text{TRUE} \ \wedge \ reg = \text{FALSE}) \ \vee$
$input = \text{TRUE} \ \wedge \ (input = \text{FALSE} \ \vee \ reg = \text{TRUE}))$
end

PULSER1

when

$mode = cir$

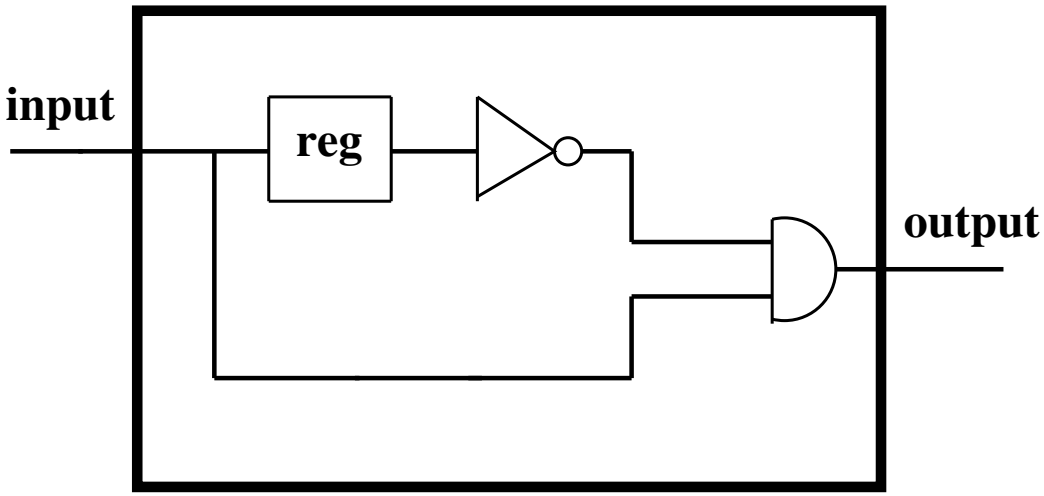
then

$mode := env$

$output := \text{bool}(input = \text{TRUE} \ \wedge \ reg = \text{FALSE})$

$reg := \text{bool}(input = \text{TRUE})$

end



Final Circuit: PULSER2

cir1_PULSER2
when
$mode = cir$
$input = \text{FALSE} \ \wedge \ reg = \text{TRUE}$
then
$mode := env$
$output := \text{TRUE}$
$reg := input$
end

cir2_PULSER2
when
$mode = cir$
$input = \text{TRUE} \ \vee \ reg = \text{FALSE}$
then
$mode := env$
$output := \text{FALSE}$
$reg := input$
end

PULSER2
when
$mode = cir$
then
$mode := env$
$output := \text{bool}(input = \text{FALSE} \ \wedge \ reg = \text{TRUE})$
$reg := \text{bool}(input = \text{TRUE} \ \wedge \ (input = \text{FALSE} \ \wedge \ reg = \text{TRUE}) \ \vee$
$input = \text{TRUE} \ \wedge \ (input = \text{TRUE} \ \vee \ reg = \text{FALSE}))$
end

PULSER2

when

$mode = cir$

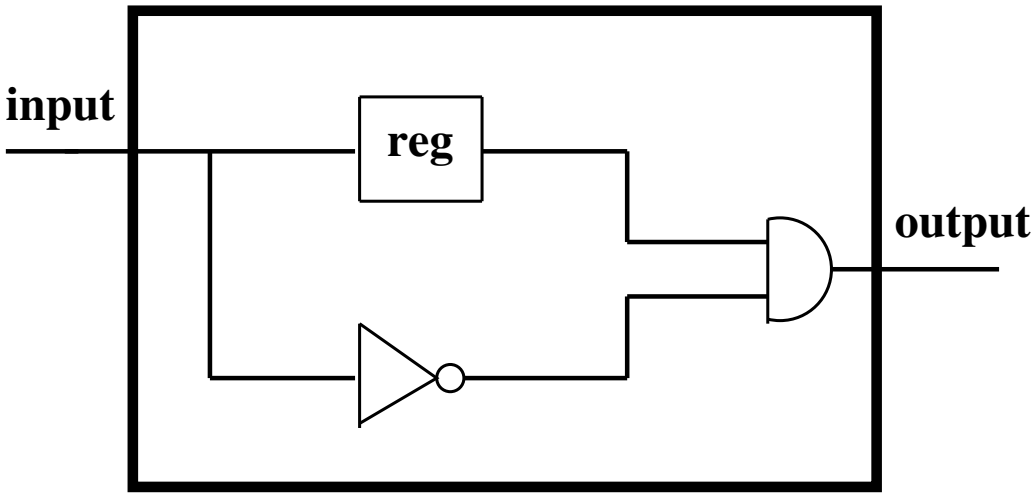
then

$mode := env$

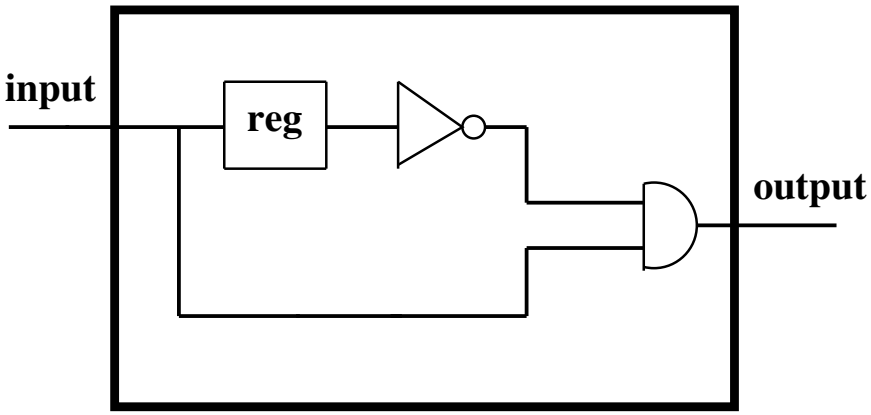
$output := \mathrm{bool}(input = \mathrm{FALSE} \ \wedge \ reg = \mathrm{TRUE})$

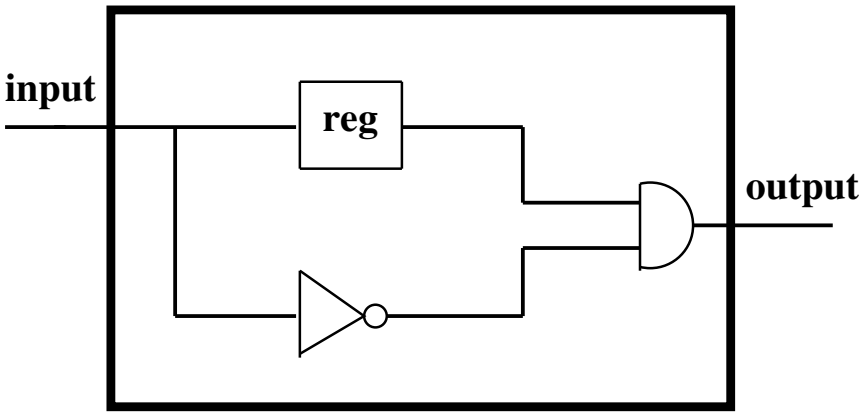
$reg \quad := \mathrm{bool}(input = \mathrm{TRUE})$

