Electronic design with evolutionary algorithms

Adrian Thompson et al. (University of Sussex, UK)

Review by Alain Girault

The idea

To use evolutionary algorithms in order to design electronic circuits

➪ sometimes known as "genetic algorithms"

In order to obtain circuits with an emergent behaviour

Emergent = behaviour that cannot be predicted in detail given only the knowledge of the individual components and connections

➪ too complex for a human brain to understand!

The experiments

Try to design a robust electronic circuit for tone detection

Use Xilinx FPGA XC6216 circuits

Size used = 10×10 cells only, i.e., 100 cells (out of 64×64)

Use 4 different circuits under 4 different temperature conditions

➪ hence robustness

Specifications of the tone detector

input = square wave either 1kHz or 10kHz

output = 0 or 1 depending on the input

Specifications of the tone detector

input = square wave either 1kHz or 10kHz

output = 0 or 1 depending on the input

Easy to design with conventional method

But you would get a much bigger circuit

The evolution strategy

A (1+1) Evolution Strategy [Schwefel & Rudolph 1995]

A mutation selects one of the FPGA's 100 cells at random,

selects one of that cell's 10 muxes at random,

reconfigures it to select a different input at random

This mutation is applied three times to produce each offspring

Fitness function

F is used to select the best offspring at each generation

c is the FPGA chip number $\in [1, 4]$

S_1/S_{10} is the set of 1kHz/10kHz tone tests

t is the tone test number $\in S_1 \cup S_{10}$ and $T = |S_1 \cup S_{10}|$

The output of c is fed to an analogue integrator giving a value i_t^c proportional to the average output voltage of c over the test t

$$E^c = \frac{1}{2T} \left| \sum_{t \in S_1} i_t^c - \sum_{t \in S_{10}} i_t^c \right| \quad \text{and} \quad F = \min_{c=1}^{4} (E^c)$$

The algorithm

Download the initial parent configuration to the FPGAs
Measure F_{parent} over 50 tone tests
repeat
    Generate three mutations to update all FPGAs
    Measure $F_{offspring}$ over 24 tone tests
    if $F_{offspring} \geq F_{parent}$ **then**
        The offspring becomes the new parent
    else
        Revert the three mutations
    end if
    if 15 offspring have failed to replace their parent **then**
        Reset all the FPGAs and reconfigure them from the parent
    end if
until user decides

Choice of Adam/Eve

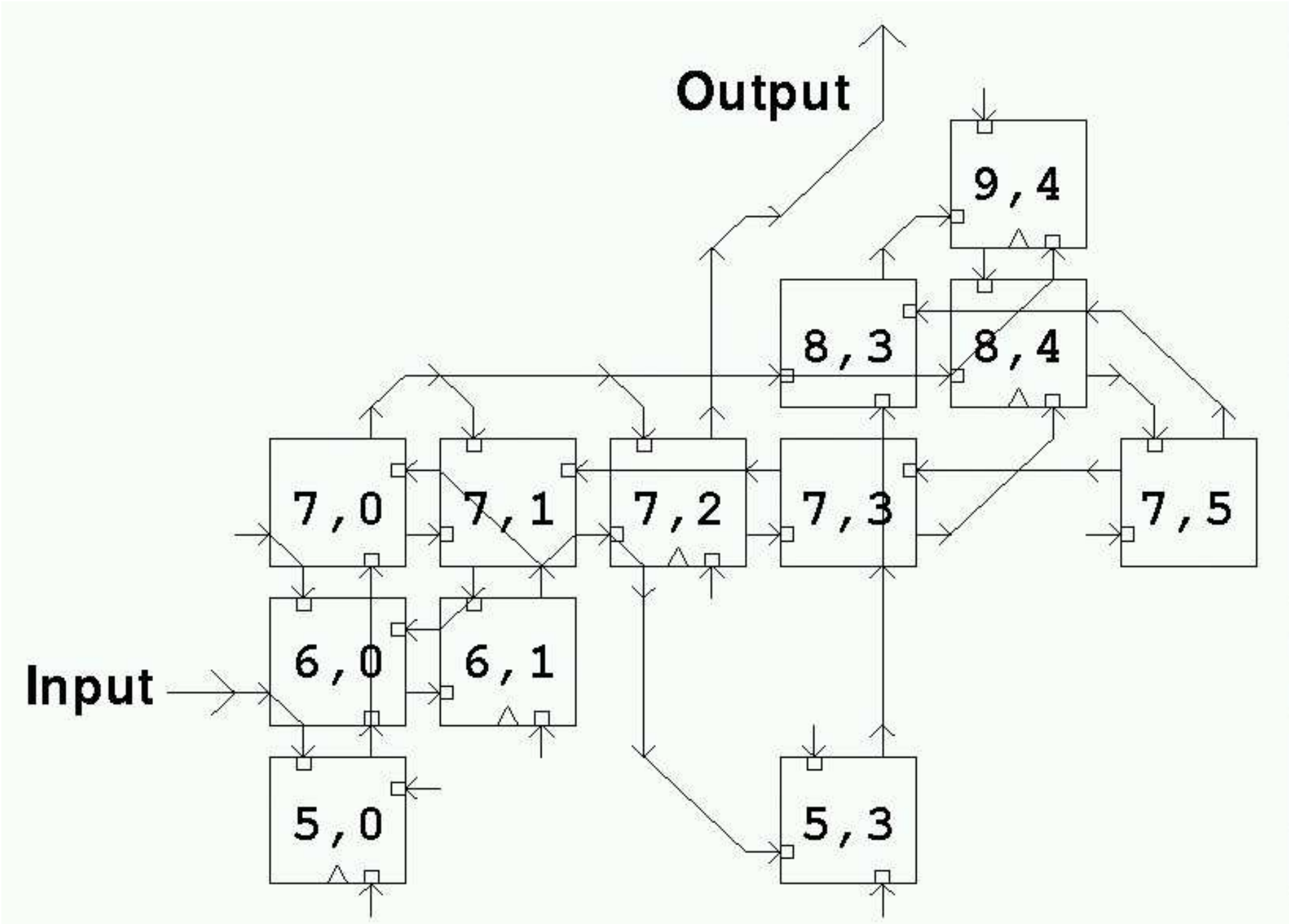
Generate circuits at random until one is found to have an above average fitness test:

- 75679 individuals were generated at random
- $F_{Adam/Eve}$ = 0.43

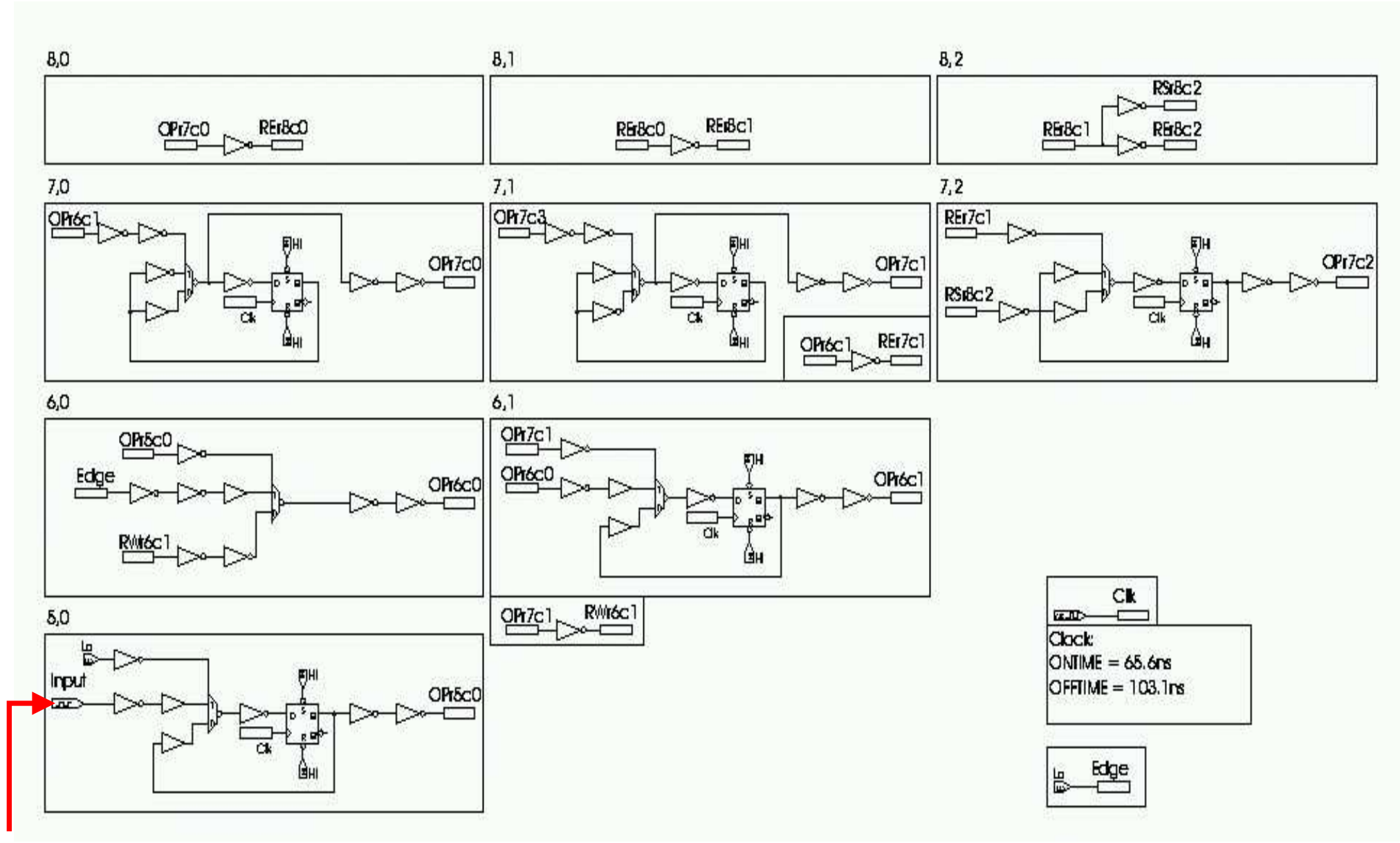
25000 triple-mutations out of 861348 attempted

