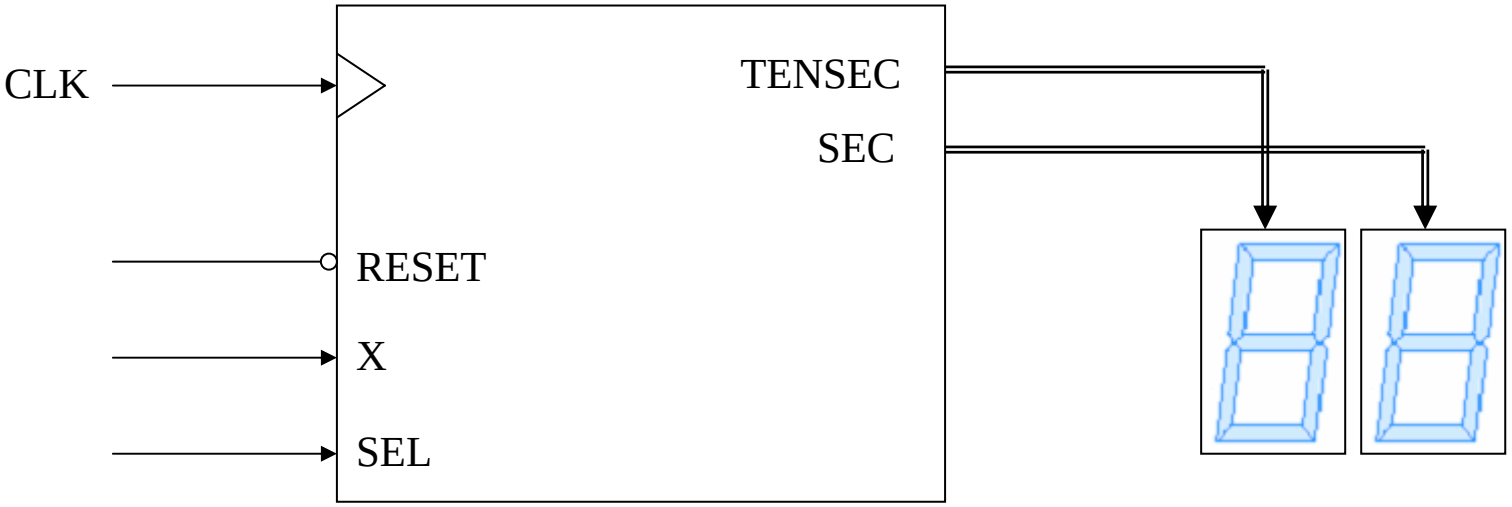
Stop Watch
(System Controller Approach)

Problem

- Design a “stop watch” that can measure times taken for two events
  - Inputs
    - CLK = 16 Hz
    - RESET: Asynchronously reset everything
    - X: comes from push button
      - First Push: Start timer
      - Second Push: Store the time taken for the first event
      - Third Push: Store the time taken for the second event
    - SEL: select output (High: first event time, Low: second event time)
  - Outputs
    - Two decimal digits to be connected to two seven segment displays
      - Can display 00 ~ 99

