A VARIABLE-SIZE SHIFT/ADD MULTIPLIER ARCHITECTURE

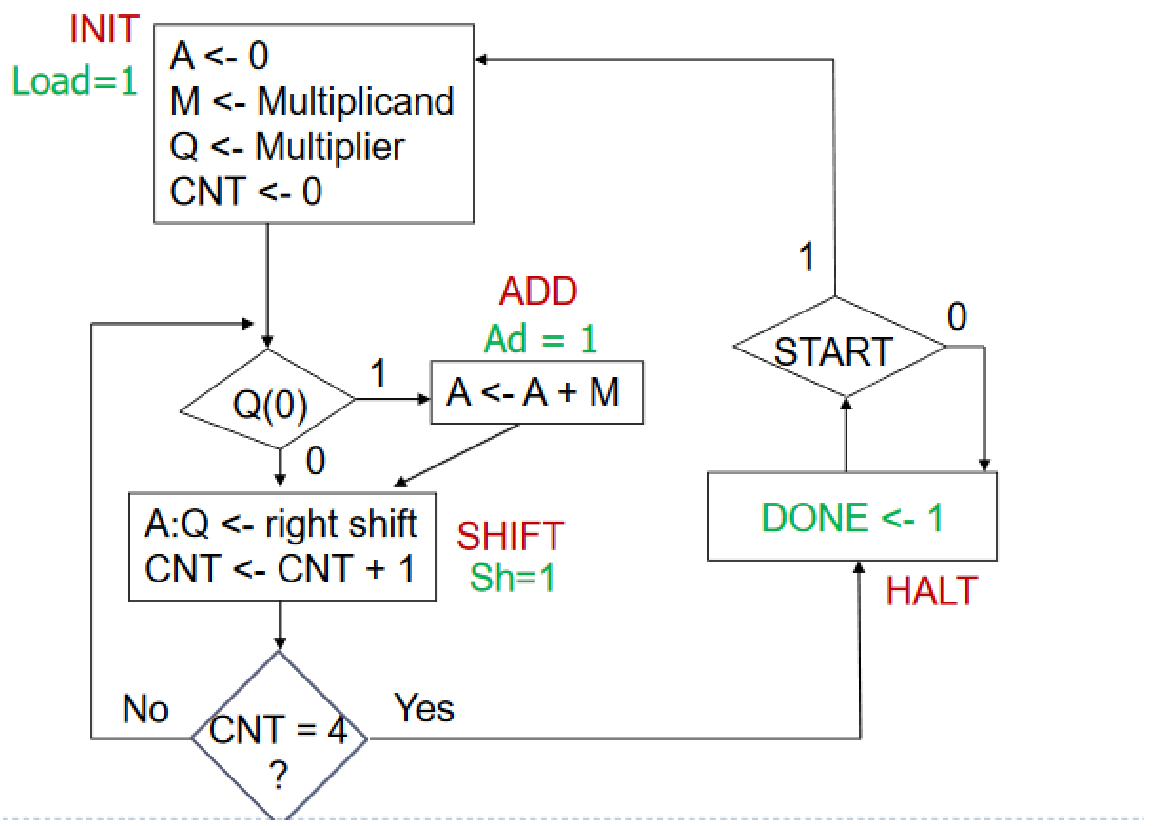
VLSI II – 2018/2019 – TEAM 8

EVANGELOU GEORGIOS