- F_{final} = 6.17

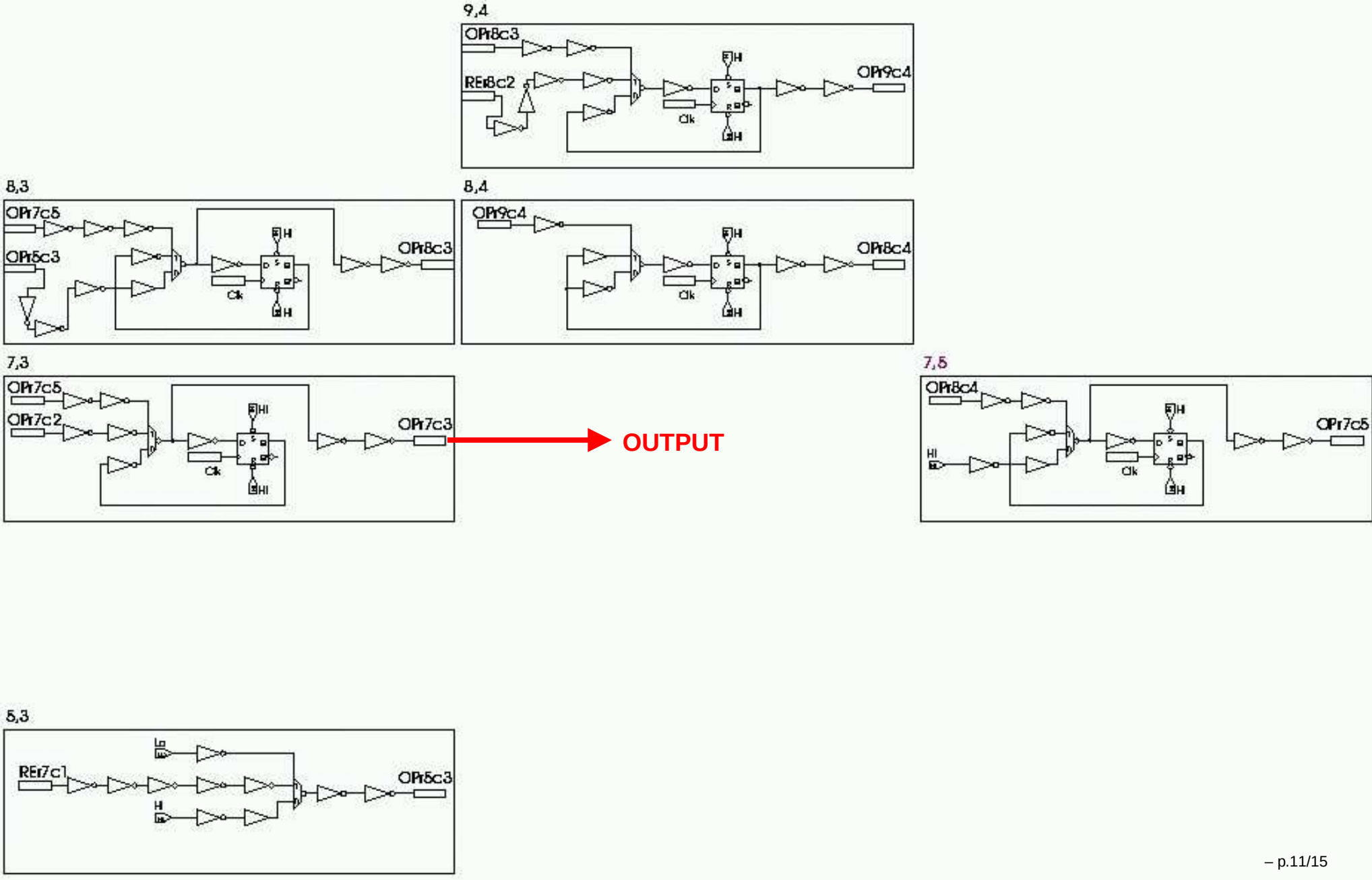
Functional part



Left part

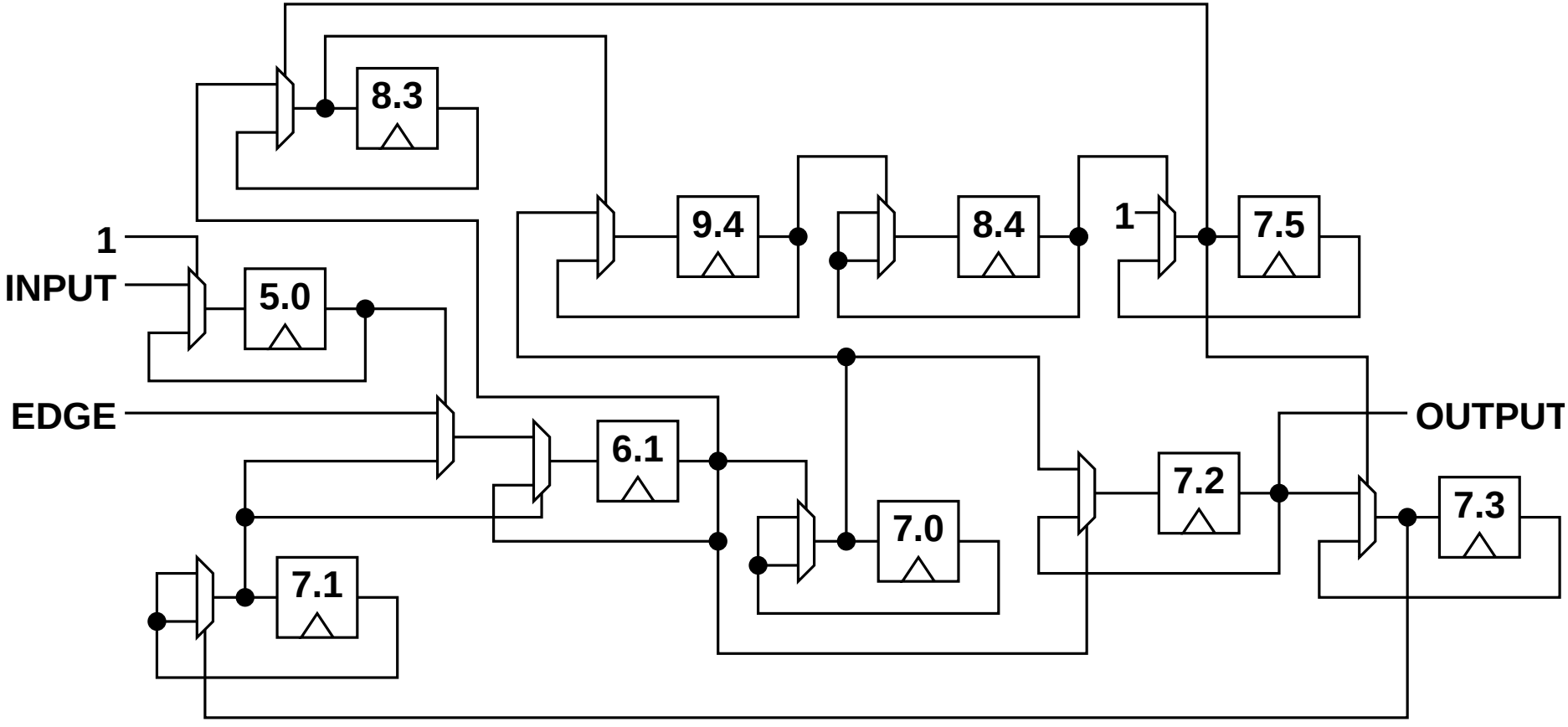

8,0
OPr7c0
REr8c0
8,1
REr8c0
REr8c1
8,2
RSr8c2
REr8c1
REr8c2
7,0
OPr6c1
Clk
OPr7c0
7,1
OPr7c3
Clk
OPr7c1
OPr6c1
REr7c1
7,2
REr7c1
RSr8c2
Clk
OPr7c2
6,0
OPr5c0
Edge
RWr6c1
OPr6c0
6,1
OPr7c1
OPr6c0
Clk
OPr6c1
OPr7c1
RWr6c1
5,0
Input
Clk
OPr5c0
Clock
ONTIME = 65.6ns
OFFTIME = 103.1ns
Edge
INPUT