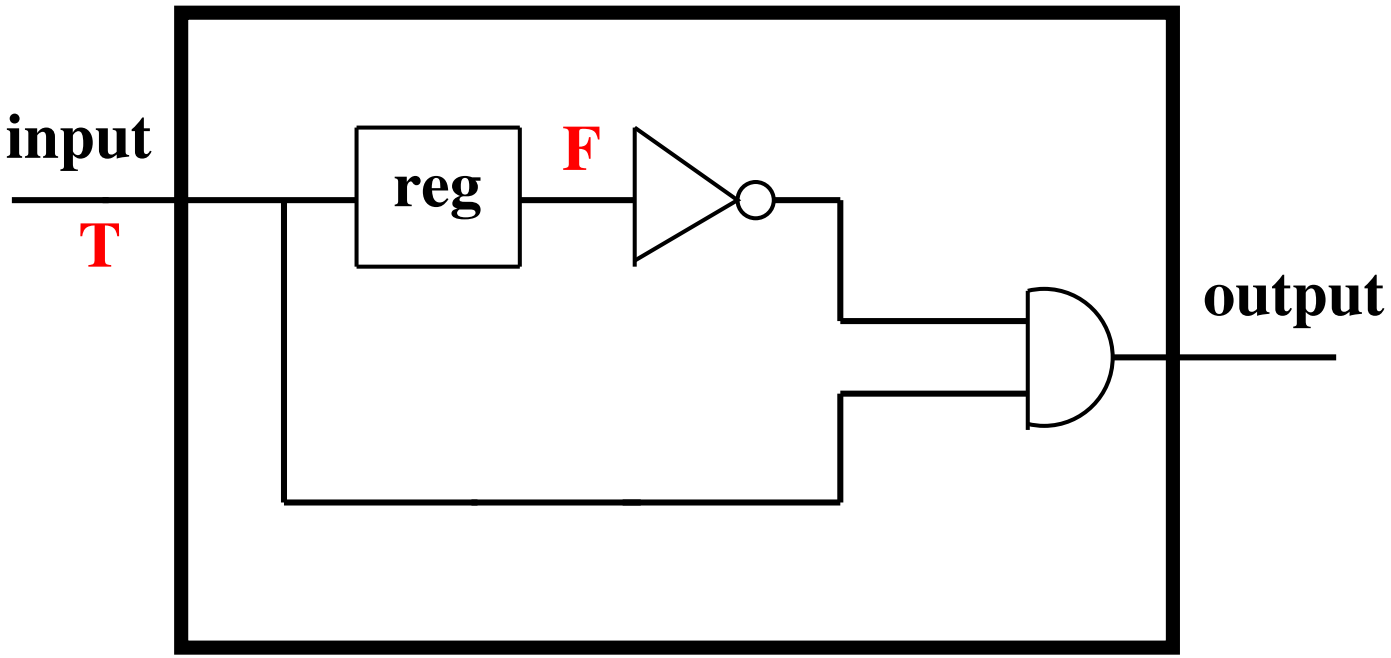
end

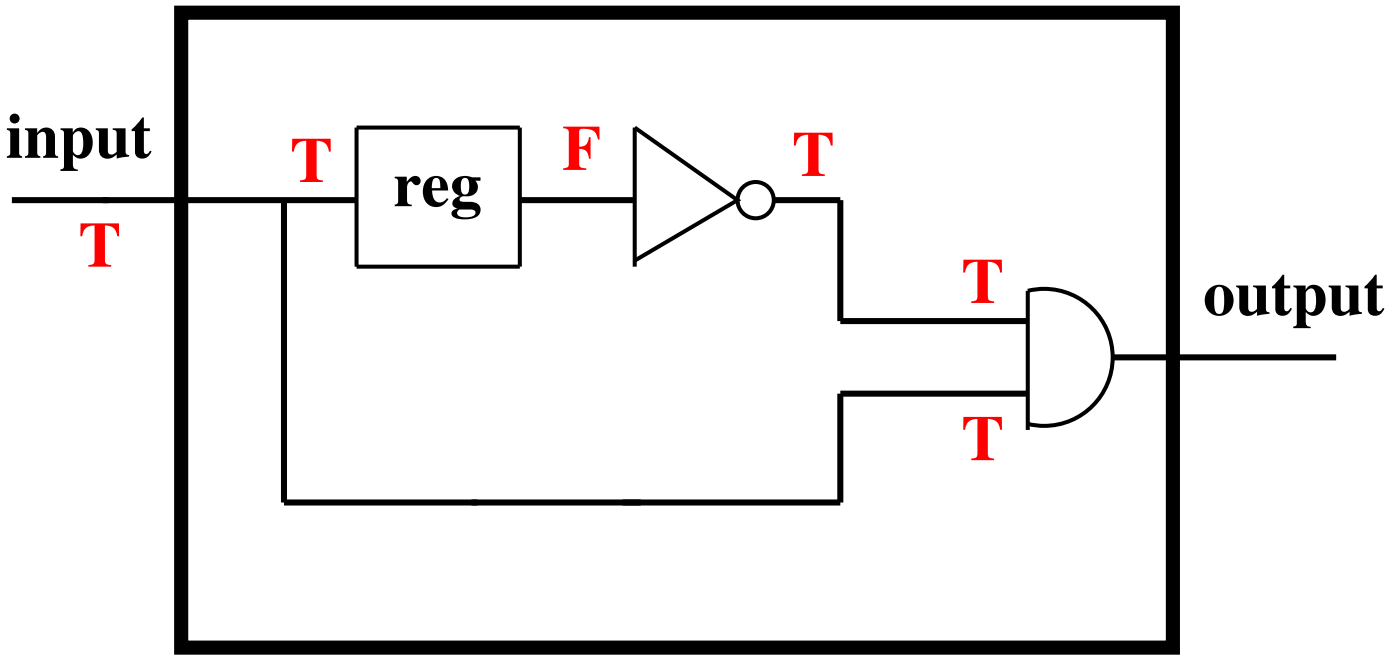


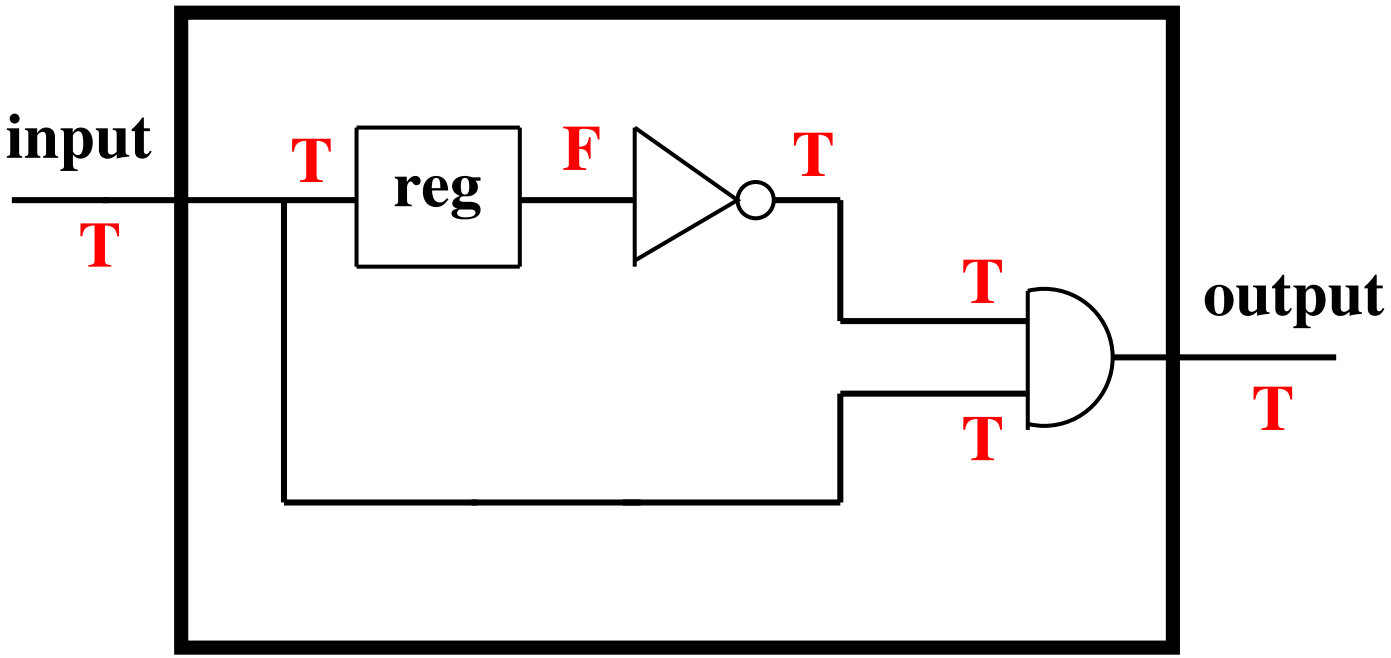
Both Circuits

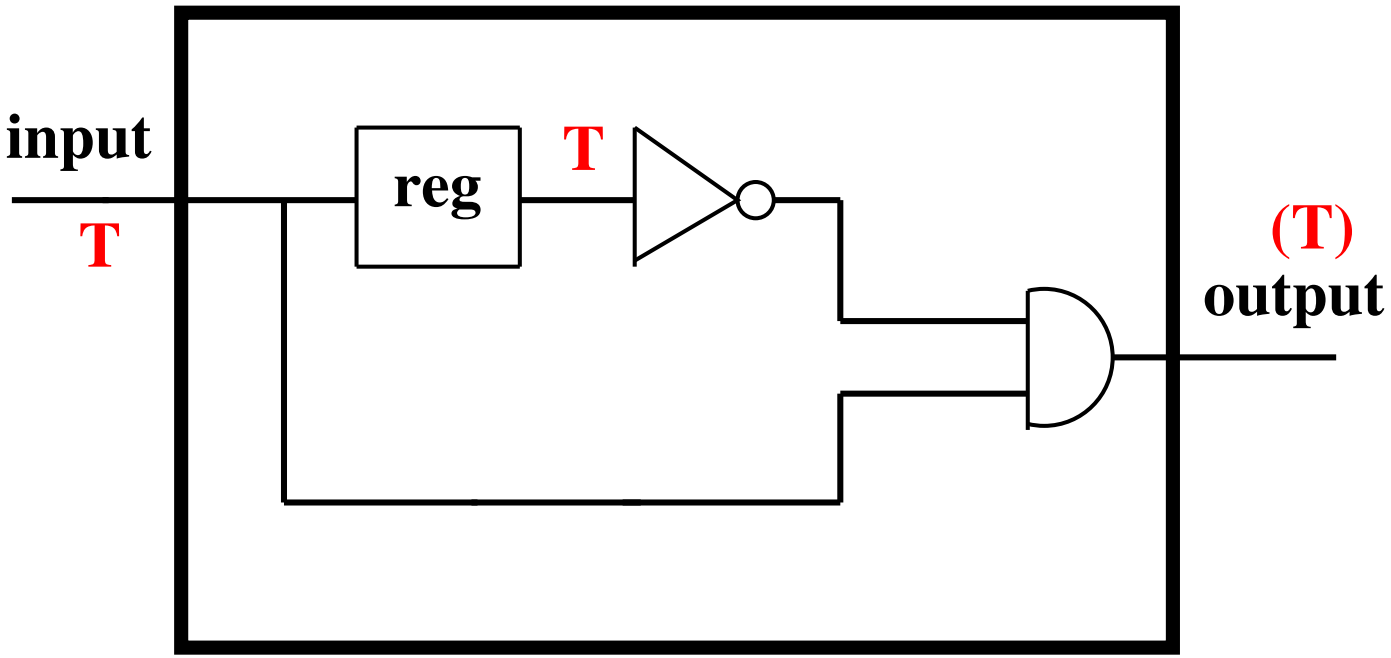


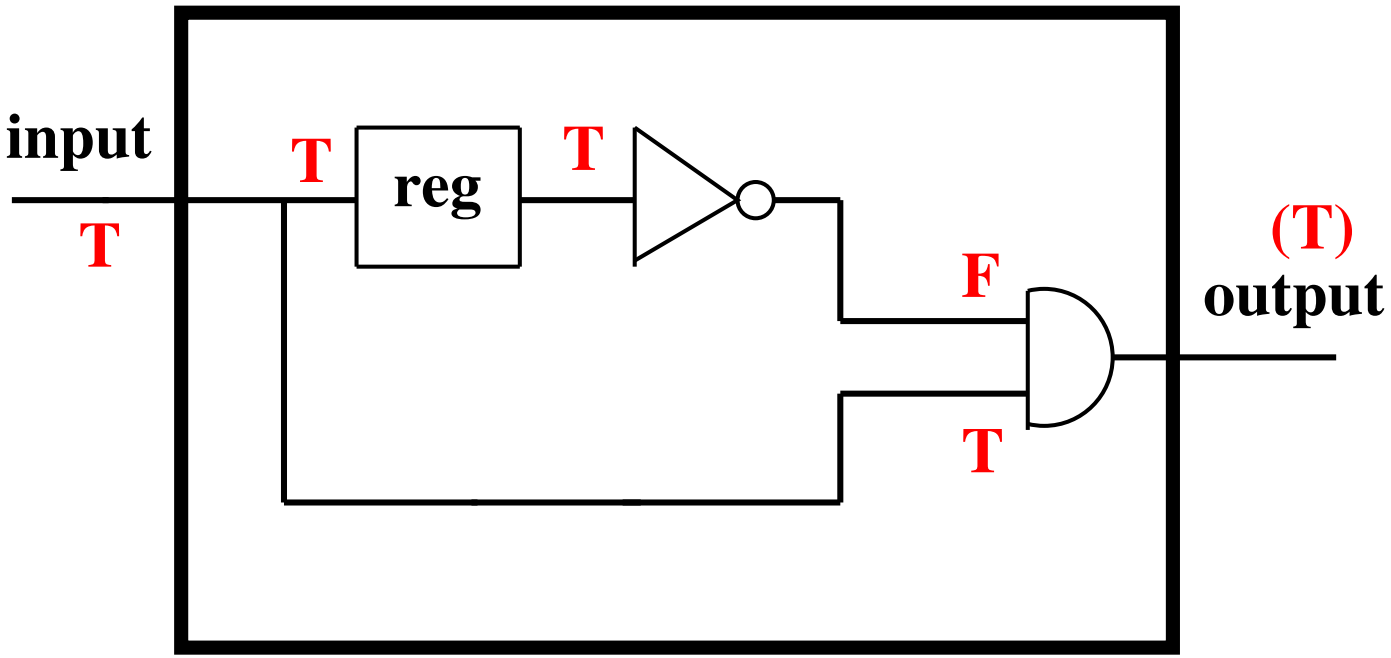


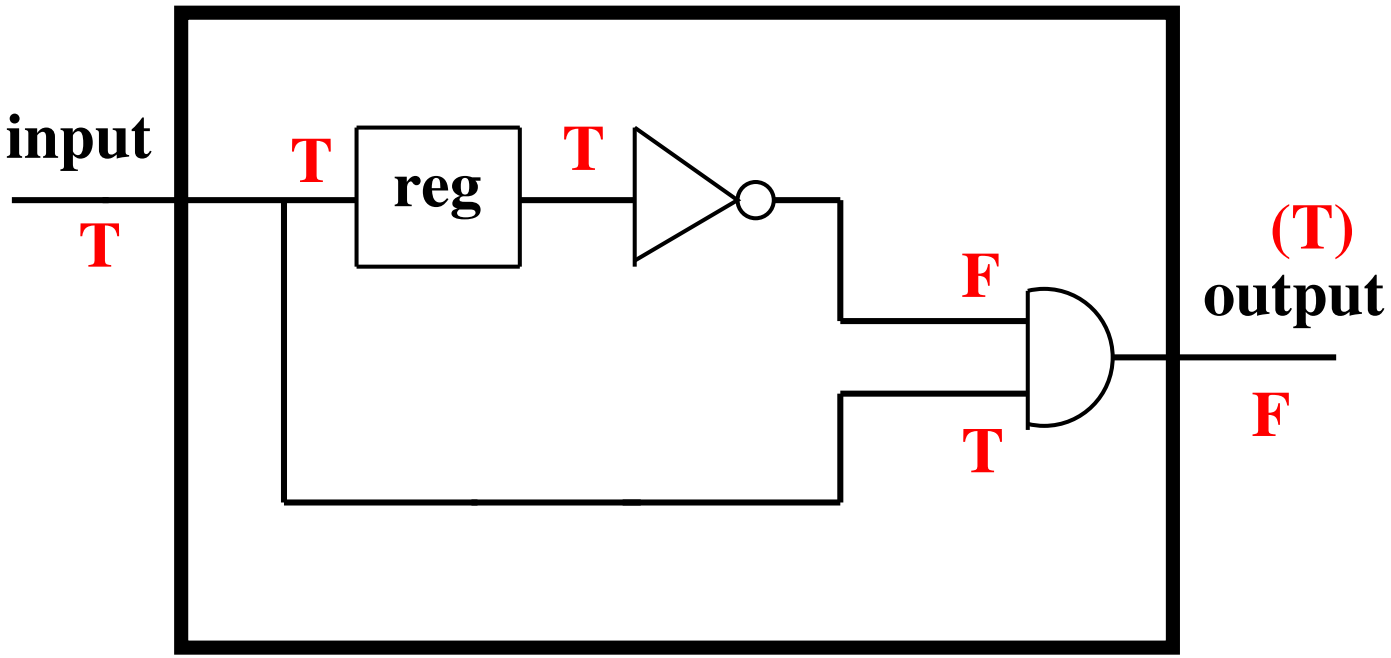

input
T
reg
F
output