NIKOLAOS ROUSSOS

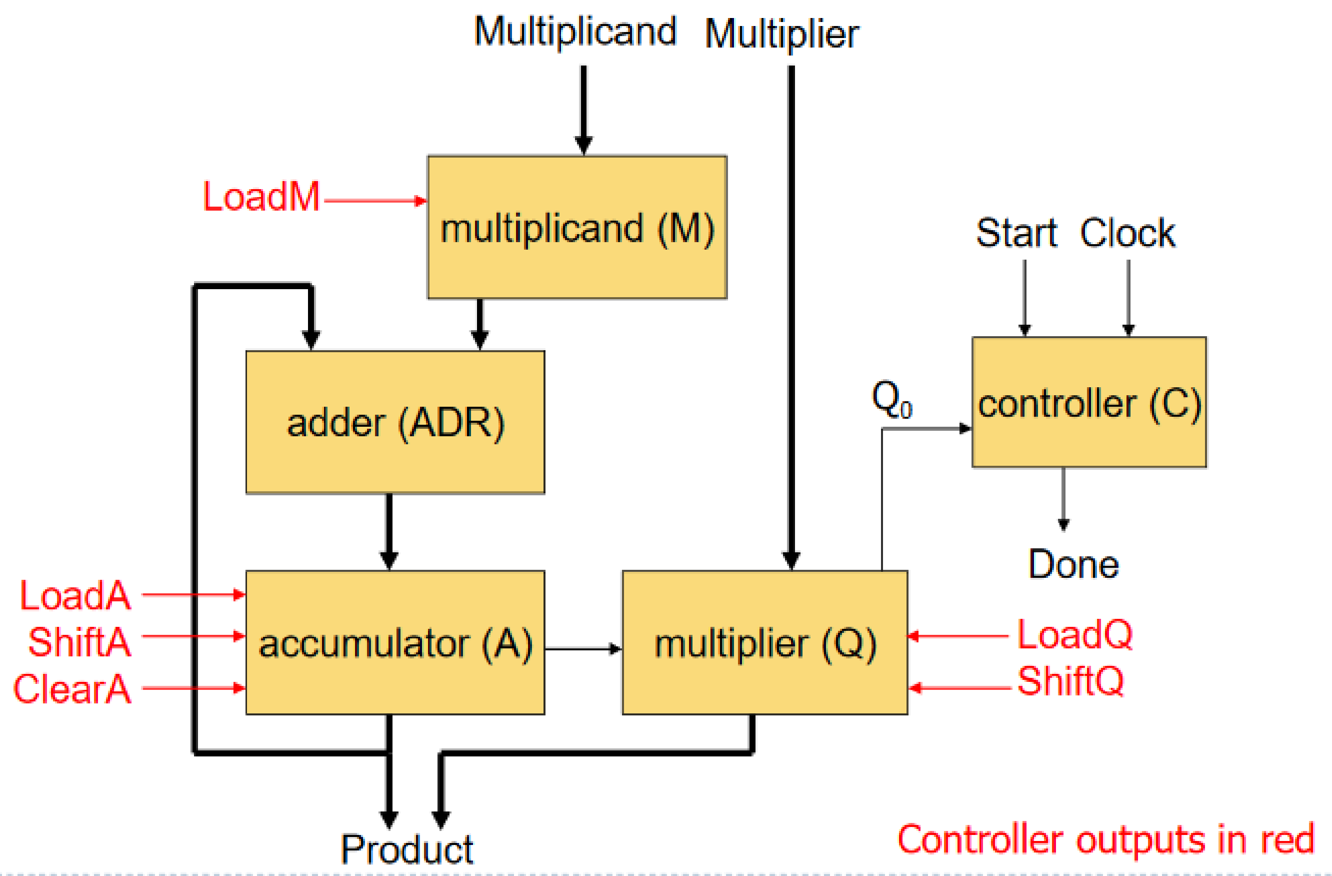
THE IMPLEMENTED ALGORITHM

“Add and shift” multiply algorithm (Moore model)