Right part

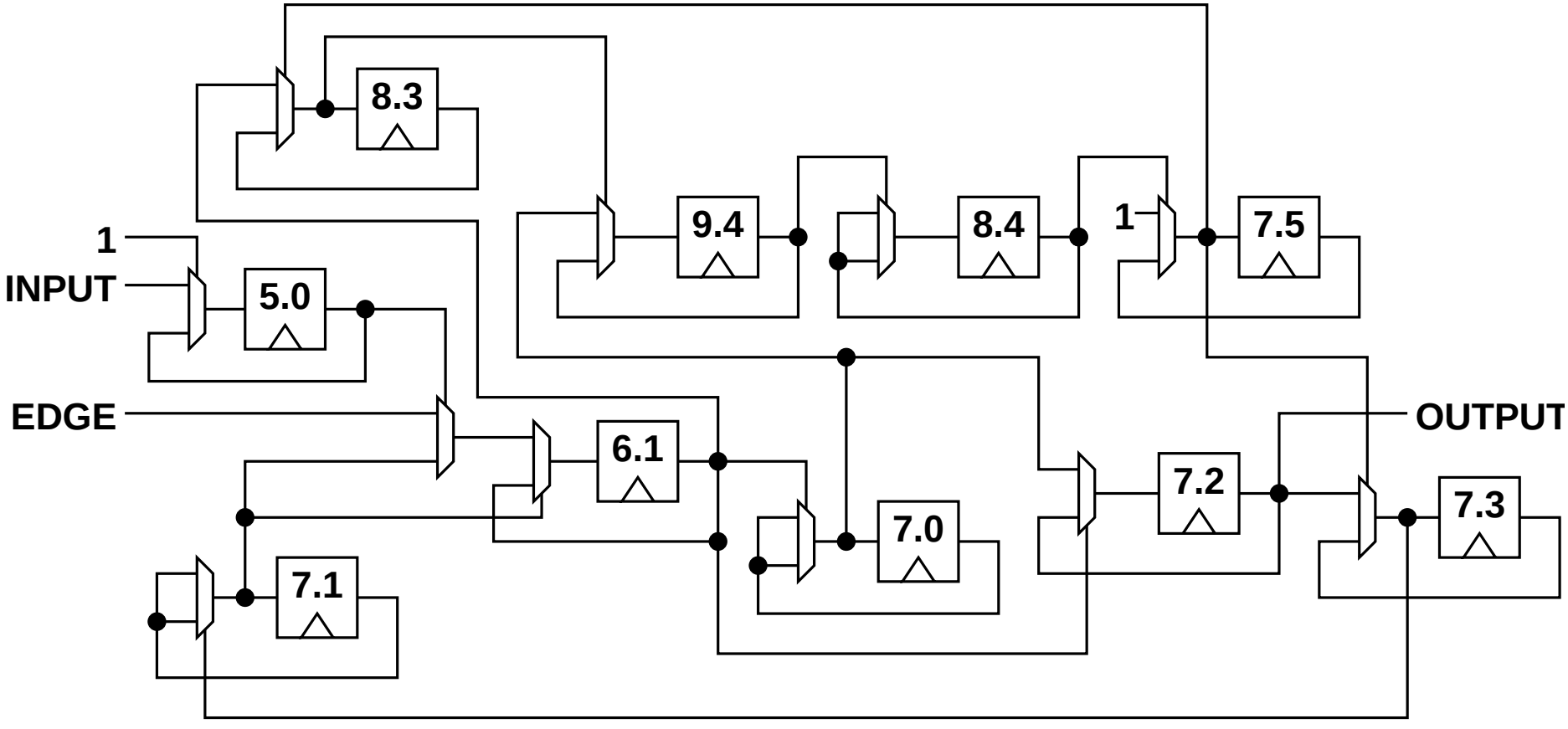

9,4
OPr8c3
REr8c2
OPr9c4
Clk
H
H
8,3
OPr7c5
OPr5c3
OPr8c3
Clk
H
H
8,4
OPr9c4
OPr8c4
Clk
H
H
7,3
OPr7c5
OPr7c2
OPr7c3
Clk
HI
HI
OUTPUT
7,5
OPr8c4
HI
OPr7c5
Clk
H
H
5,3
REr7c1
H
OPr5c3

A lot of nested loops

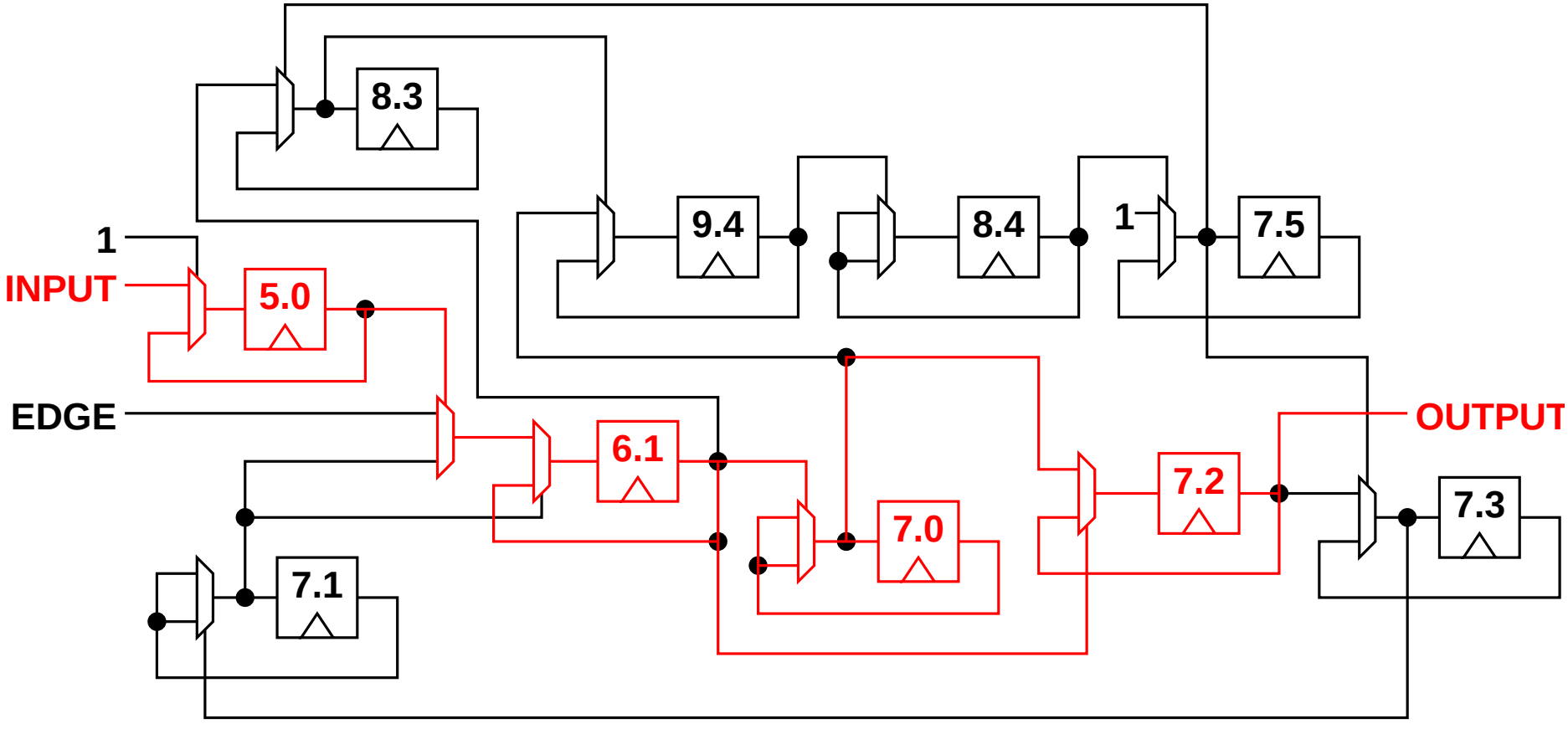
101 basic gates (inverters), 12 muxes, 10 latches (D flip-flop)