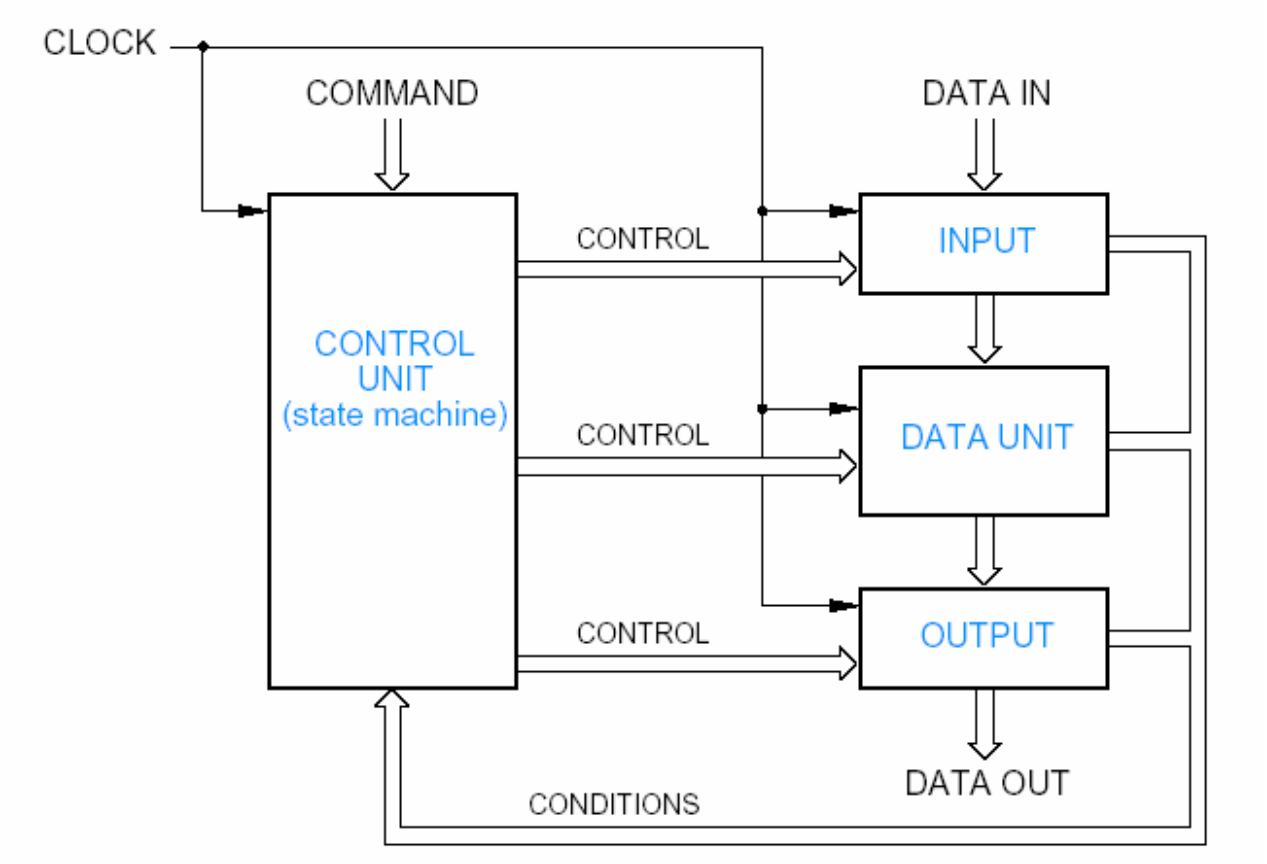
Overall Architecture Design

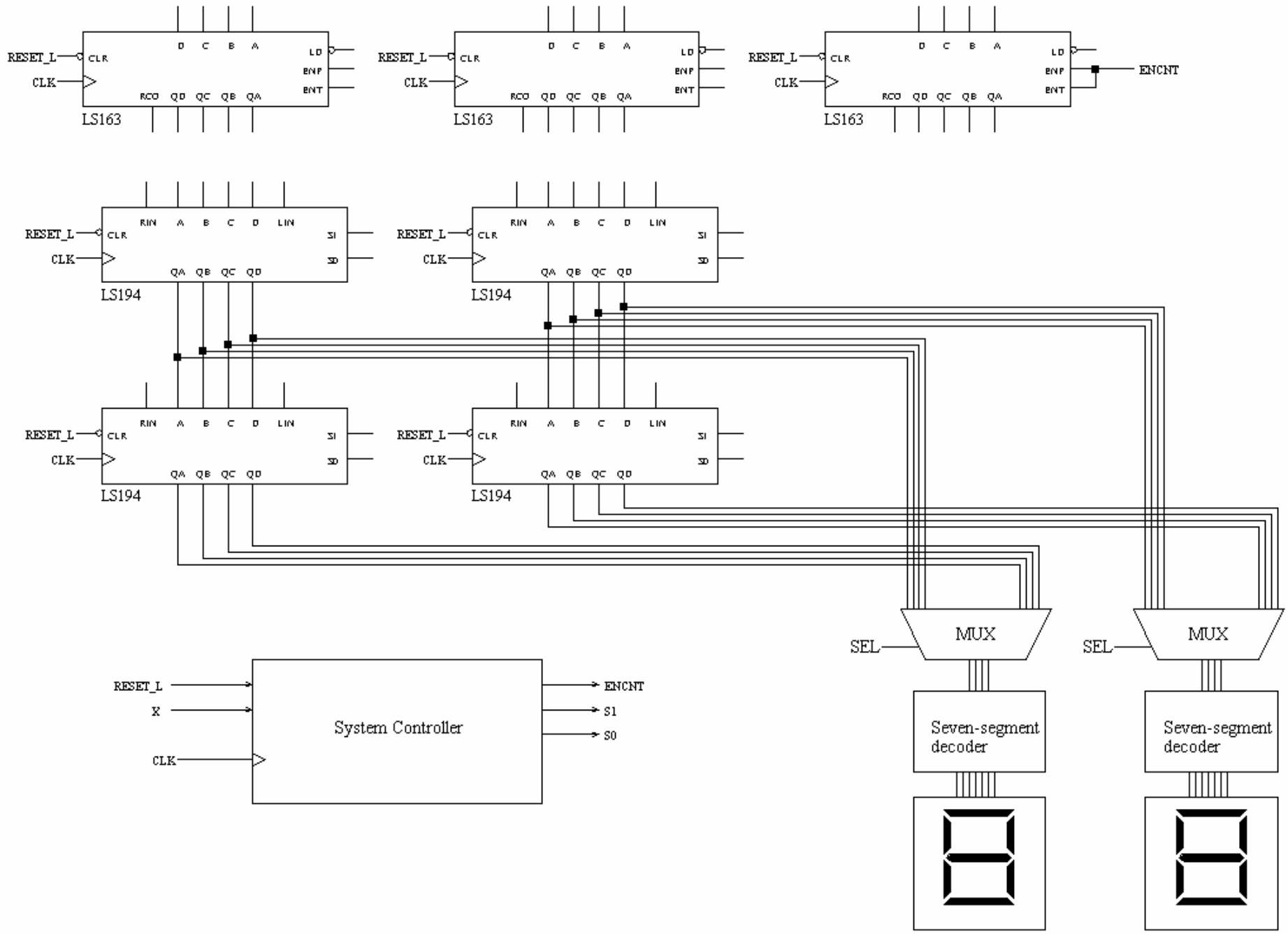


- Which LBB (among those we learn in the class) and how many?
- How to decompose the whole system into data part and control part?

Design System Architecture

- First step: divide the system into a control unit and data unit
  - Data unit – stores, routes, combines, and generally process data
  - Control unit – starting & stopping the process, testing conditions, and deciding what to do next



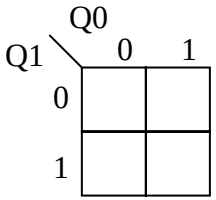

RESET_L
CLK
CLR
D C B A
LD
ENP
ENT
RCO QD QC QB QA
LS163
ENCNT
RIN A B C D LIN
SI
SD
QA QB QC QD
LS194
MUX
SEL
Seven-segment decoder
System Controller
X
S1
S0

System controller design

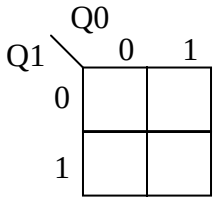
Table (Variable Entered Table)

State symbol	P.S. Q1 Q0	N.S. D1 D0	Output S1 S0 ENCNT
A	0 0		
B	0 1		
C	1 1		
unused	1 0		