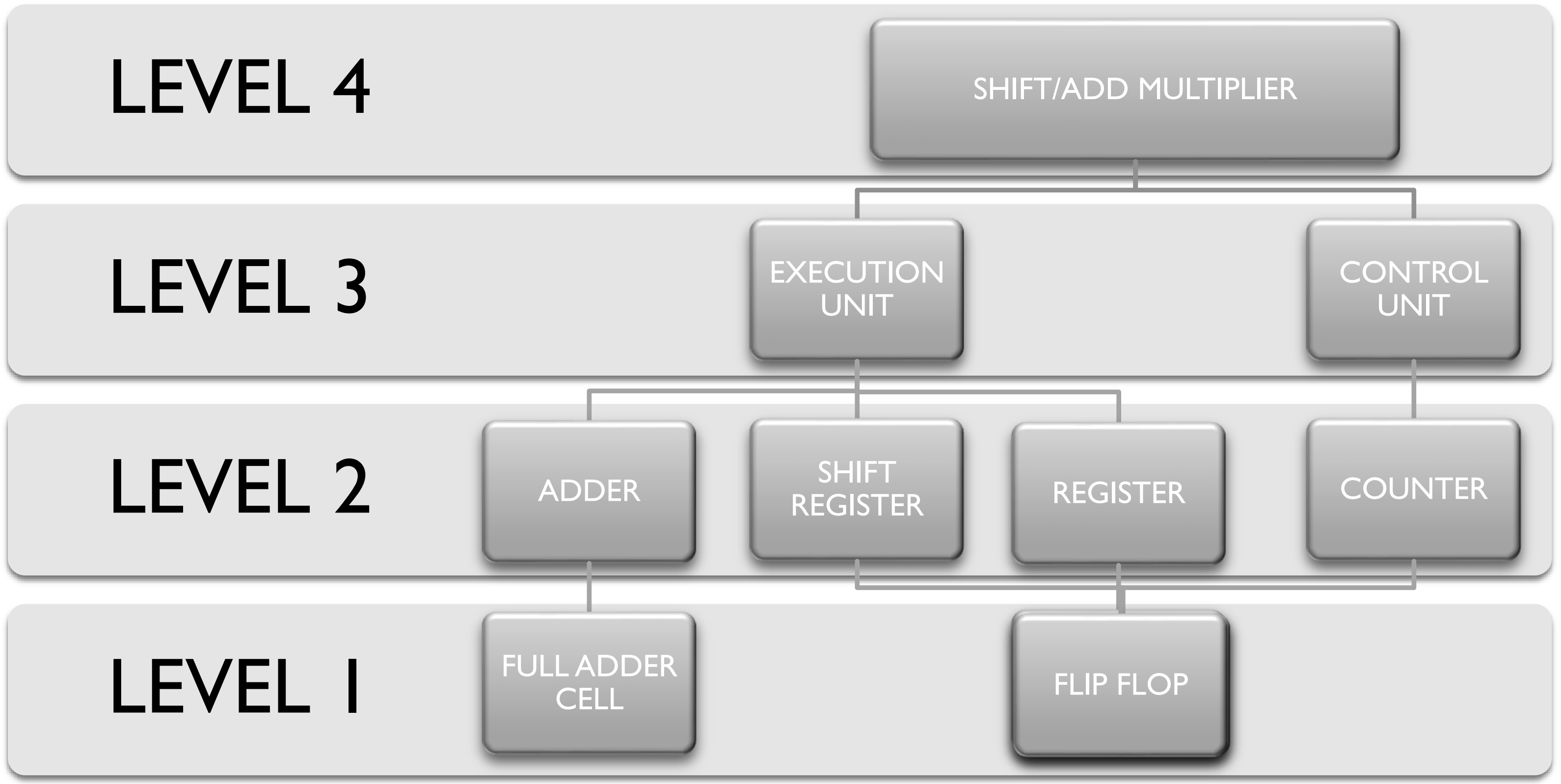


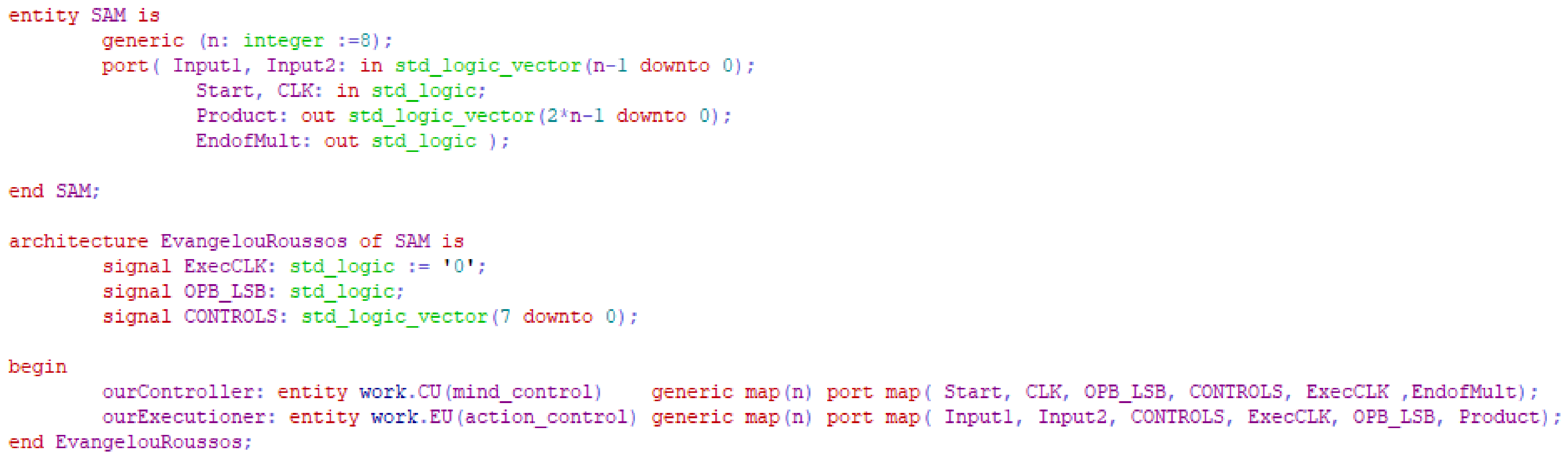
THE MAIN COMPONENTS

System Example: 8x8 multiplier



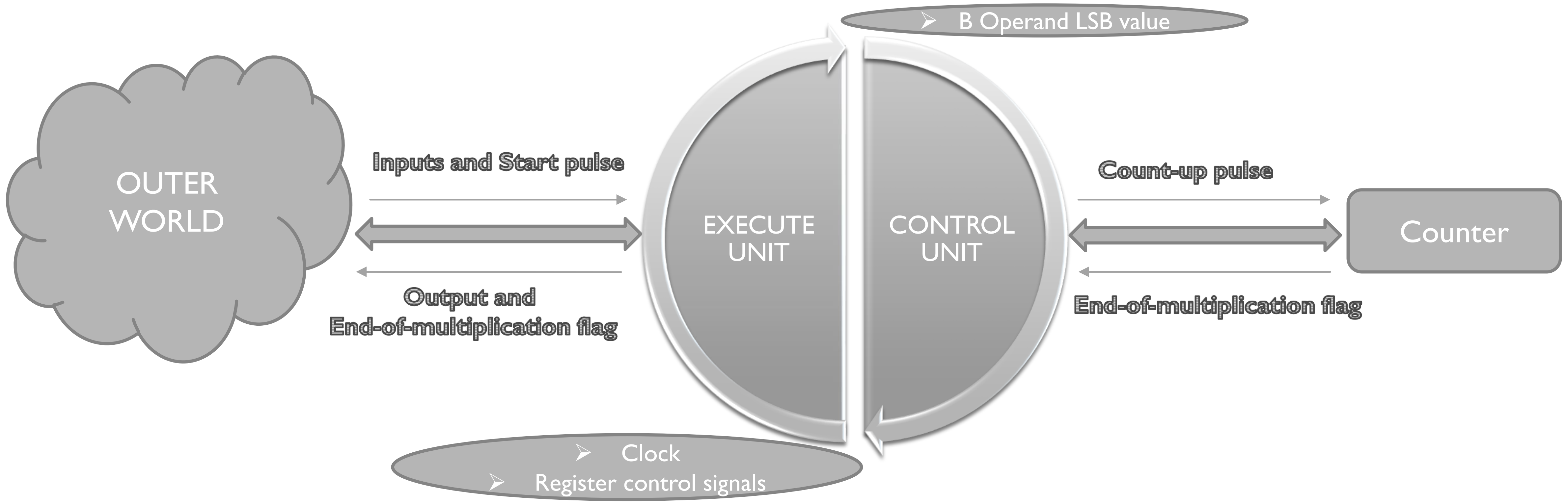
OUR BOTTOM-UP ARCHITECTURE



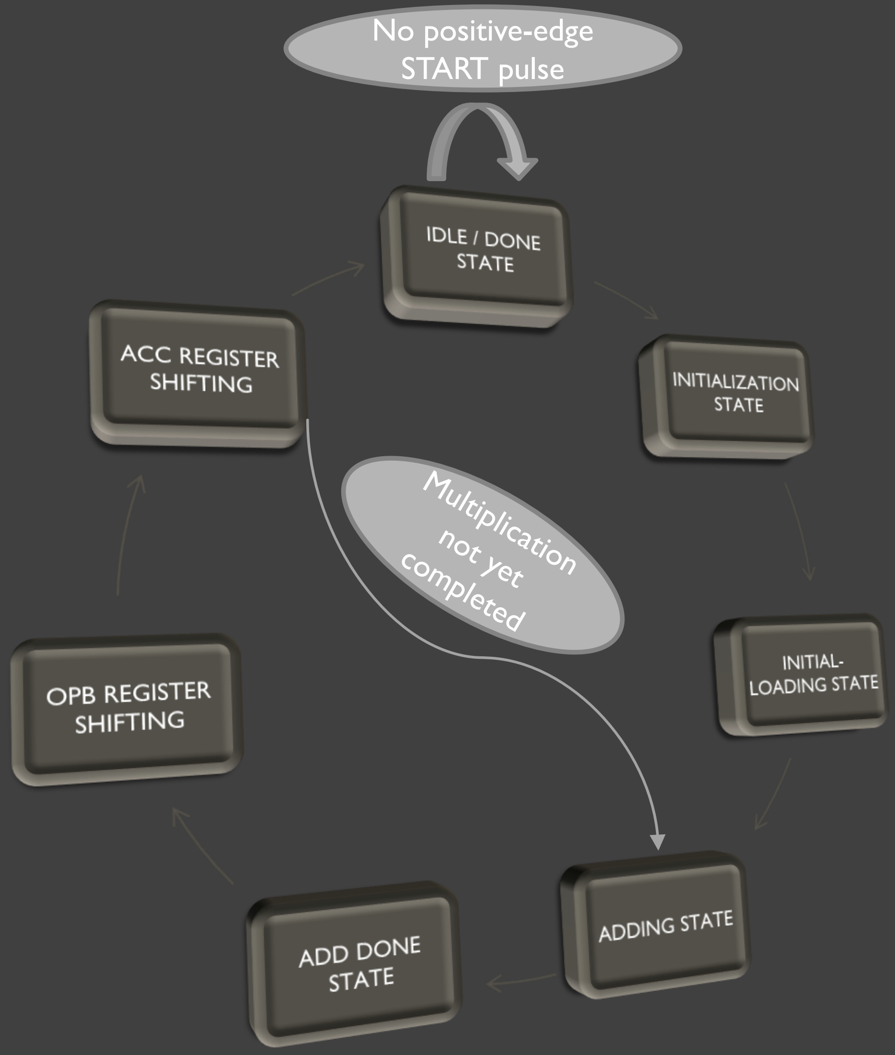


```vhdl
entity SAM is
        generic (n: integer :=8);
        port( Input1, Input2: in std_logic_vector(n-1 downto 0);
                Start, CLK: in std_logic;
                Product: out std_logic_vector(2*n-1 downto 0);
                EndofMult: out std_logic );

end SAM;

architecture EvangelouRoussos of SAM is
        signal ExecCLK: std_logic := '0';
        signal OPB_LSB: std_logic;
        signal CONTROLS: std_logic_vector(7 downto 0);

begin
        ourController: entity work.CU(mind_control)    generic map(n) port map( Start, CLK, OPB_LSB, CONTROLS, ExecCLK ,EndofMult);
        ourExecutioner: entity work.EU(action_control) generic map(n) port map( Input1, Input2, CONTROLS, ExecCLK, OPB_LSB, Product);
end EvangelouRoussos;
```