Core mechanism

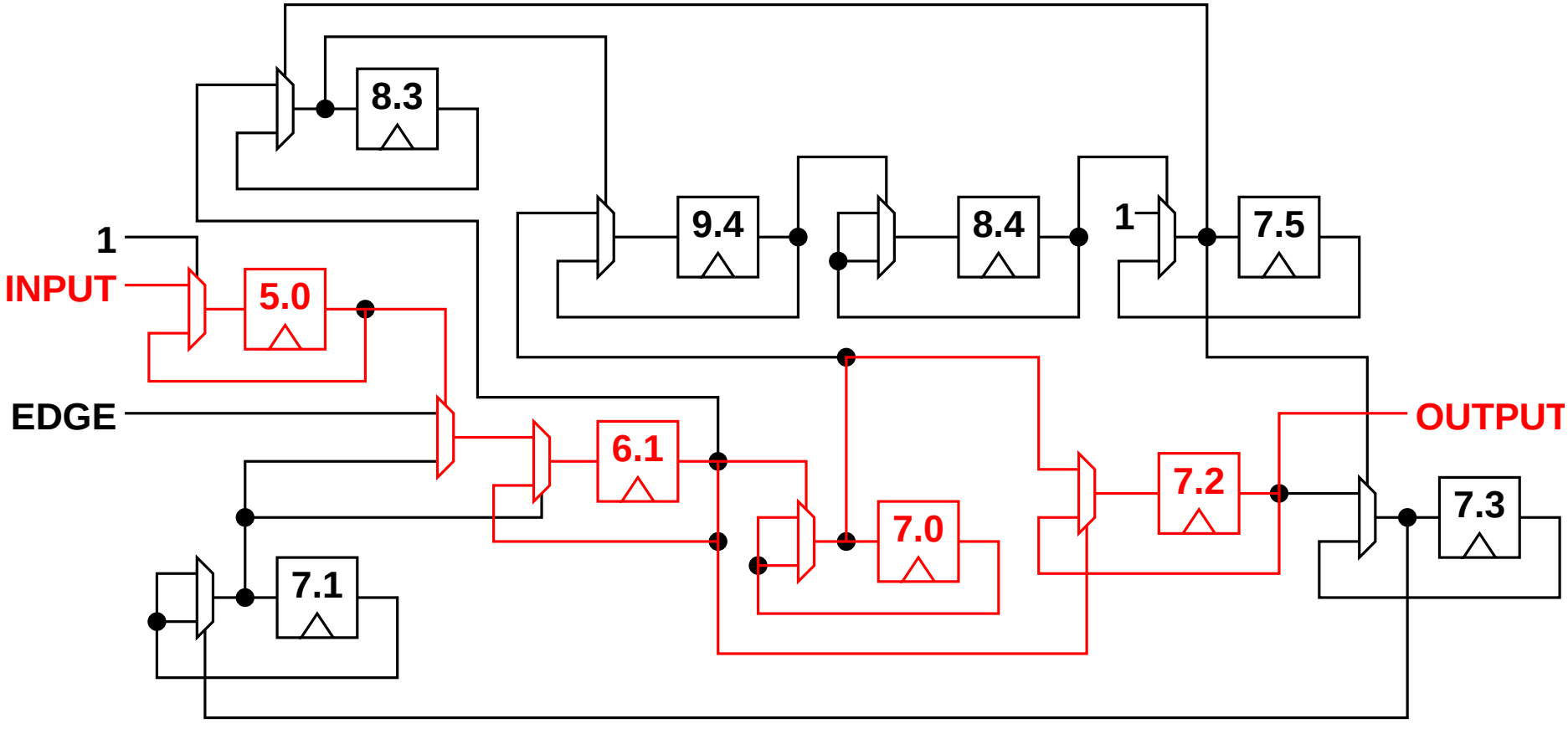


Core mechanism



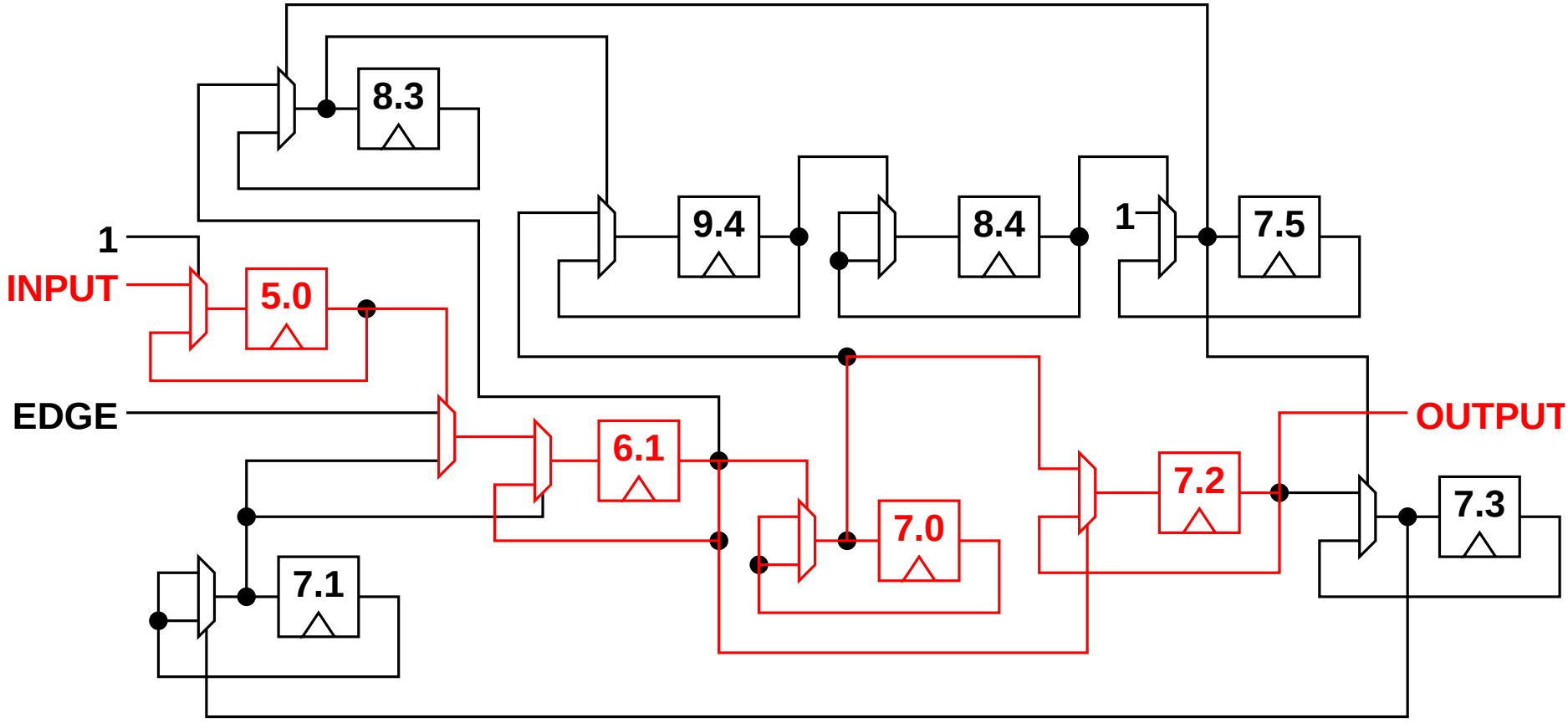
Only four cells are used: (5,0), (6,1), (7,0), and (7,2)

Core mechanism



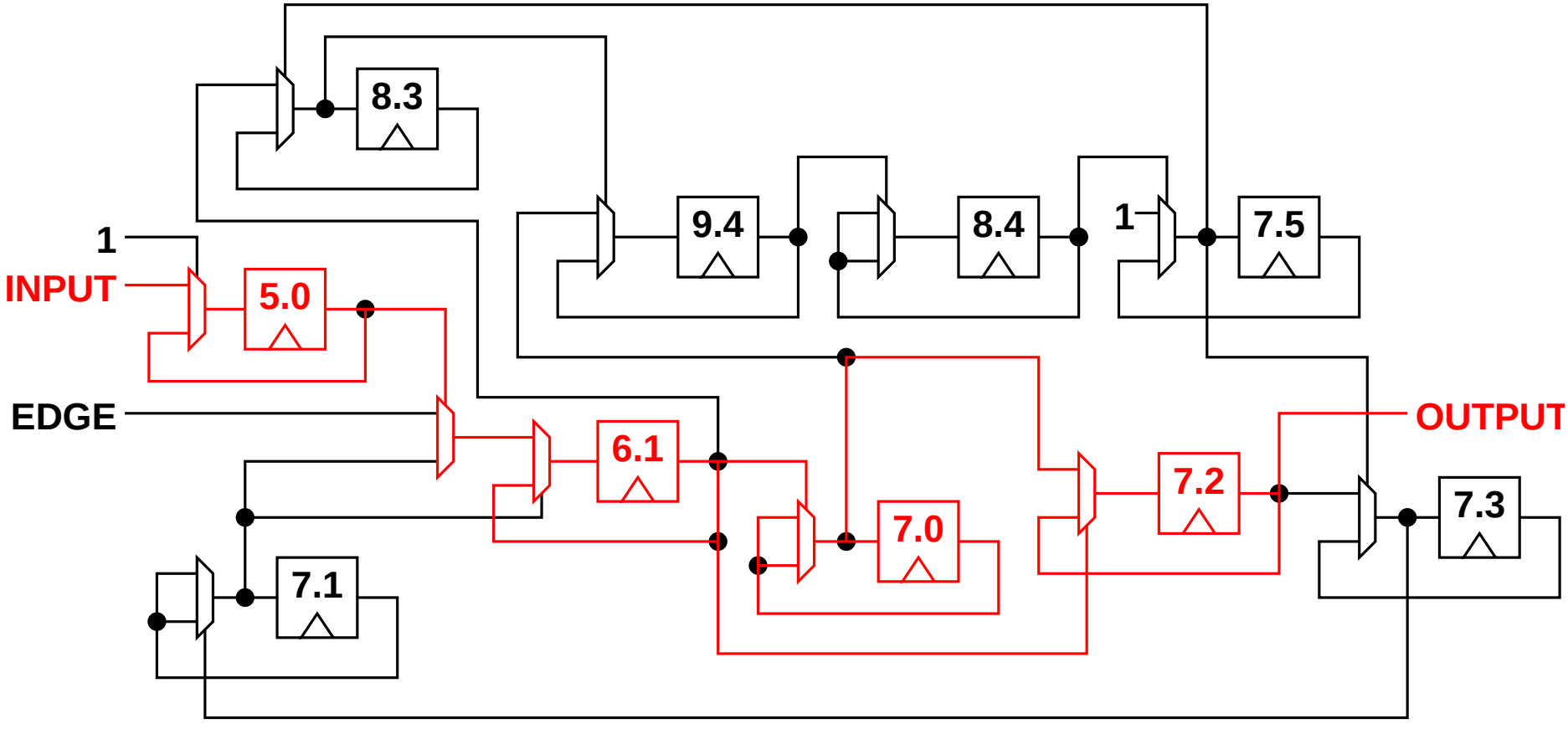
The input is first retimed in cell (5,0) to 6kHz