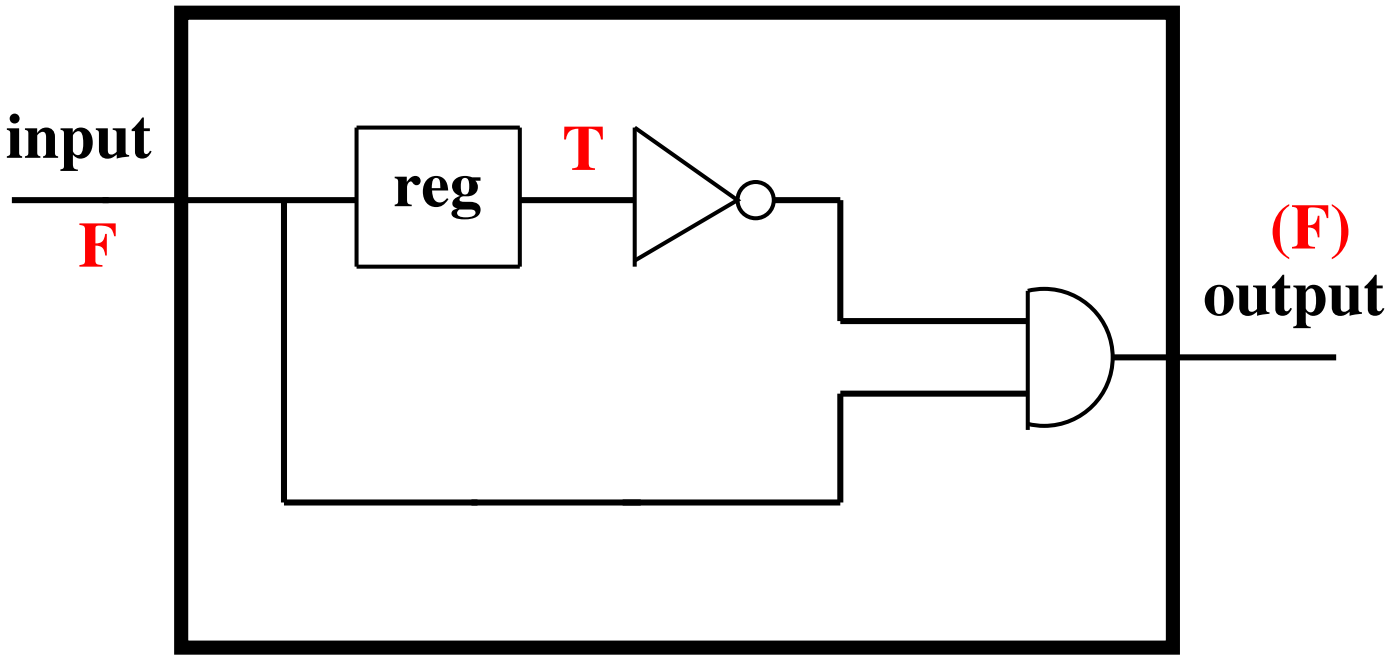

input
T
T
reg
F
T
T
T
output


input
T
T
reg
F
T
T
T
output
T

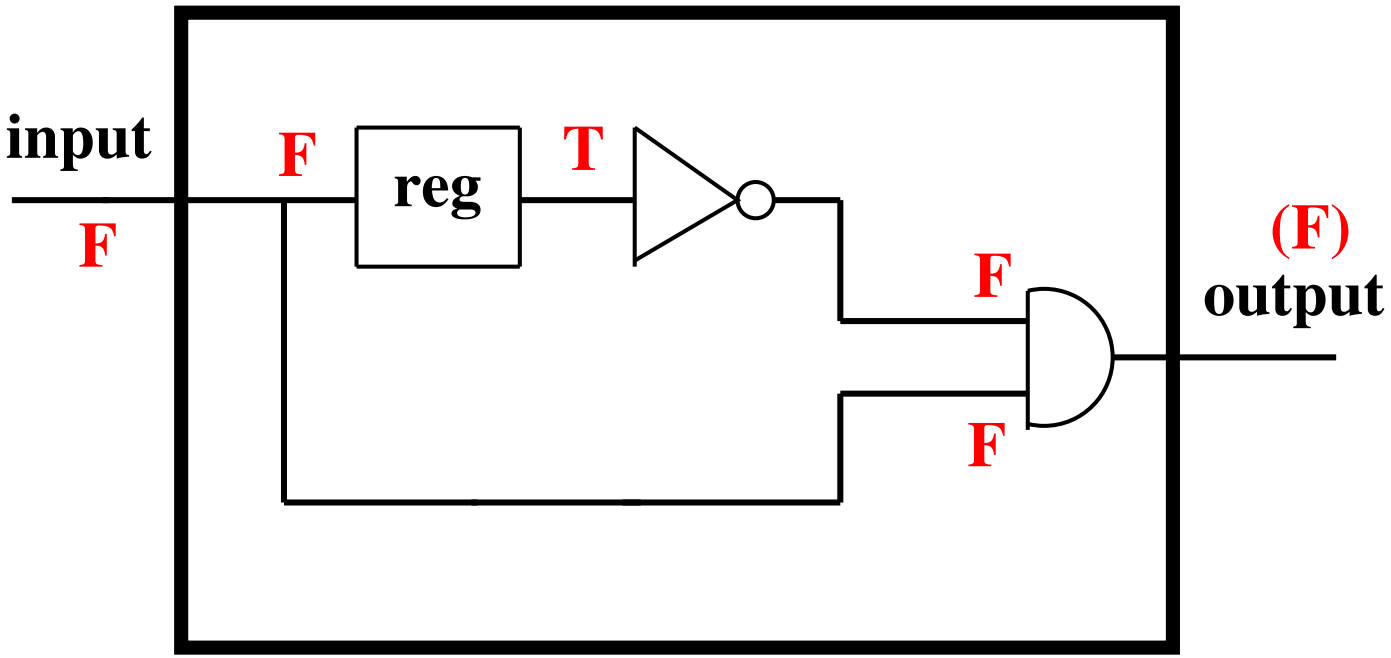

input
T
reg
T
(T)
output

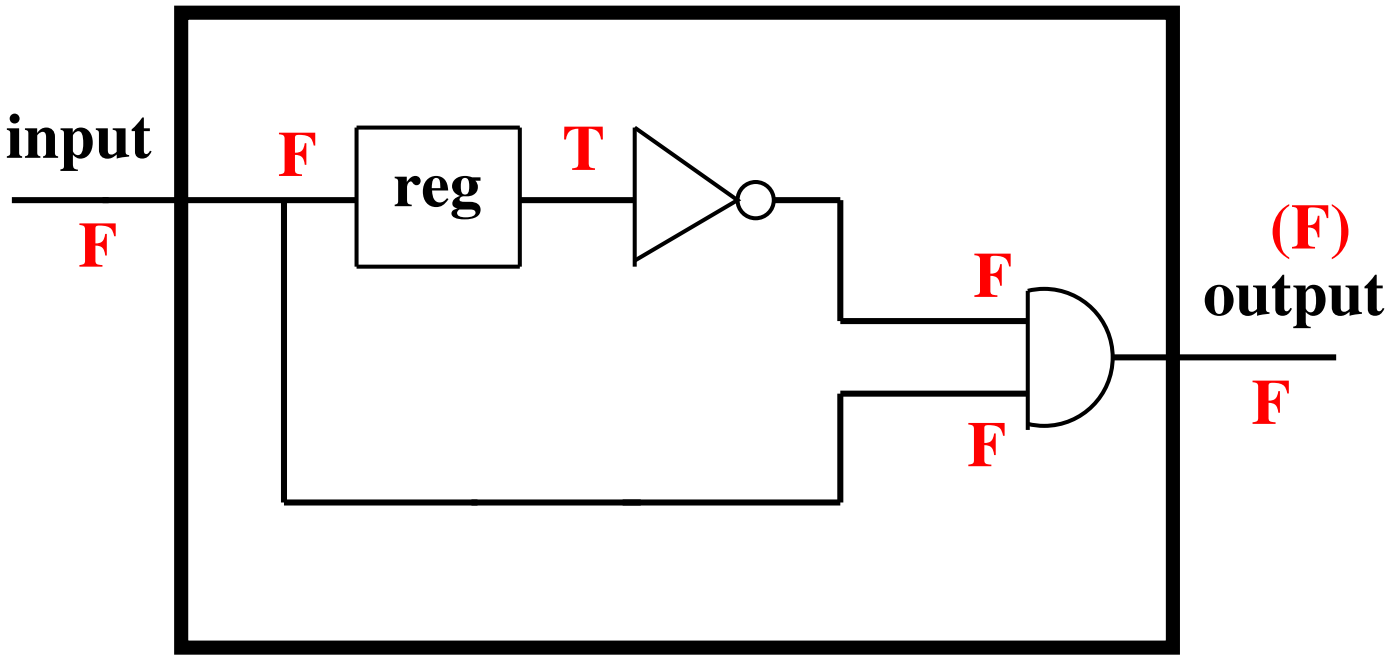

input
T
T
reg
T
F
T
(T)
output

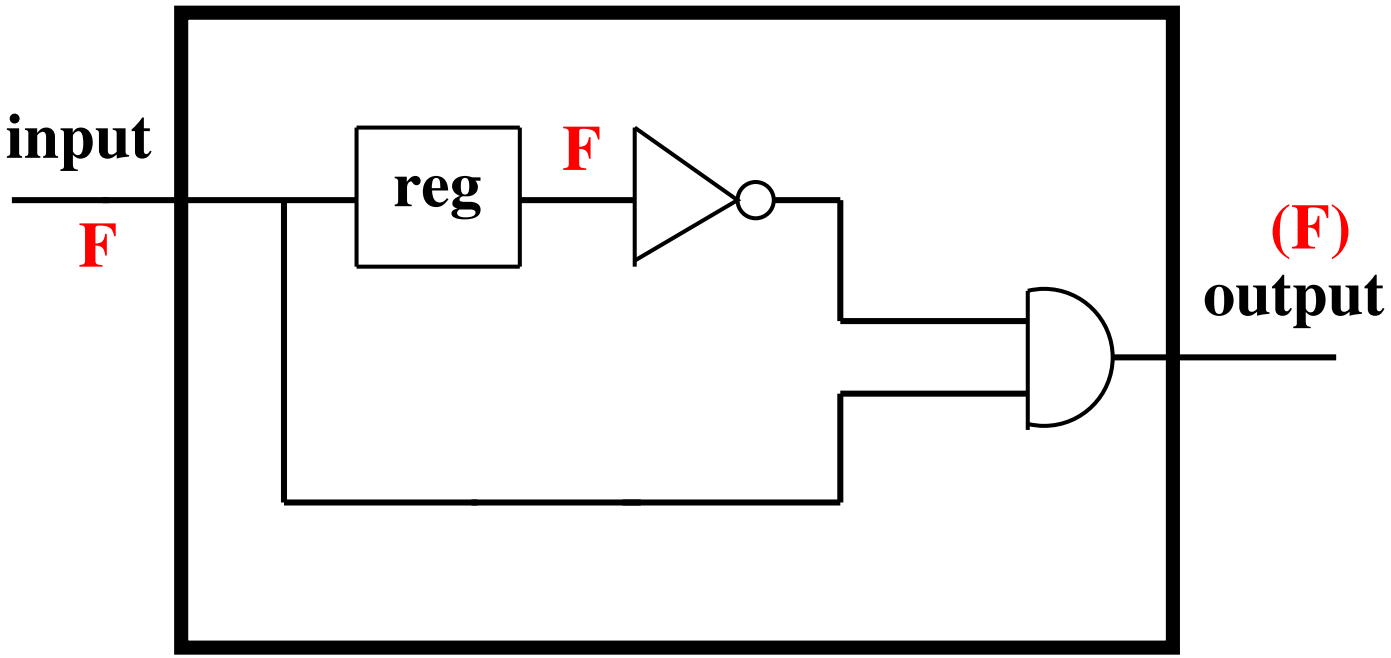

input
T
T