LEVEL 4 IMPLEMENTATION



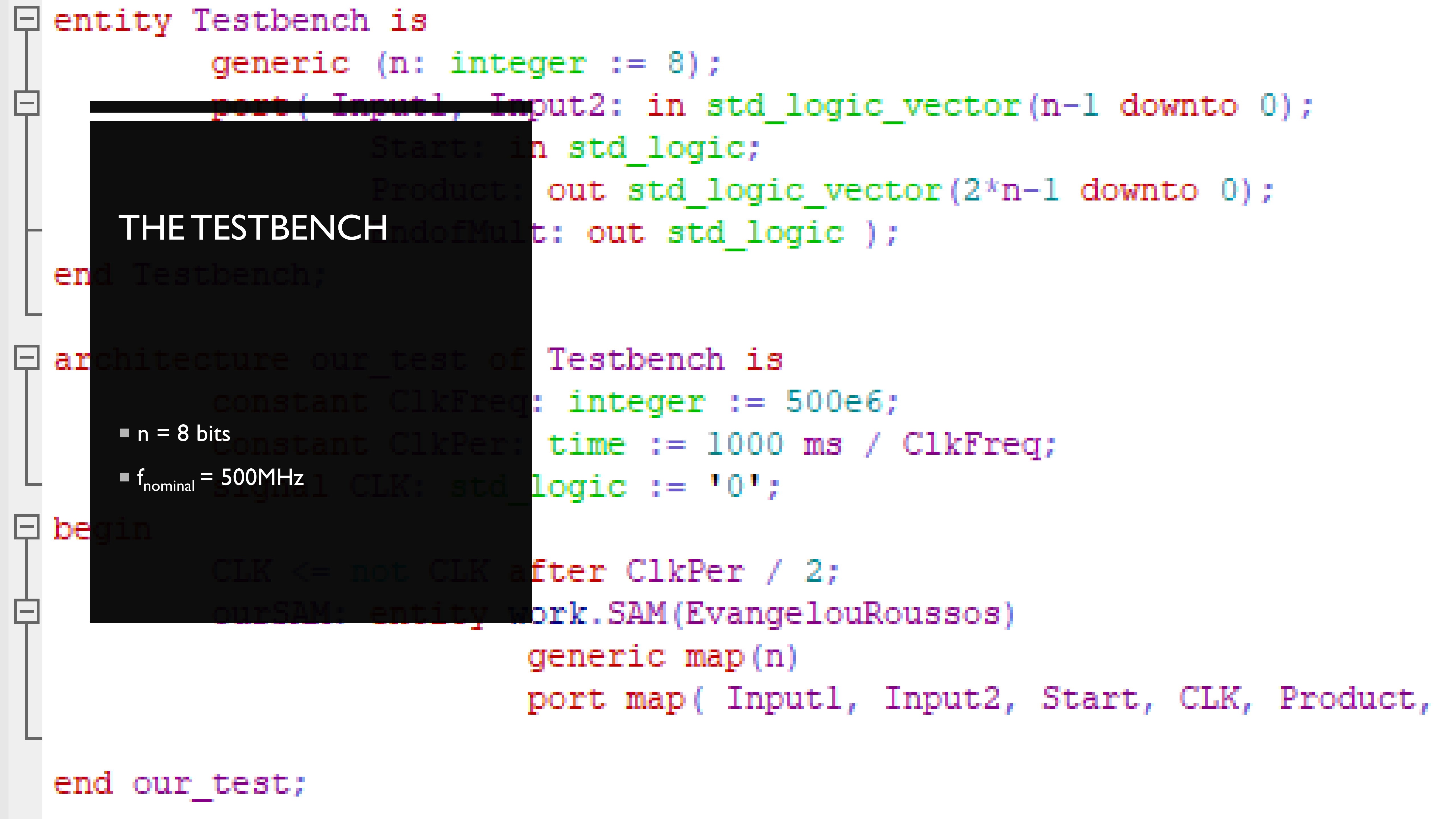
UNIT COMMUNICATION



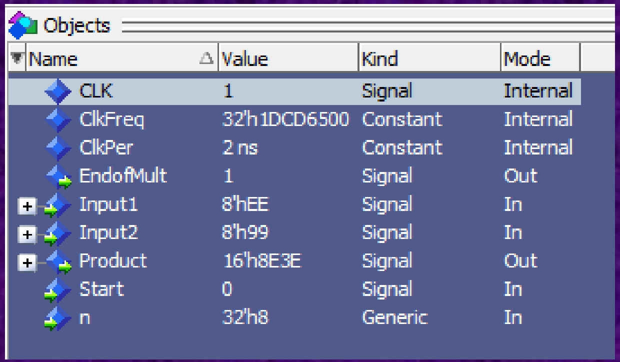
OUR SYSTEM AS A FINITE-STATE MACHINE

THE TESTBENCH

- n = 8 bits
- $f_{nominal}$ = 500MHz

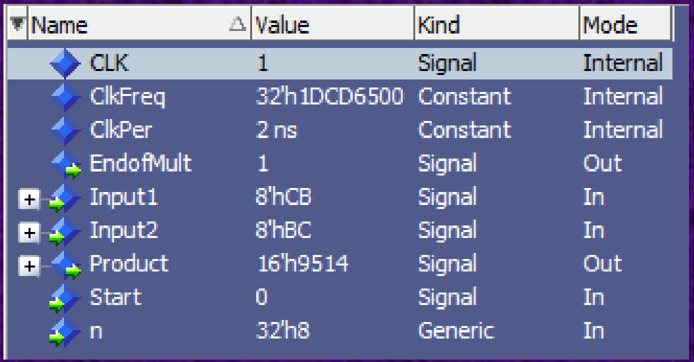


```vhdl
entity Testbench is
        generic (n: integer := 8);
        port( Input1, Input2: in std_logic_vector(n-1 downto 0);
                Start: in std_logic;
                Product: out std_logic_vector(2*n-1 downto 0);
                EndofMult: out std_logic );
end Testbench;

architecture our_test of Testbench is
        constant ClkFreq: integer := 500e6;
        constant ClkPer: time := 1000 ms / ClkFreq;
        signal CLK: std_logic := '0';

begin
        CLK <= not CLK after ClkPer / 2;
        ourSAM: entity work.SAM(EvangelouRoussos)
                        generic map(n)
                        port map( Input1, Input2, Start, CLK, Product,

end our_test;
```