Core mechanism



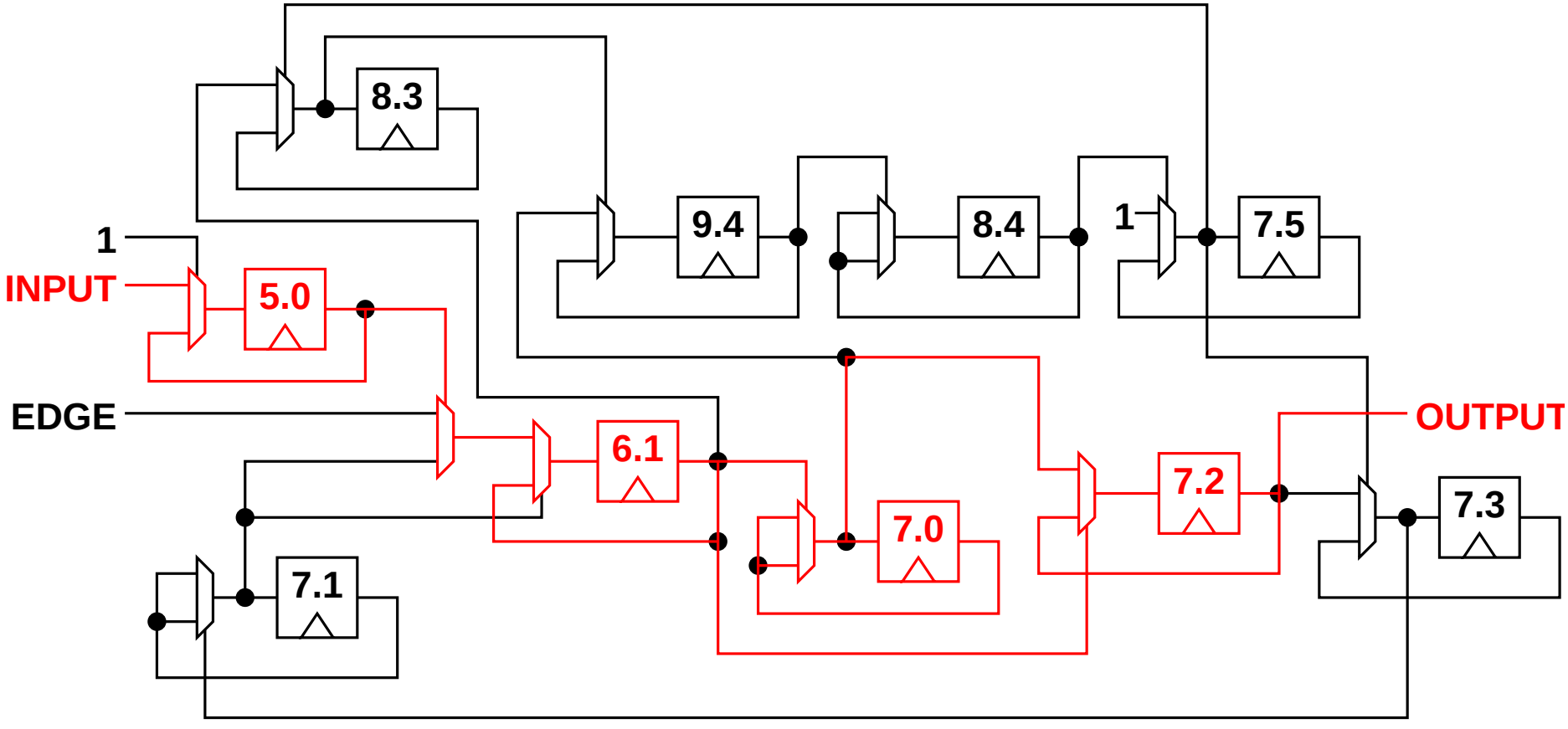
When the retimed input is high, cell (7,0) toggles at the clock frequency

Core mechanism



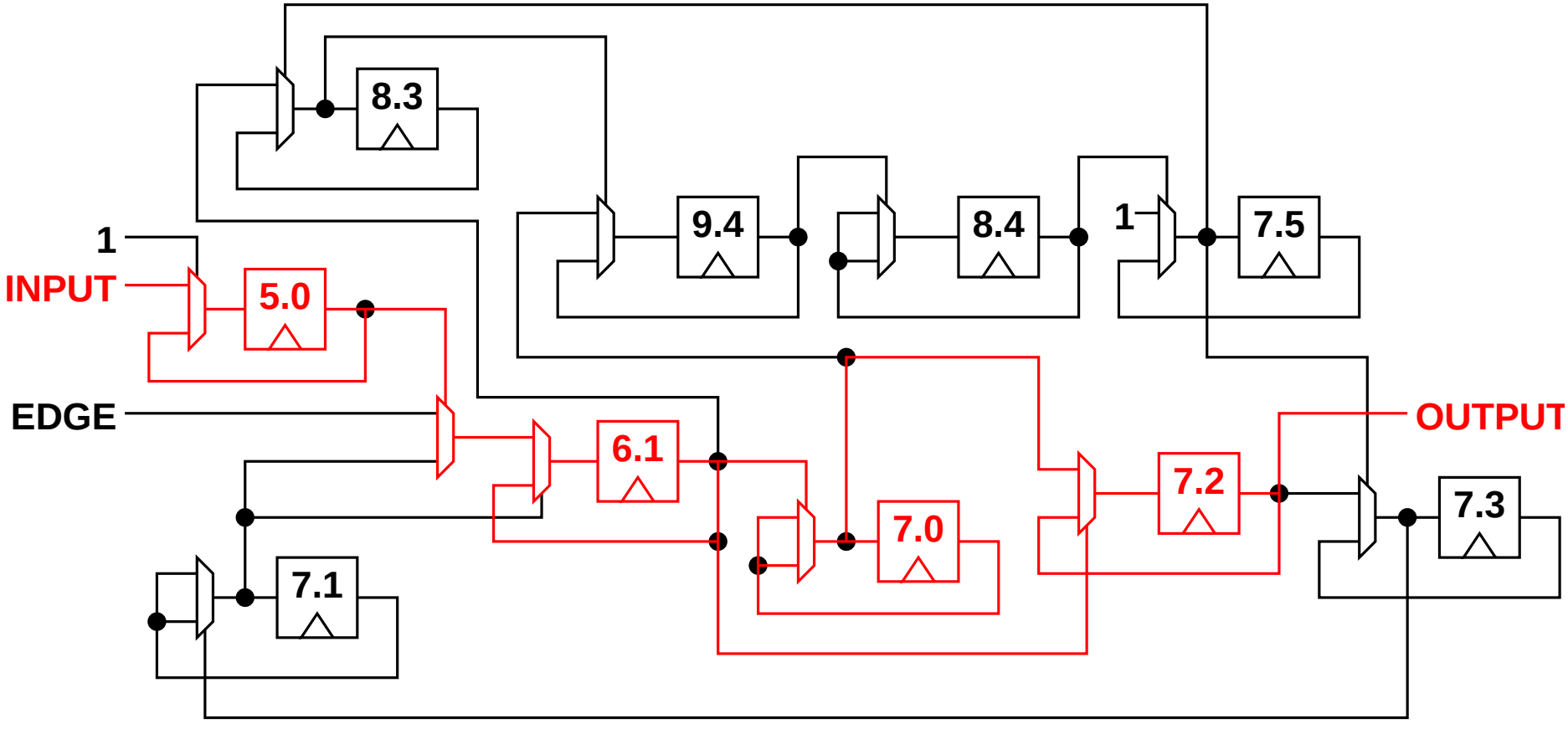
When the retimed input is low, this oscillation stops