reg
T
F
T
(T)
output
F

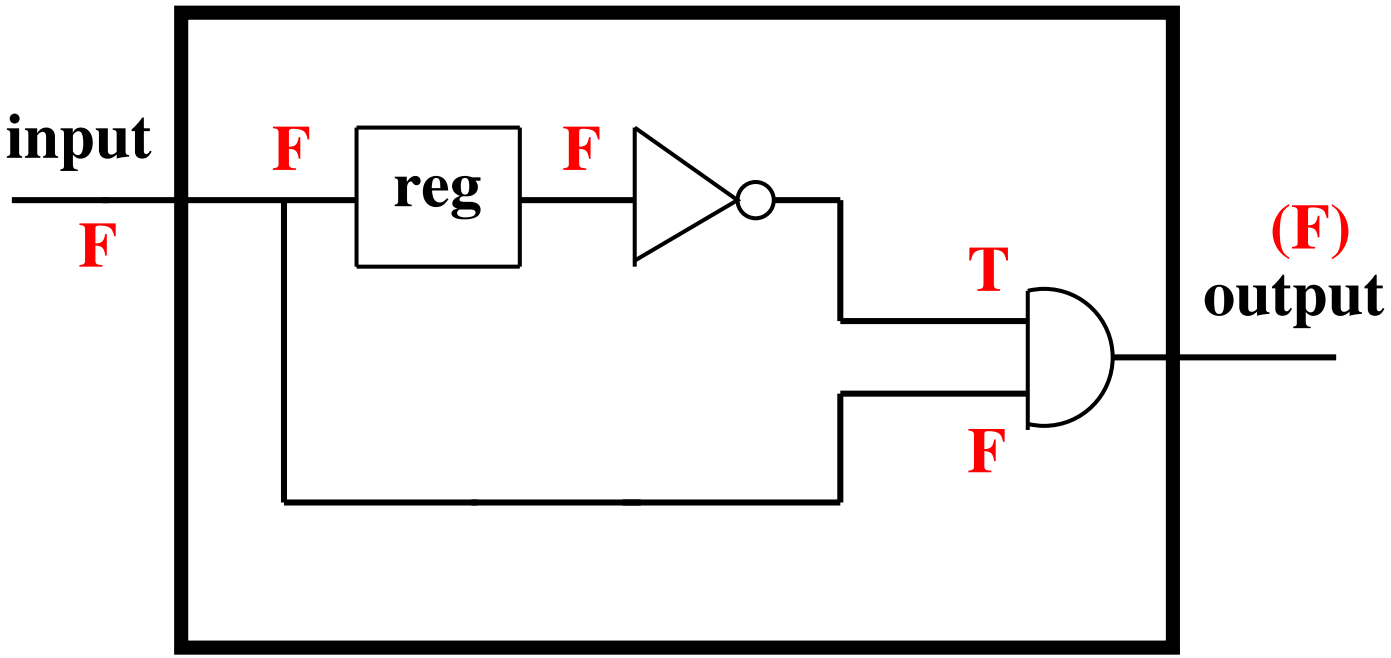

input
F
reg
T
(F)
output

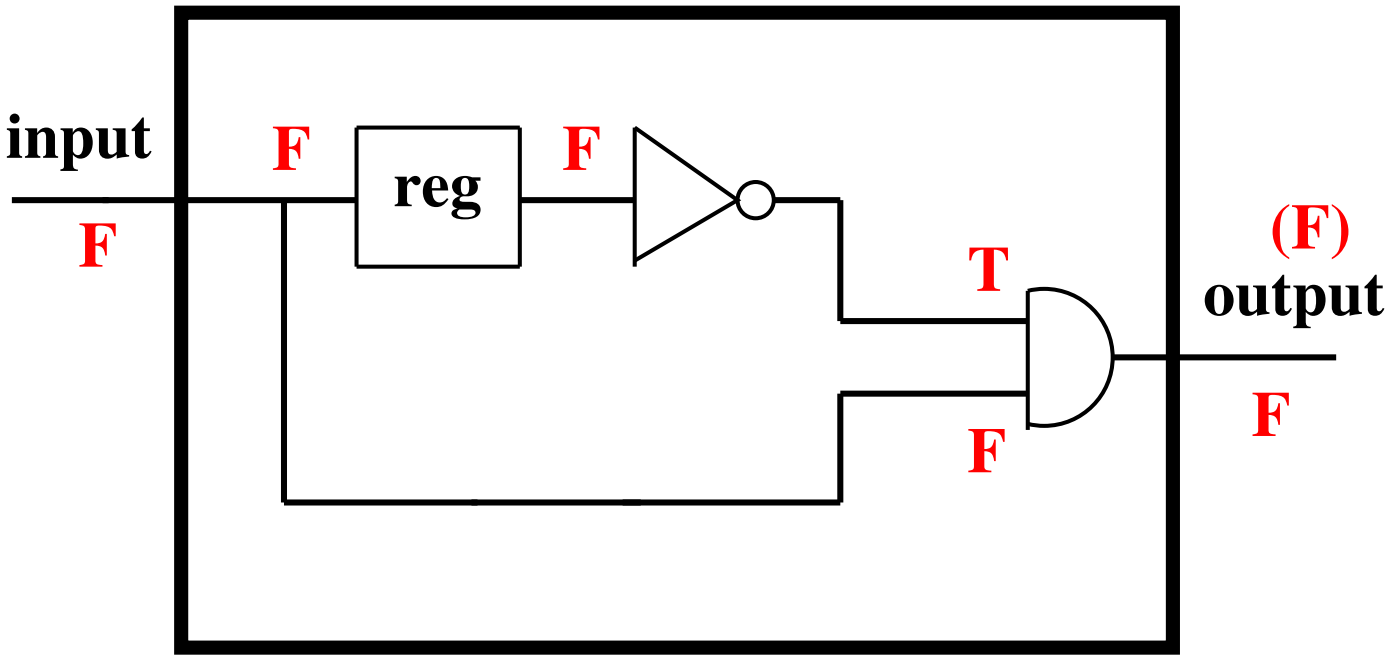
Computation with PULSER1

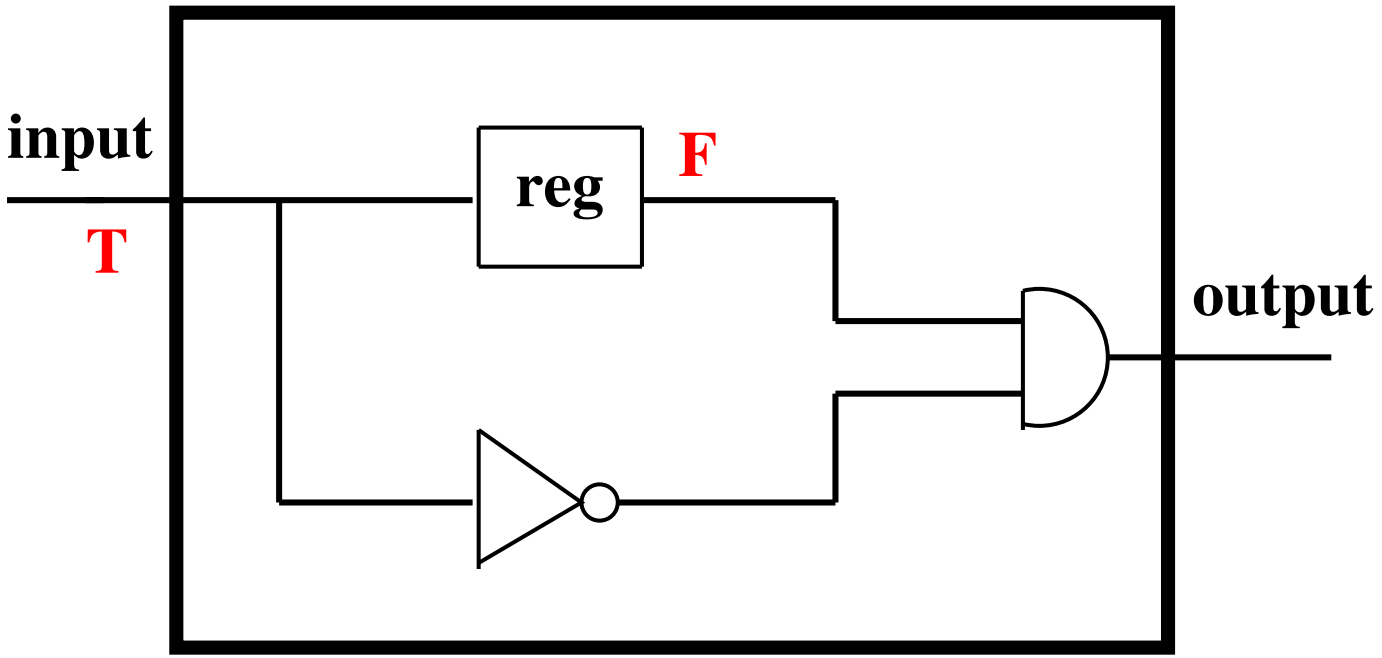


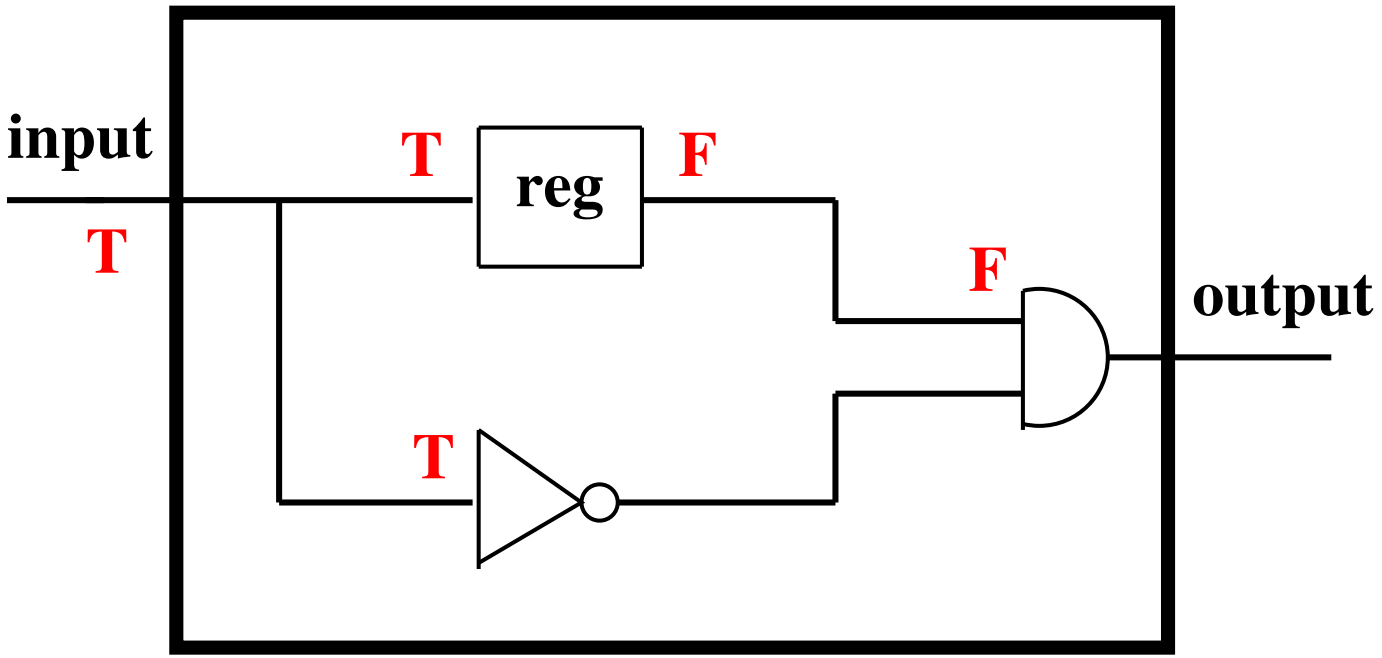

input
F
F
reg
T
F
F
(F)
output
F

