Politecnico di Torino

Specification and simulation of digital systems

Assignment discussion

Candidate:
Giuseppe Carella (231559)

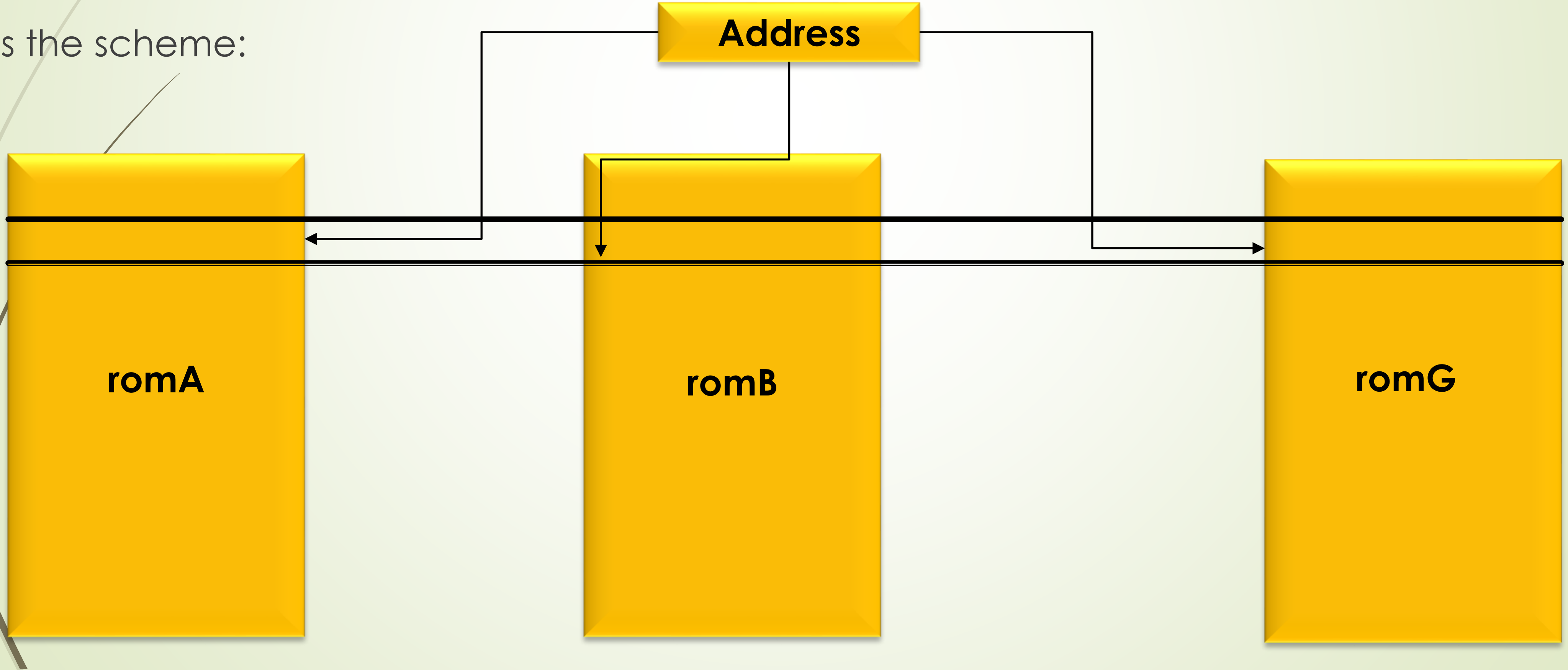
General information

- 1024 values for minuendo.
- 1024 values for subtracting.
- 1024 values for difference.
- Each value is on 16 bits.
- Clock period: 6.67 ns.
- Frequency: 150 Mhz.
- Number of clock cycle for each wrong value on "debug_port": 2 clock cycle.

The address bus is on 11 bits. I use 11 bits, because using 10 bits the maximum value is 1023. I need for a value bigger than 1023 to go out from the cycle.