Core mechanism



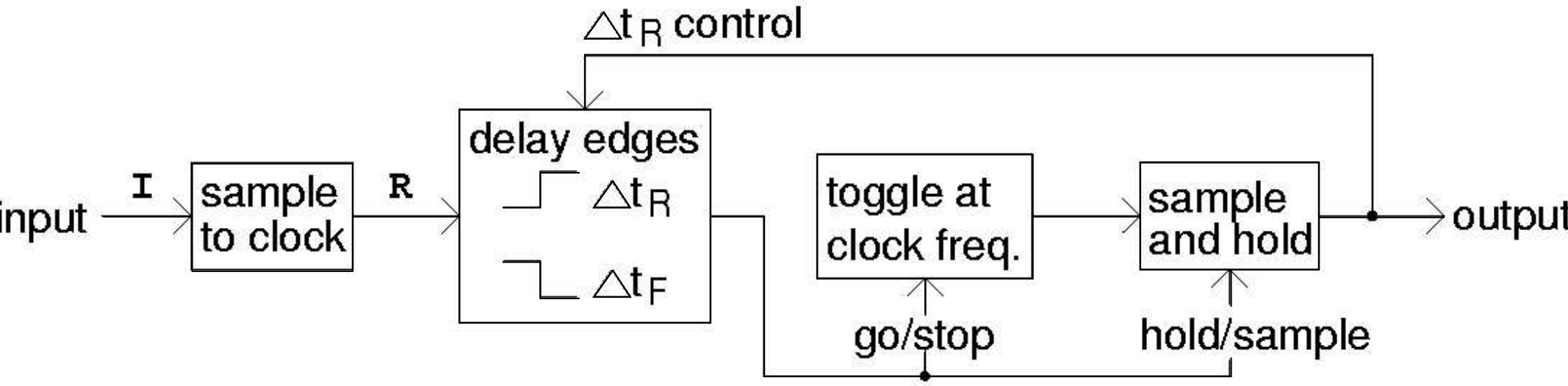
The number of times the oscillator toggles is completely determined by how long the raw input is high, and hence the input frequency

Core mechanism



Finally, cell (7,2) holds the final value of the previous oscillation while the next one is going on

Global mechanism

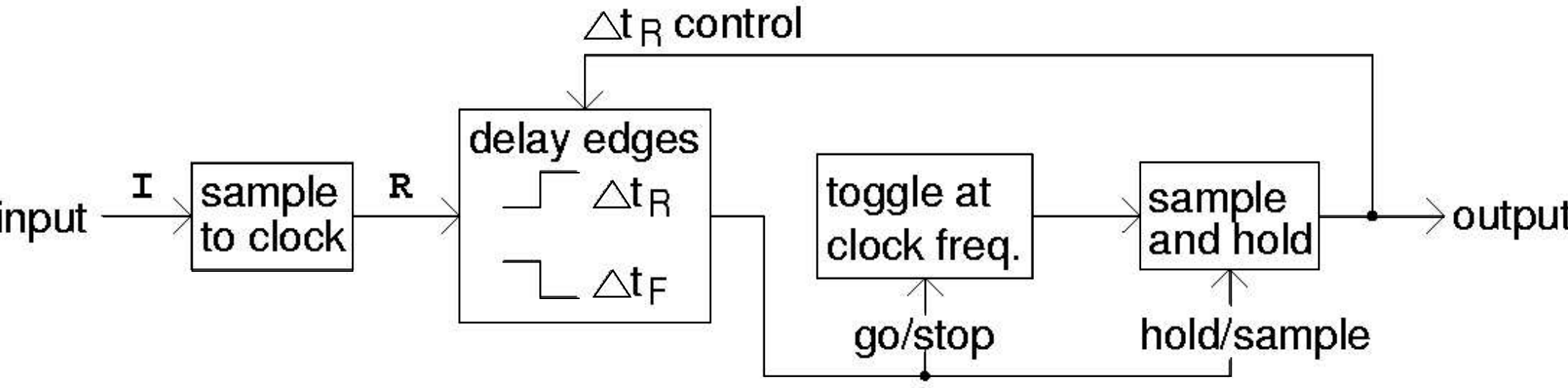


This core mechanism with cells (5,0), (6,1), (7,0), and (7,2) produces a constant output for one input frequency, and a toggling output for the other input frequency

The other cells in the circuit serve to delay the retimed input

This delay is constant for falling edges, but variable for rising edges, function of the current circuit output

Global mechanism



If the input is 1kHz while the output is high (or 10kHz/low),
odd togglings in a high half cycle of the input
➪ the output will change state

If the input is 1kHz while the output is low (or 10kHz/high),
even togglings in a high half cycle of the input
➪ the output will remain constant

"The implementation of the variable delay is not yet understood"

Results and conclusion

The final circuit works perfectly from -27°to 60°

When the input changes, the output changes after several cycles

Due to the stabilisation of the cycles involving latches

This digital circuit uses analogue time delays, which are avoided in digital design!

The evolution strategy does not care about design rules!