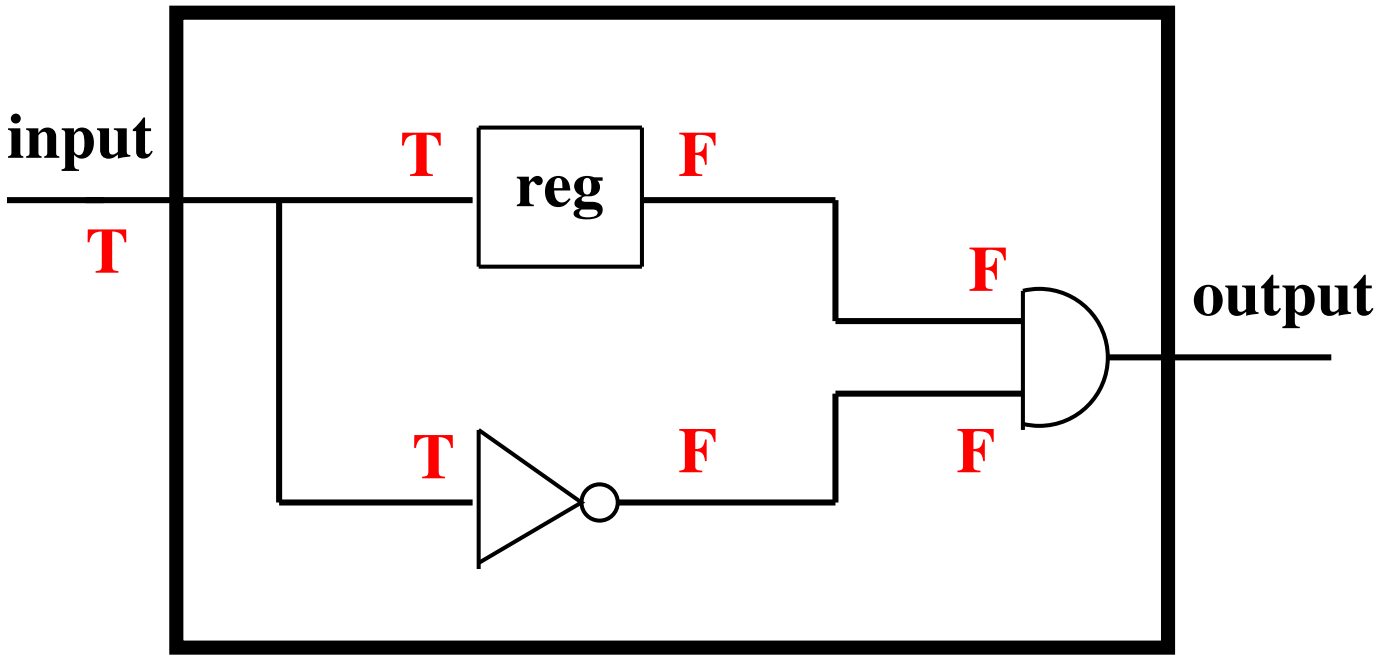

input
F
reg
F
(F)
output

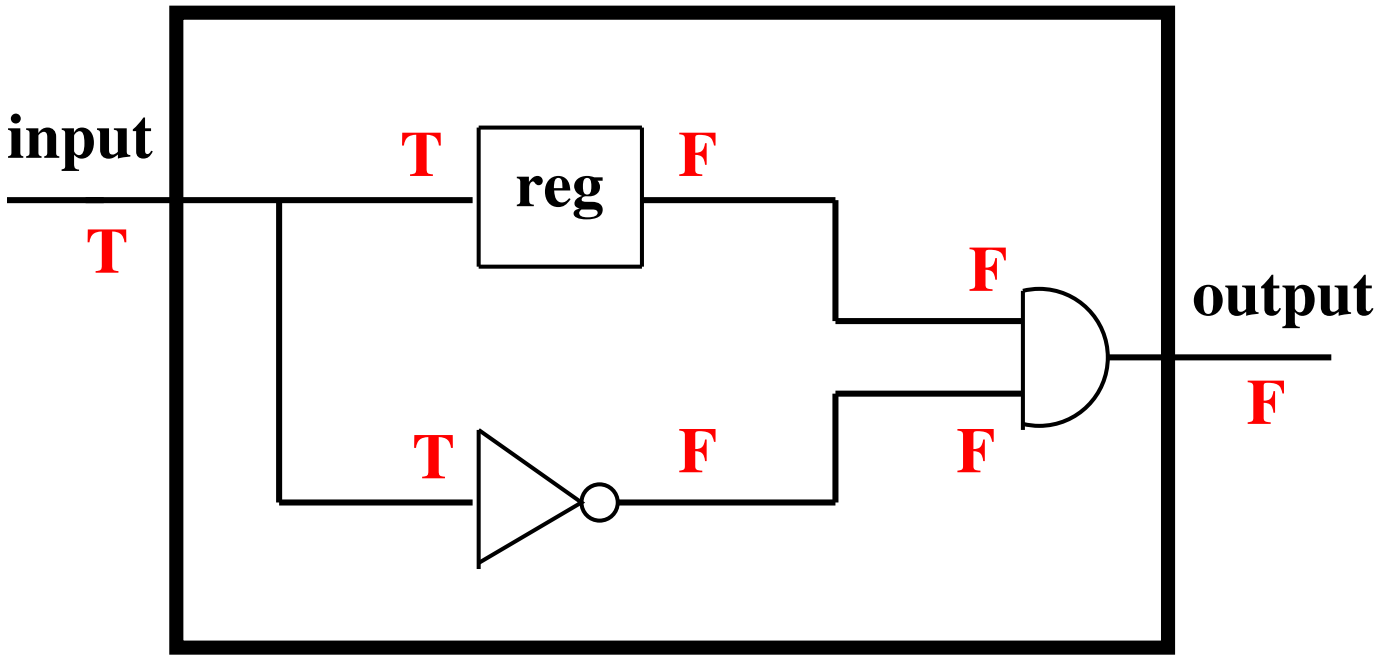

input
F
F
reg
F
T
F
(F)
output

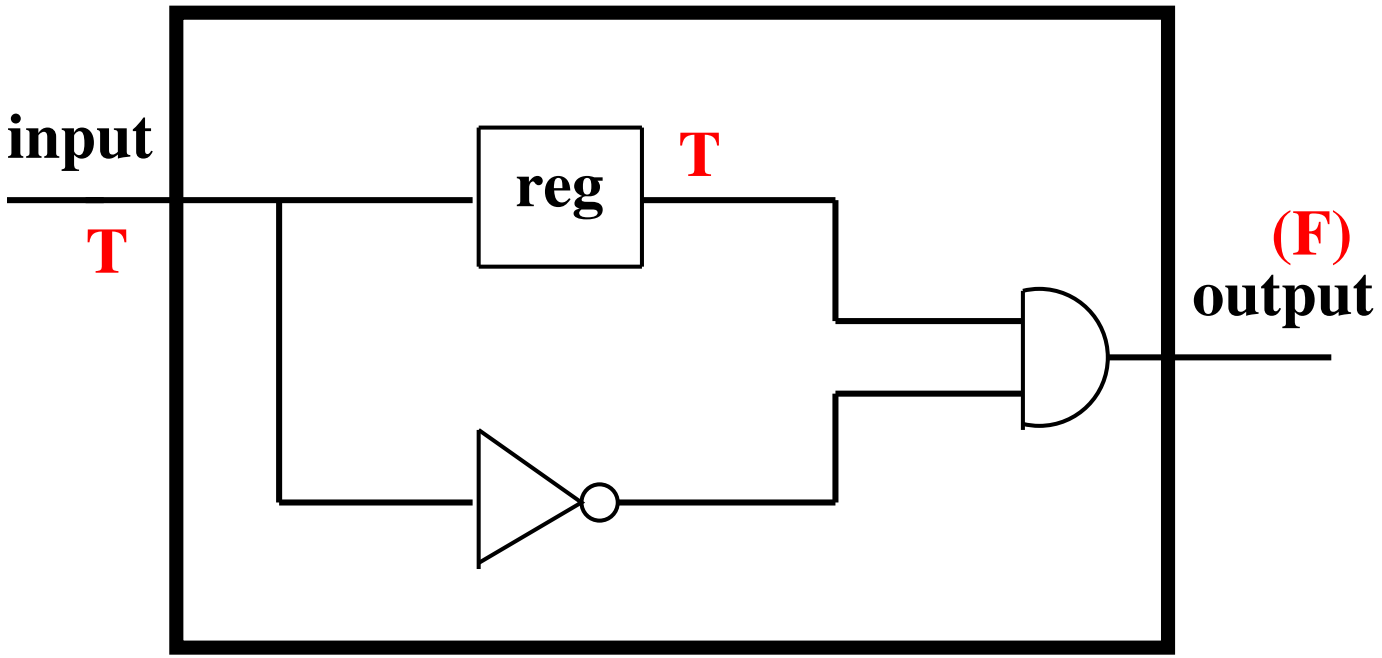

input
F
F
reg
F
T
F
(F)
output
F

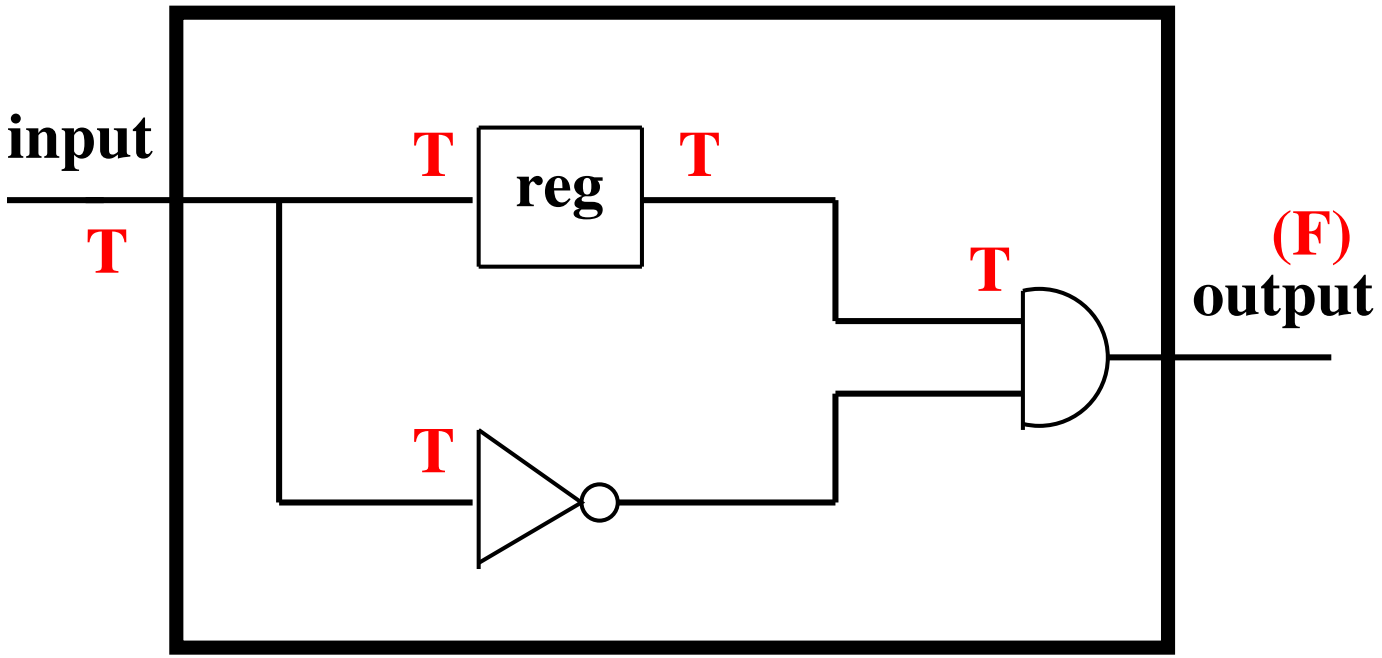

input
T
reg
F
output

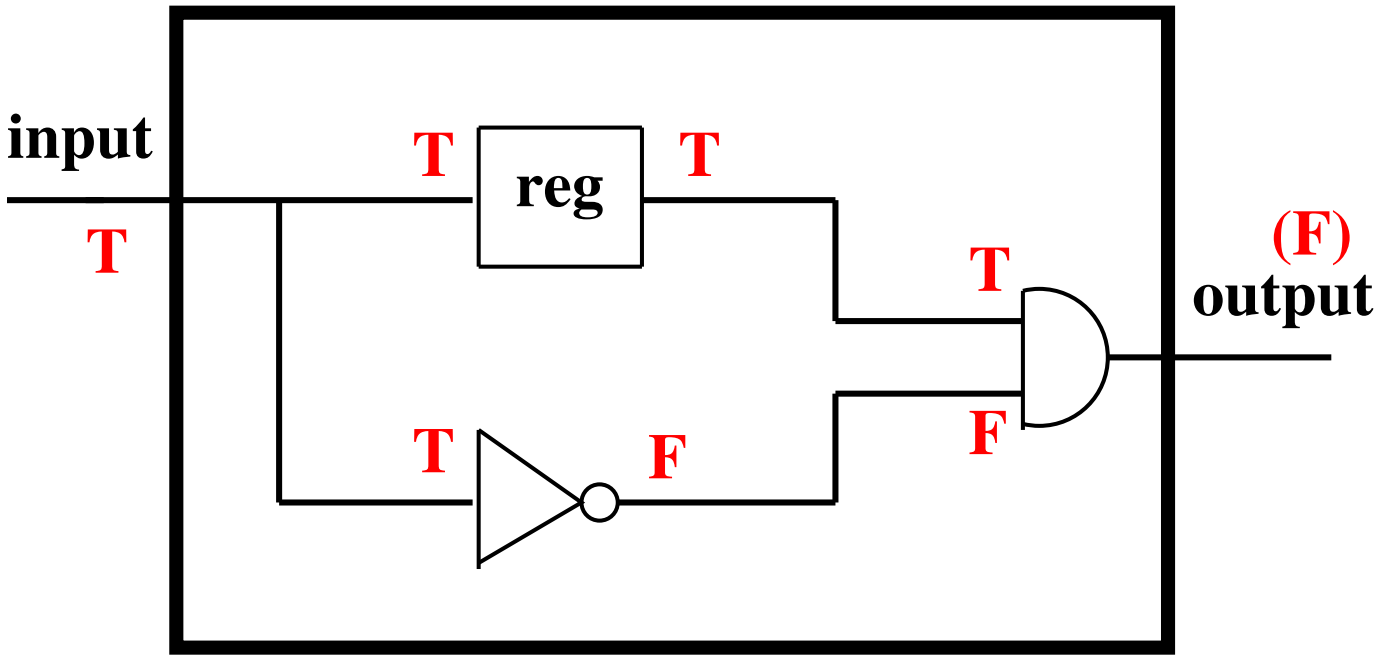