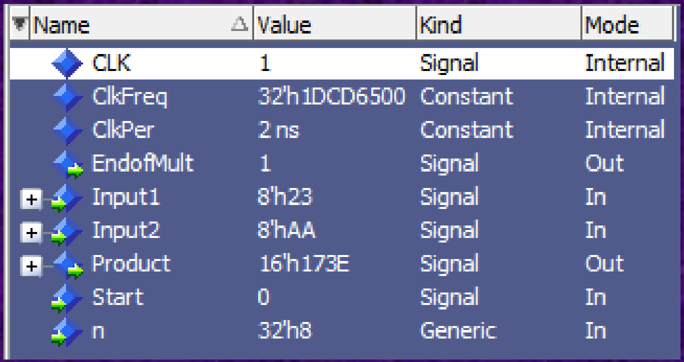


Objects

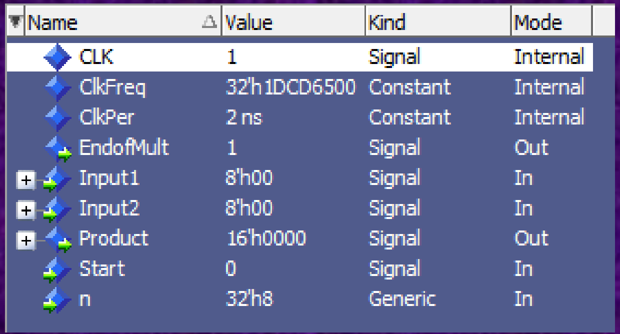
Name	Value	Kind	Mode
CLK	1	Signal	Internal
ClkFreq	32'h1DCD6500	Constant	Internal
ClkPer	2 ns	Constant	Internal
EndofMult	1	Signal	Out
Input1	8'hEE	Signal	In
Input2	8'h99	Signal	In
Product	16'h8E3E	Signal	Out
Start	0	Signal	In
n	32'h8	Generic	In



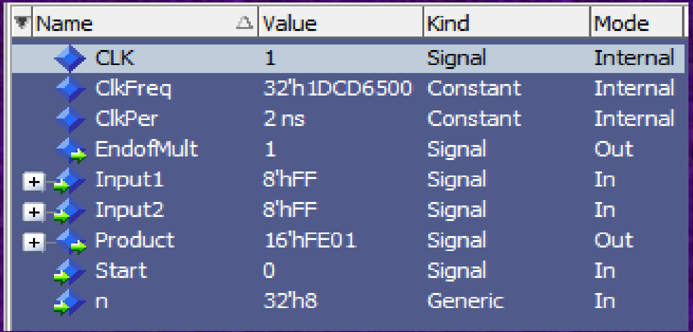
Name	Value	Kind	Mode
CLK	1	Signal	Internal
ClkFreq	32'h1DCD6500	Constant	Internal
ClkPer	2 ns	Constant	Internal
EndofMult	1	Signal	Out
Input1	8'hCB	Signal	In
Input2	8'hBC	Signal	In
Product	16'h9514	Signal	Out
Start	0	Signal	In
n	32'h8	Generic	In



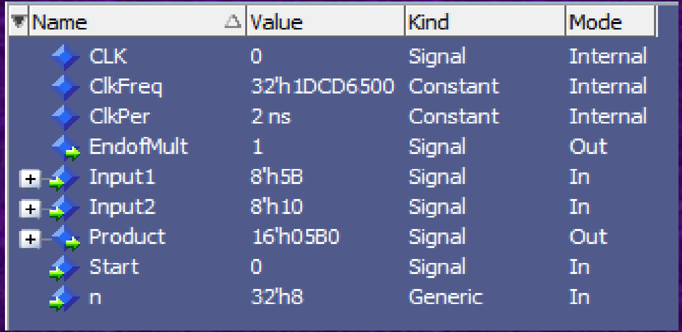
Name	Value	Kind	Mode
CLK	1	Signal	Internal
ClkFreq	32'h1DCD6500	Constant	Internal
ClkPer	2 ns	Constant	Internal
EndofMult	1	Signal	Out
Input1	8'h23	Signal	In
Input2	8'hAA	Signal	In
Product	16'h173E	Signal	Out
Start	0	Signal	In
n	32'h8	Generic	In