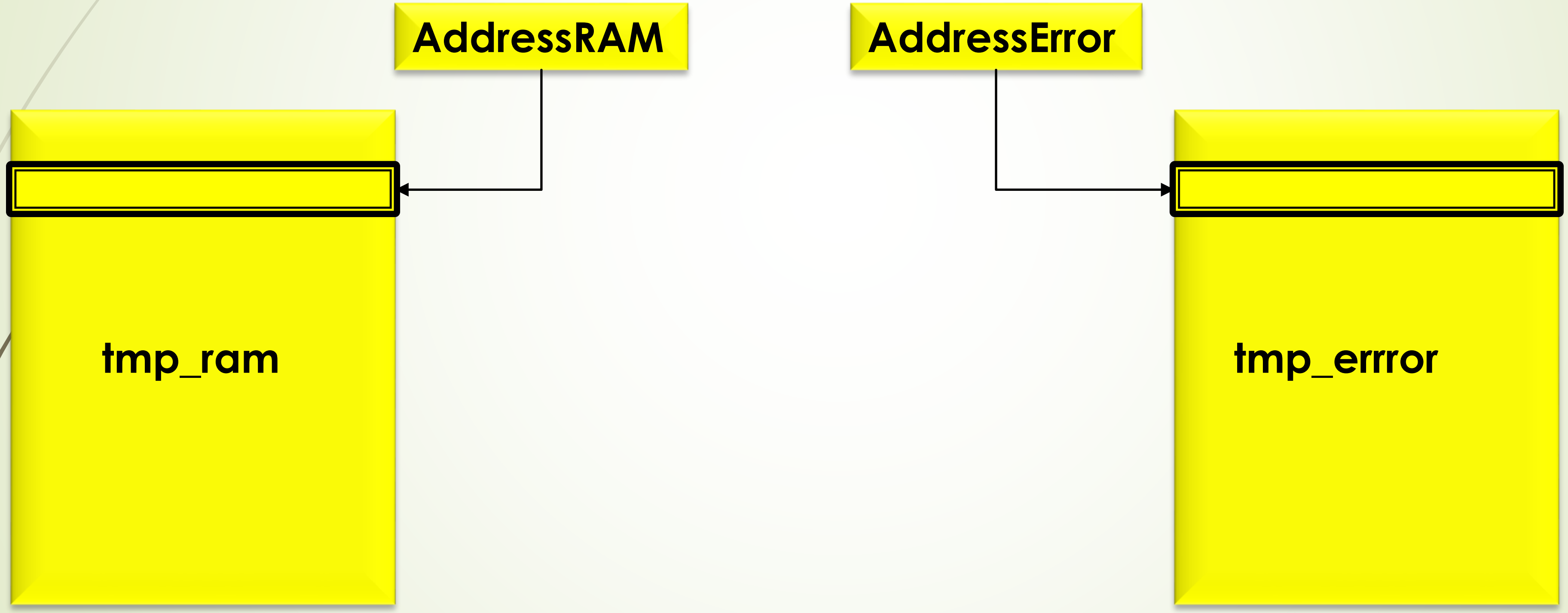
Structure of ROM

- It has been decided to use 3 ROM: two ROM for minuendo and subtracting and another one for difference. They have the same depth. It has been used three ROM, because in this way only one signal (Address) is used to store A, B and G. Using one ROM it makes the storing more difficult. The disadvantage is that it’s used more hardware (3 ROM instead 1).
- This is the scheme:



Structure of RAM

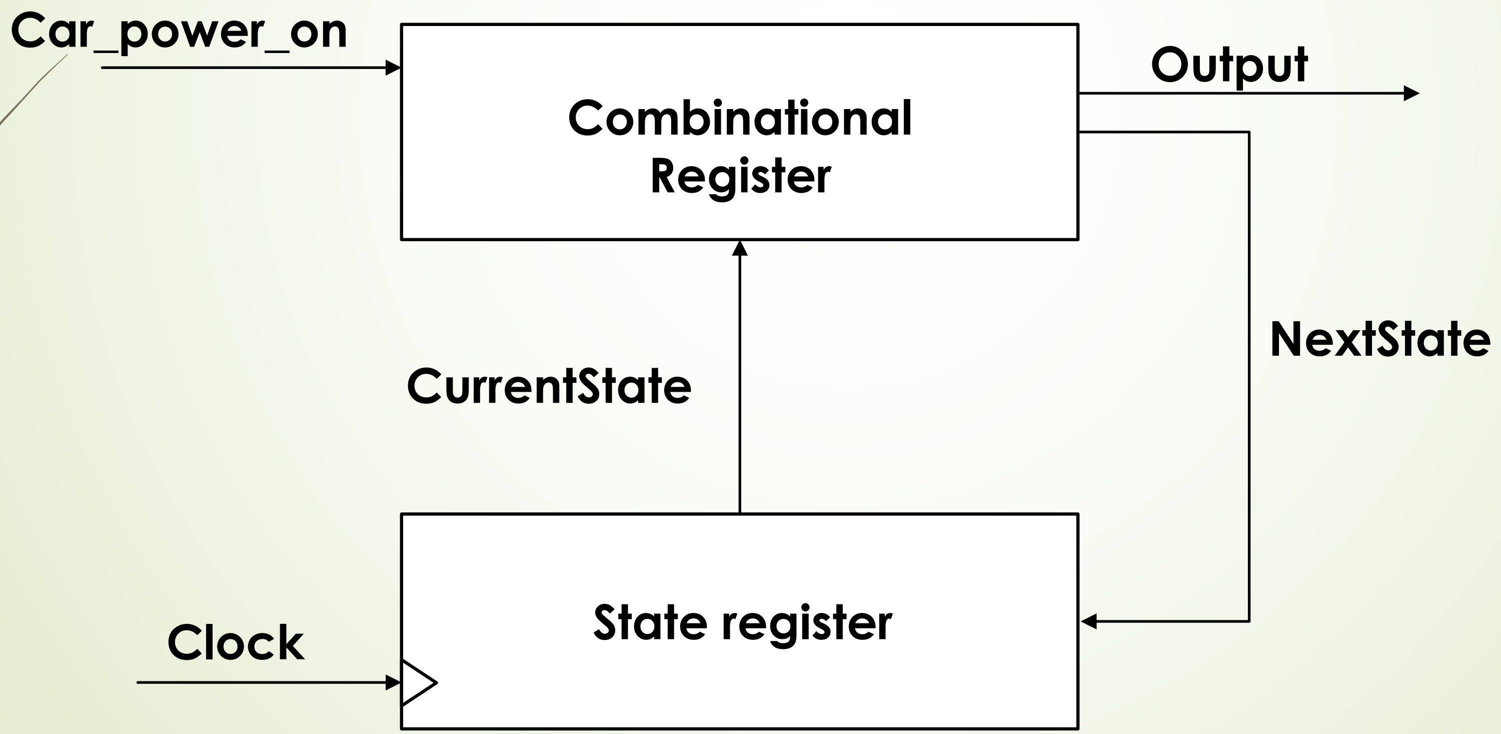
- It has been used two RAMs. The first one stores all differences. The second one stores the differences affected by error.



ECU

- It has been used two registers: combinational register and state register; therefore the ECU is a FSM on 3 bits (8 states).

This is the general scheme:



State register

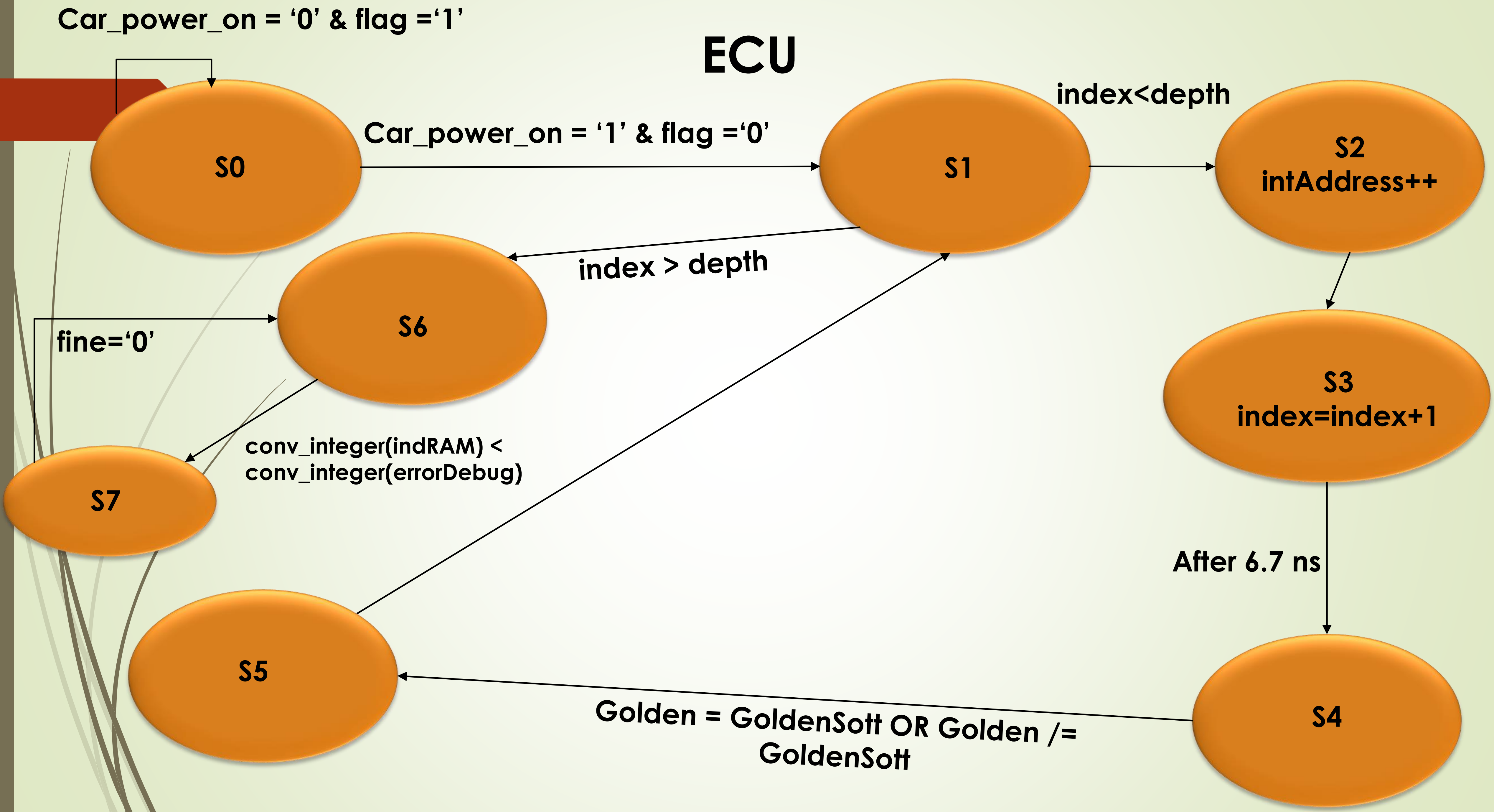
- It depends on signals “flag” and “Car_power_on”. The signal “flag” is used, because the signal “Car_power_on” is enabled for 2 us; therefore the signal “flag” is necessary when “Car_power_on” is disabled. See code to understand.

Combinational register

- **S0**: In this state variables and signal like “Address” and “ErrorDebug” have been initialized. Moreover S0 has been used to initialize the ROM using the signal “InitROM”.
- **S1**: It scans the ROM using the signals “index” and ”intAddress”. “index” is used to control the final condition of the cycle (index = depth means that all ROM values have been scanned), while “intAddress” is used to make access into ROM values.
- **S2**: In this state the signal “intAddress” has been increased.
- **S3**: It increases “index” and assigns the values coming from ROM into signals “OutA”,”OutB” and “OutG”. “OutA” and “OutB” (output of ECU) are the inputs of subtractor (component AddSub). S3 waits a few time (6.7 ns) before to go to the state S4 so as to wait for the AddSub result (GoldenSott).

- **S4**: It has been made a comparison between Golden (result coming from ROM) and GoldenSott (result coming from AddSub). If they're equals there is not any error (AssignResult <= '0'). If they're different "AssignResult <= '1'".
- **S5**: It depends on "AssignResult" value. If in S4 one error has been identified (AssignResult <= '1') the internal signal "errorDebug" increased and the writing of "tmp_ram" (WriteRAM <= '1') and "tmp_error" (ErrorRAM <= '1') are enabled.
- **S6:** This state depends on the condition "index > depth" into S1. First of all the number of errors has been stored into "debug_port". S6 controls if there are some errors or not. In this way this state decides to enable "ok_status" or "fault_status". If there are some errors, wrong values are read from "tmp_error" using the signal "indRAM" as address to make access in "tmp_error".
- **S7**: It assigns "inRAM" to "debug_port" until when "fine='0'".

ECU


Car_power_on = '0' & flag ='1'
S0
Car_power_on = '1' & flag ='0'
S1
index<depth
S2
intAddress++
S3
index=index+1
After 6.7 ns
S4
Golden = GoldenSott OR Golden /= GoldenSott
S5
index > depth
S6
fine='0'
conv_integer(indRAM) < conv_integer(errorDebug)
S7