input
T
T
reg
F
F
output
T

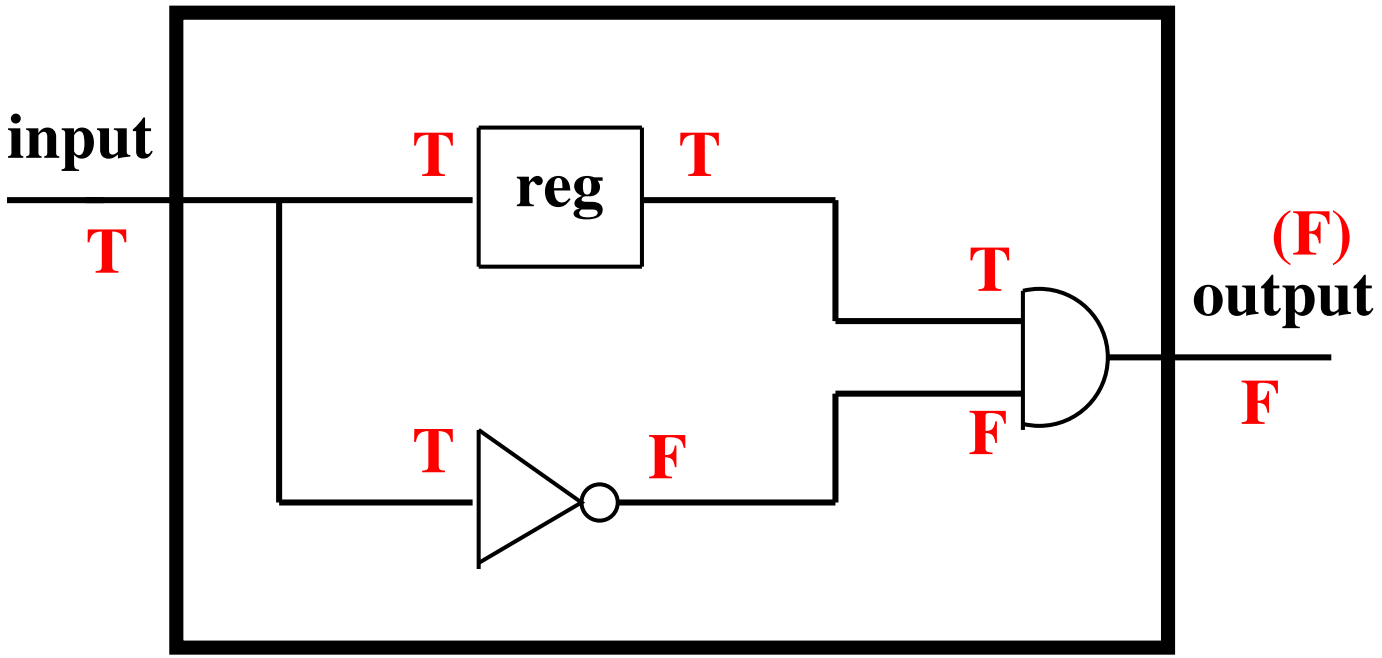

input
T
T
reg
F
F
output
T
F
F

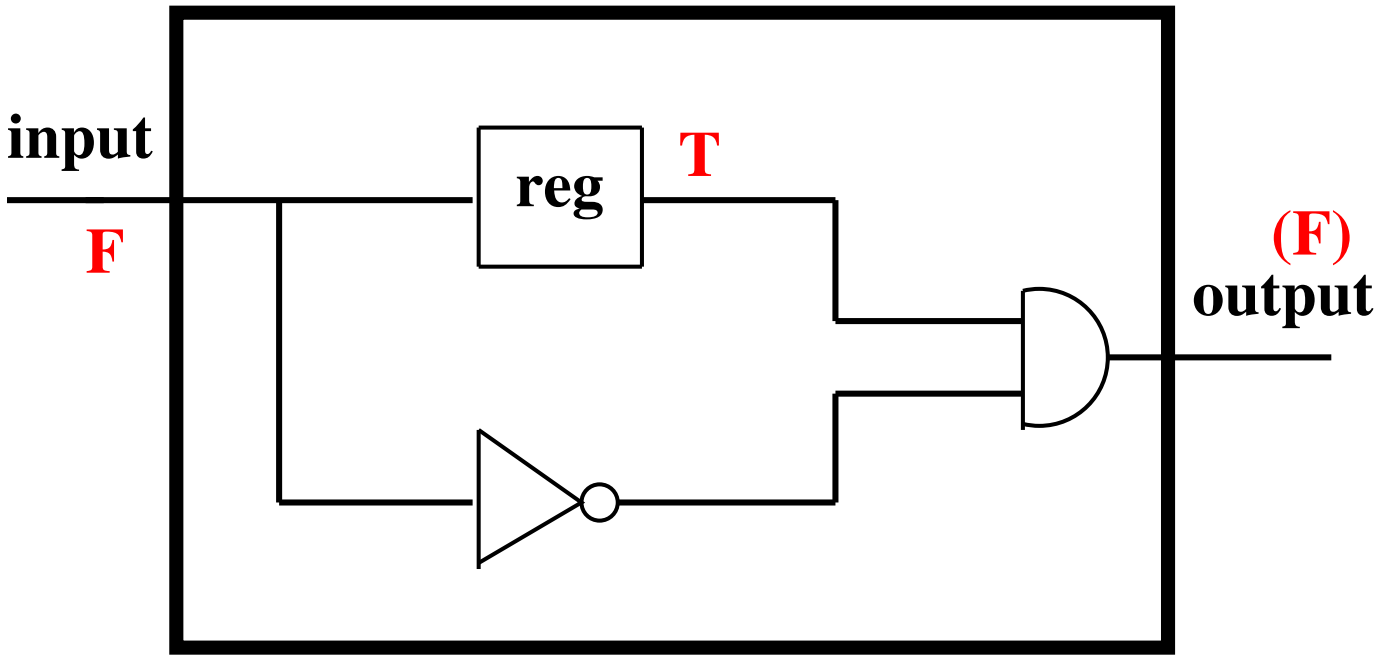

input
T
T
reg
F
F
output
F
T
F
F
F

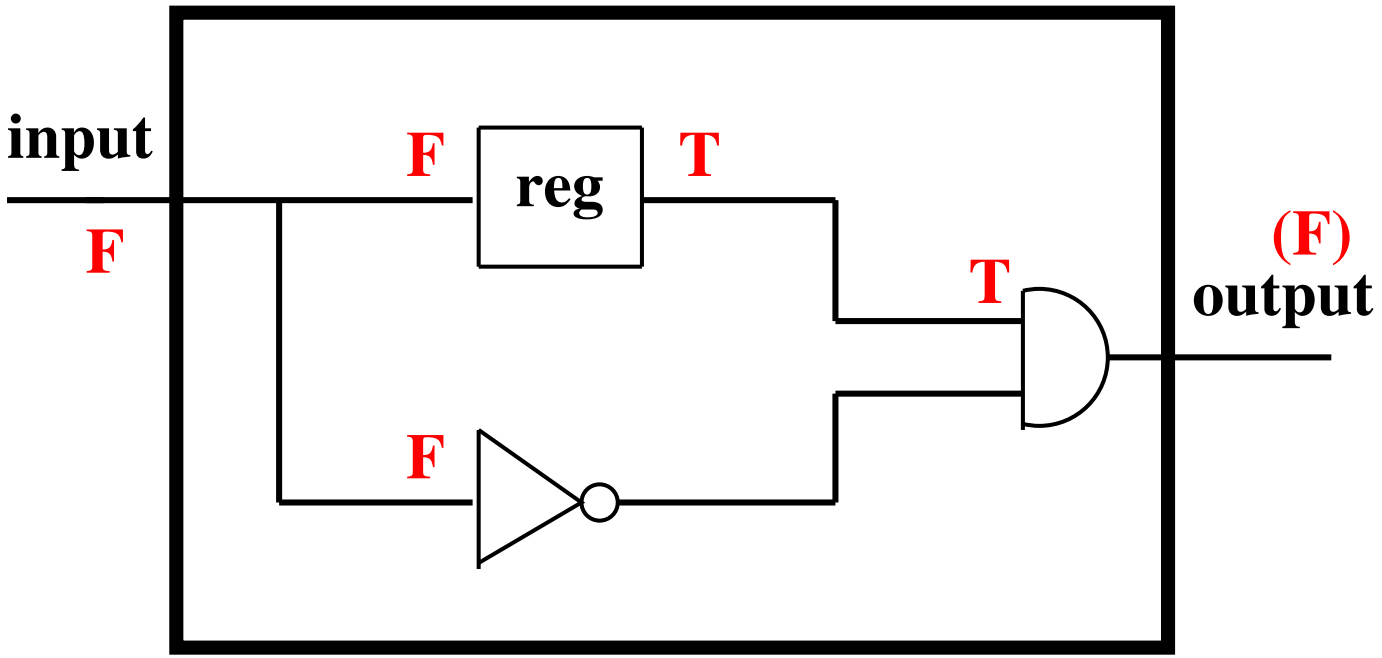

input
T
reg
T
(F)
output

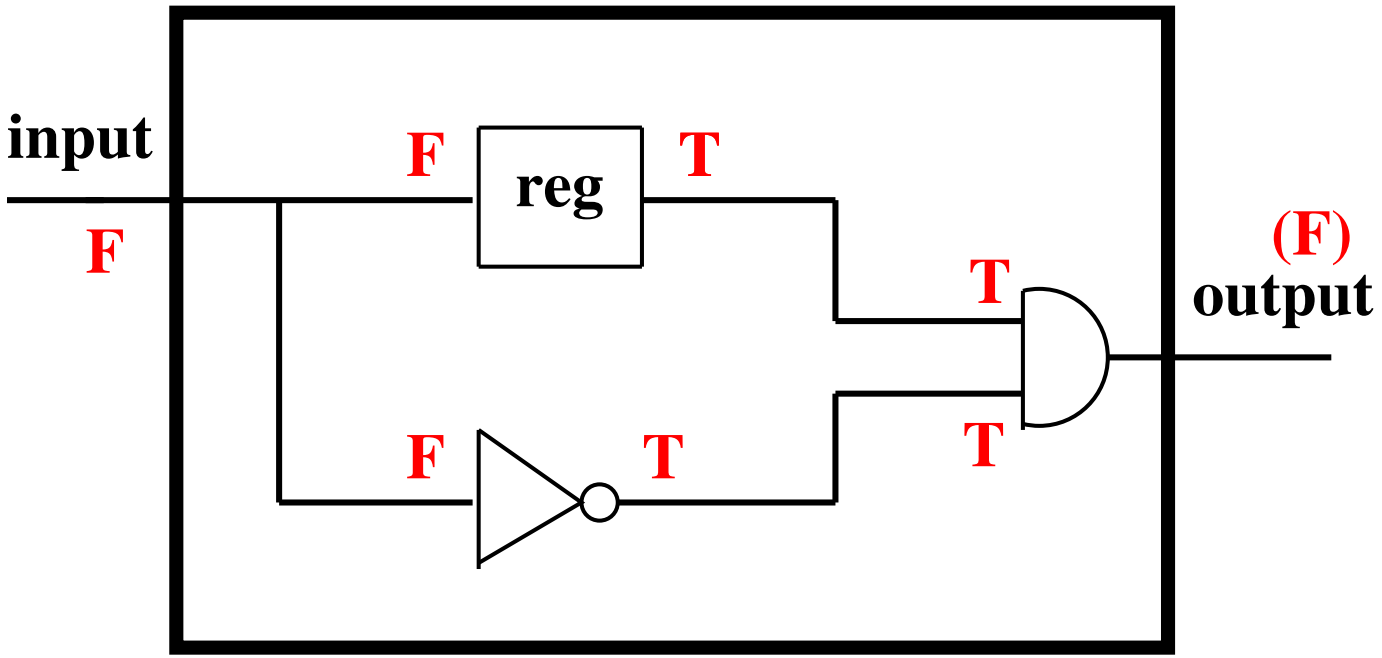