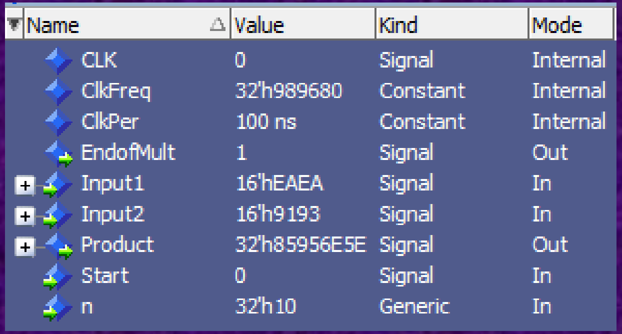
Name	Value	Kind	Mode
CLK	1	Signal	Internal
ClkFreq	32'h1DCD6500	Constant	Internal
ClkPer	2 ns	Constant	Internal
EndofMult	1	Signal	Out
Input1	8'h00	Signal	In
Input2	8'h00	Signal	In
Product	16'h0000	Signal	Out
Start	0	Signal	In
n	32'h8	Generic	In



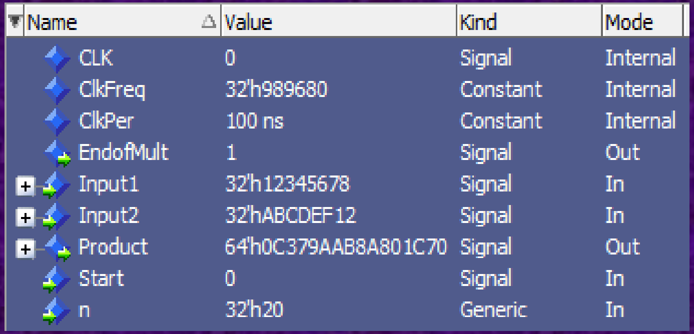
Name	Value	Kind	Mode
CLK	1	Signal	Internal
ClkFreq	32'h1DCD6500	Constant	Internal
ClkPer	2 ns	Constant	Internal
EndofMult	1	Signal	Out
Input1	8'hFF	Signal	In
Input2	8'hFF	Signal	In
Product	16'hFE01	Signal	Out
Start	0	Signal	In
n	32'h8	Generic	In



Name	Value	Kind	Mode
CLK	0	Signal	Internal
ClkFreq	32'h1DCD6500	Constant	Internal
ClkPer	2 ns	Constant	Internal
EndofMult	1	Signal	Out
Input1	8'h5B	Signal	In
Input2	8'h10	Signal	In
Product	16'h05B0	Signal	Out
Start	0	Signal	In
n	32'h8	Generic	In



Name	Value	Kind	Mode
CLK	0	Signal	Internal
ClkFreq	32'h989680	Constant	Internal
ClkPer	100 ns	Constant	Internal
EndofMult	1	Signal	Out
Input1	16'hEAEA	Signal	In
Input2	16'h9193	Signal	In
Product	32'h85956E5E	Signal	Out
Start	0	Signal	In
n	32'h10	Generic	In



Name	Value	Kind	Mode
CLK	0	Signal	Internal
ClkFreq	32'h989680	Constant	Internal
ClkPer	100 ns	Constant	Internal
EndofMult	1	Signal	Out
Input1	32'h12345678	Signal	In
Input2	32'hABCDEF12	Signal	In
Product	64'h0C379AAB8A801C70	Signal	Out
Start	0	Signal	In
n	32'h20	Generic	In

FURTHER IMPROVEMENTS:

Blocked Carry Select Adder for higher number of bits

Unification of registers ACC and OPB, in order to execute shifting in one clock cycle instead of two

Conditional Execution of Addition clock cycles depending on operator's B least significant bit

Add extra Addition clock cycles and increase clock frequency