input
T
T
reg
T
T
T
(F)
output

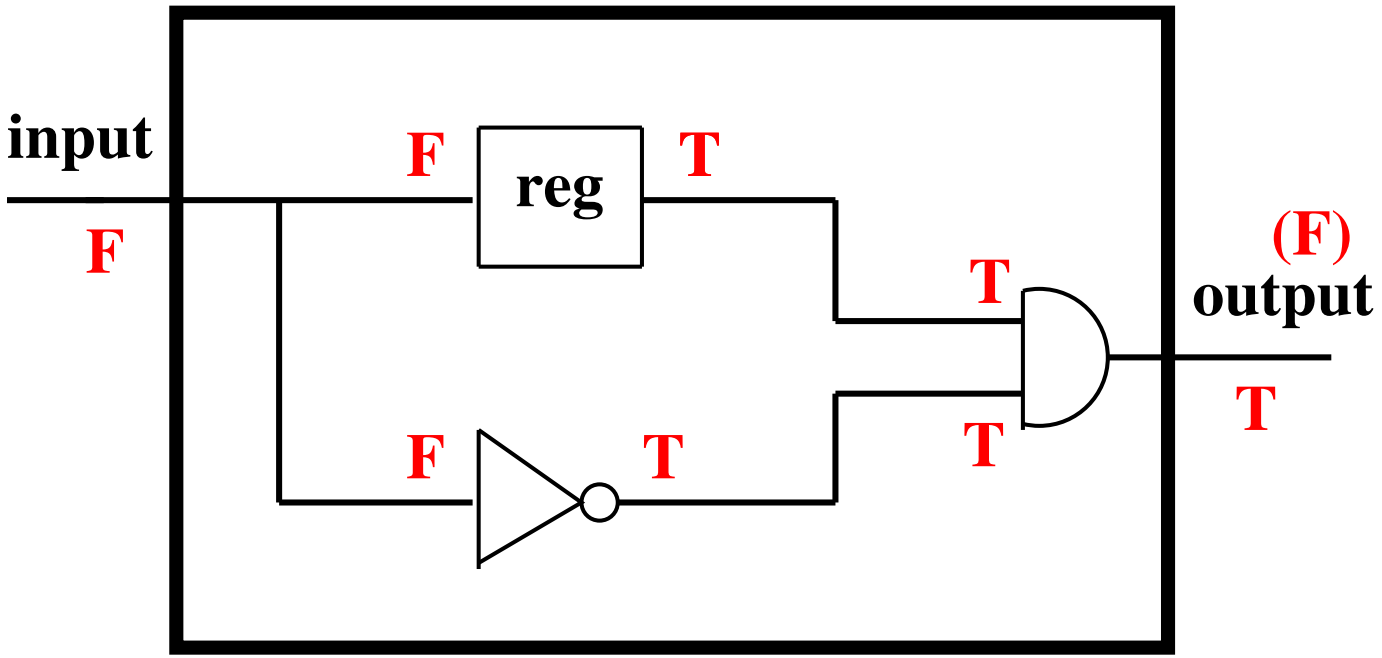

input
T
T
reg
T
T
T
(F)
output
T
F
F

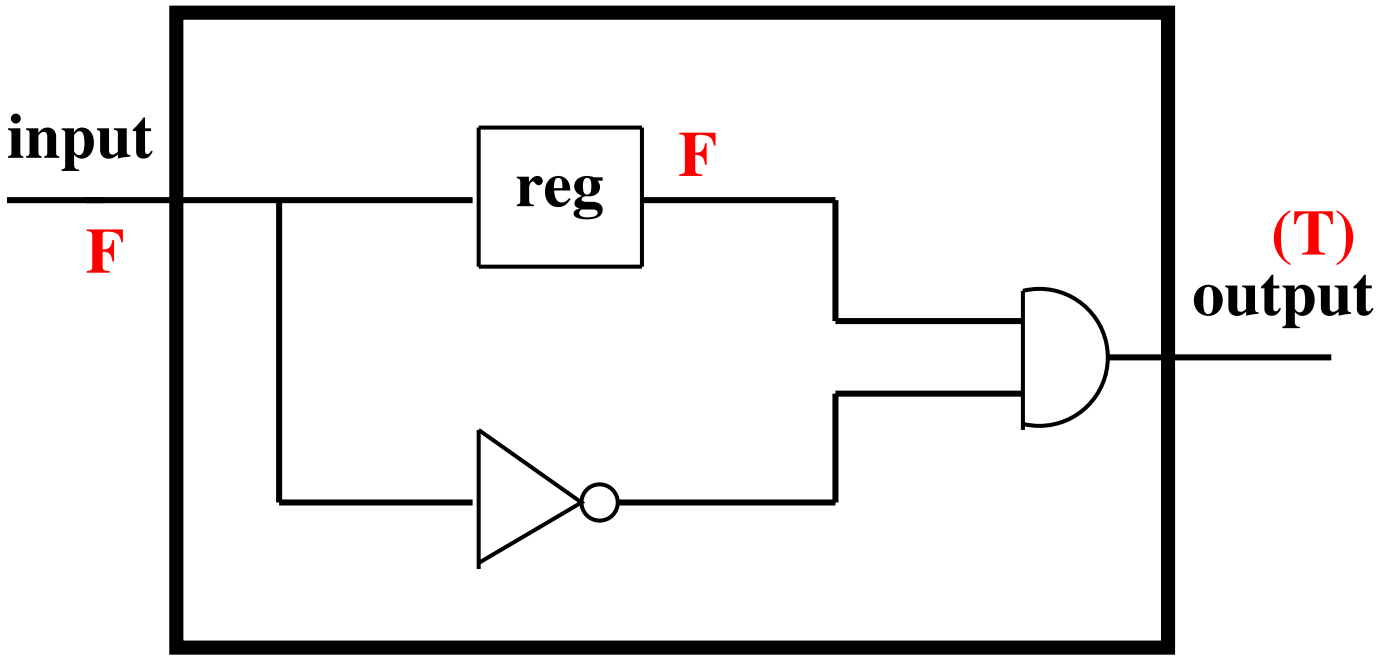

input
T
T
reg
T
T
T
(F)
output
F
T
F
F

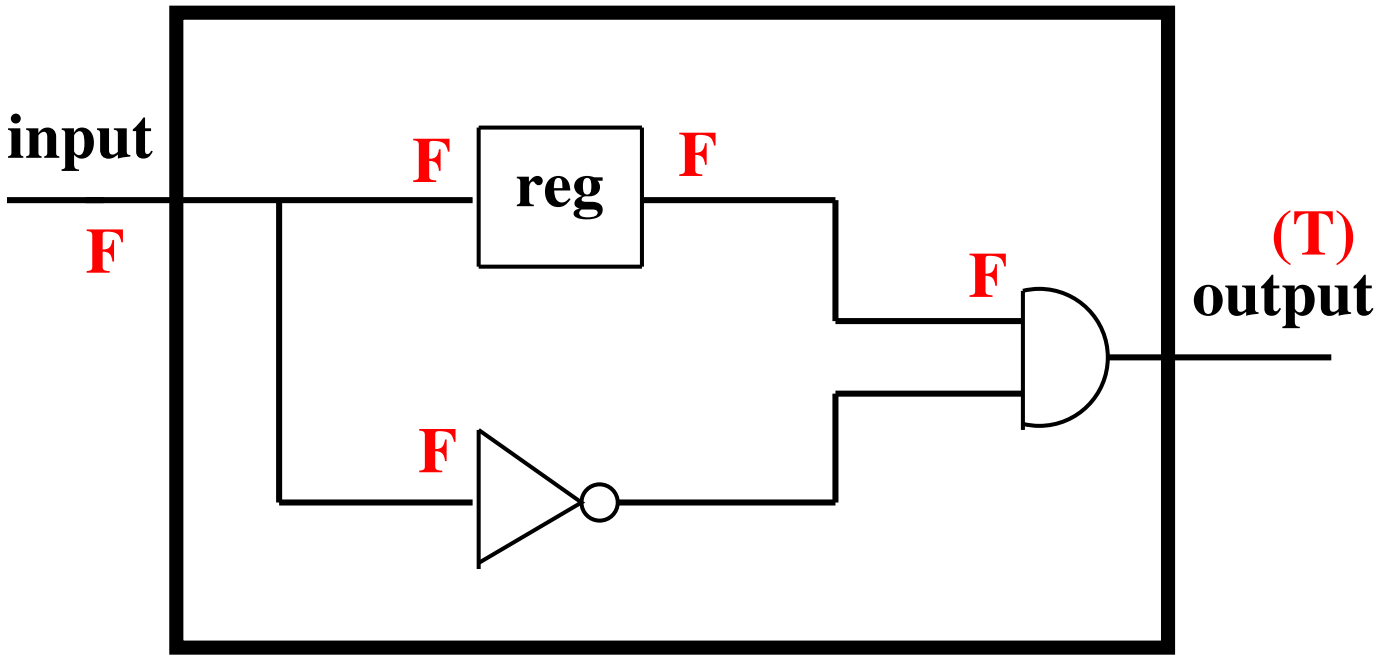

input
F
reg
T
(F)
output

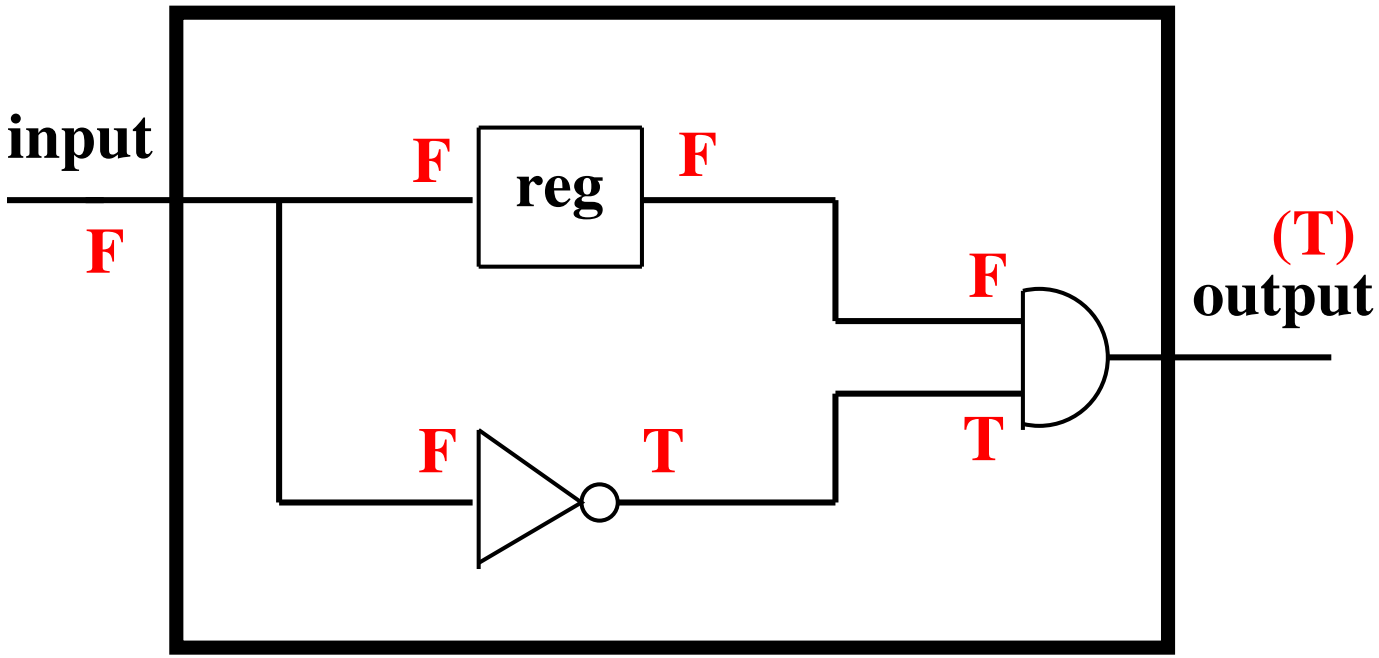

input
F
F
reg
T
T
F
(F)
output

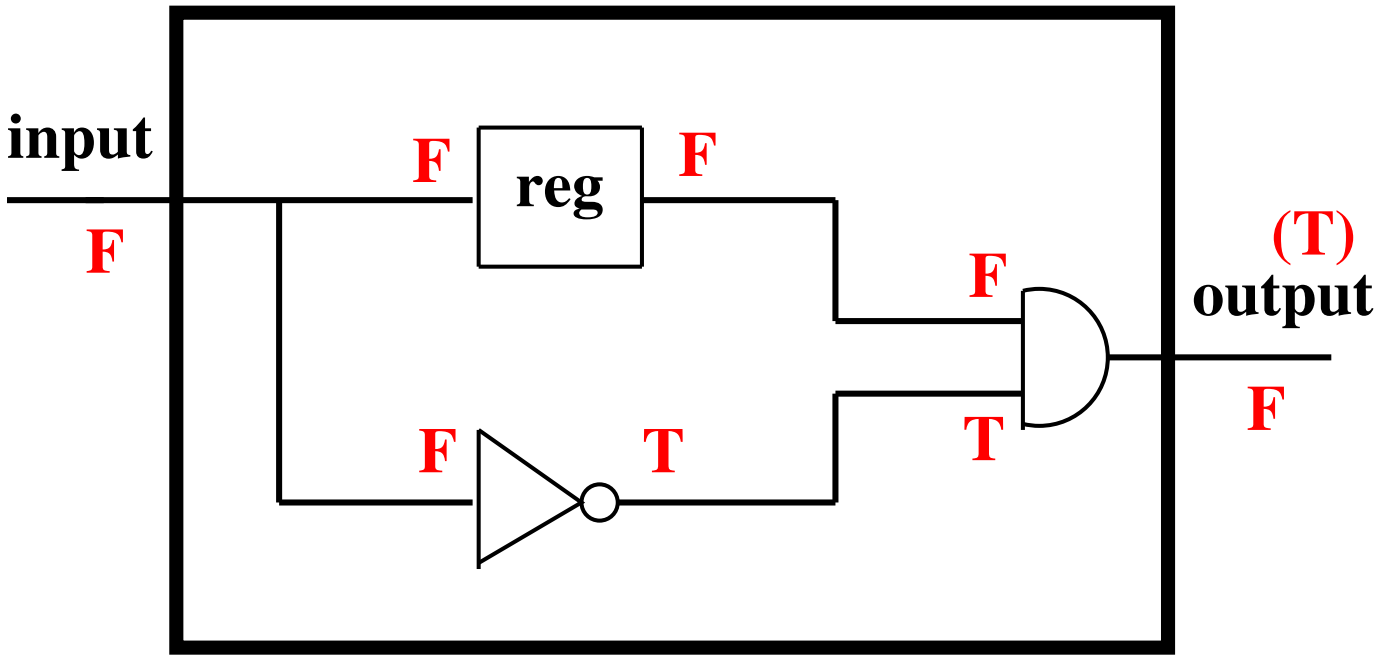

input
F
F
reg
T
F
T
T
(F)
output
F
T
T


input
F
F
reg
T
T
F
T
T
(F)
output
T


input
F
reg
F
(T)
output


input
F
F
reg
F
F
F
(T)
output


input
F
F
reg
F
F
F
(T)
output
F
T
T


input
F
F
reg
F
F
(T)
output
F
F
T
F
T