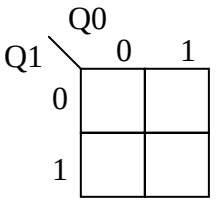
Excitation Eqs, Output Eqs



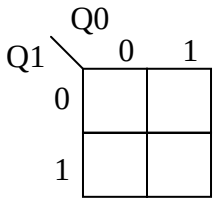
D1=



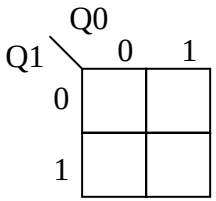
D0=



S1=

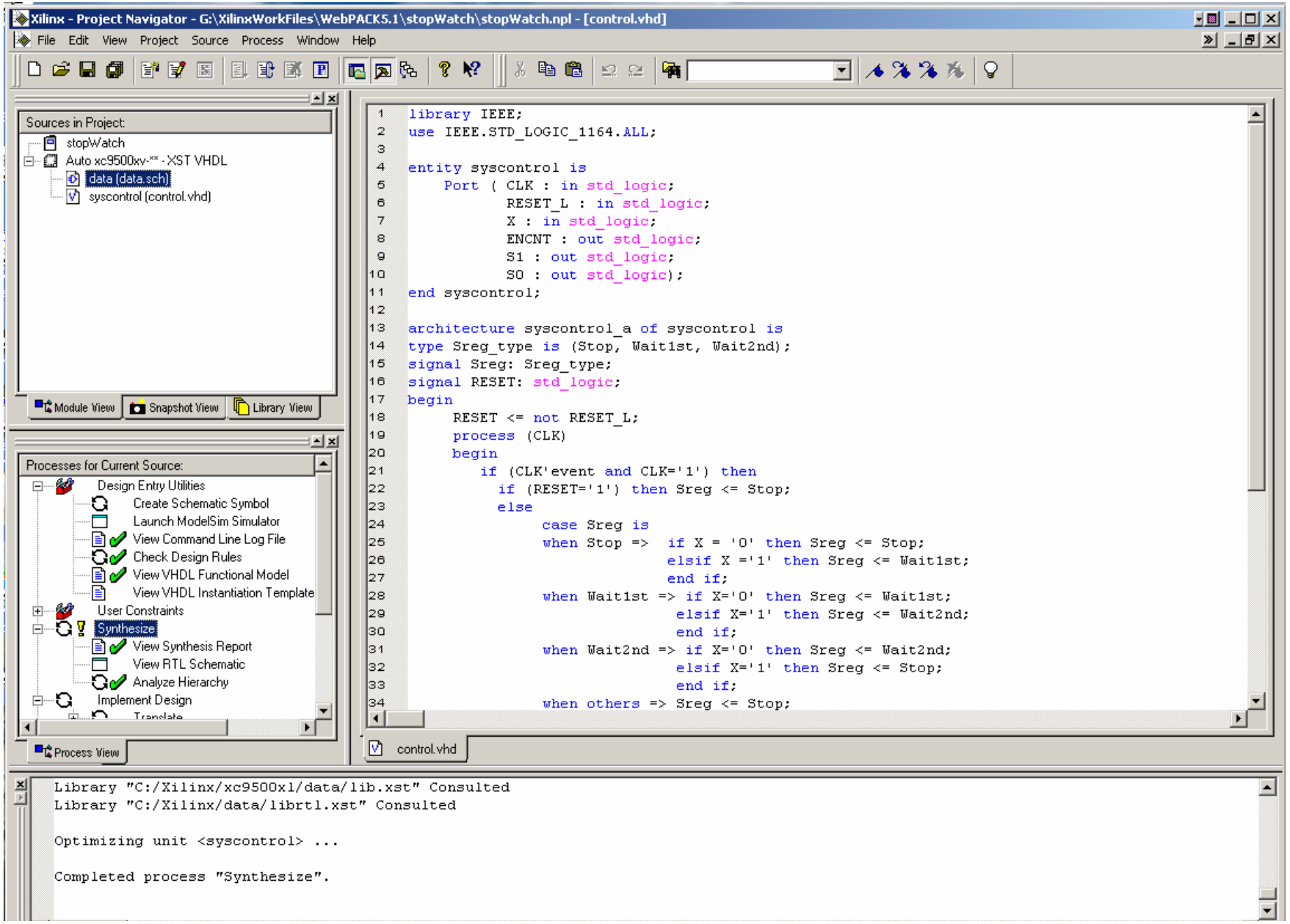


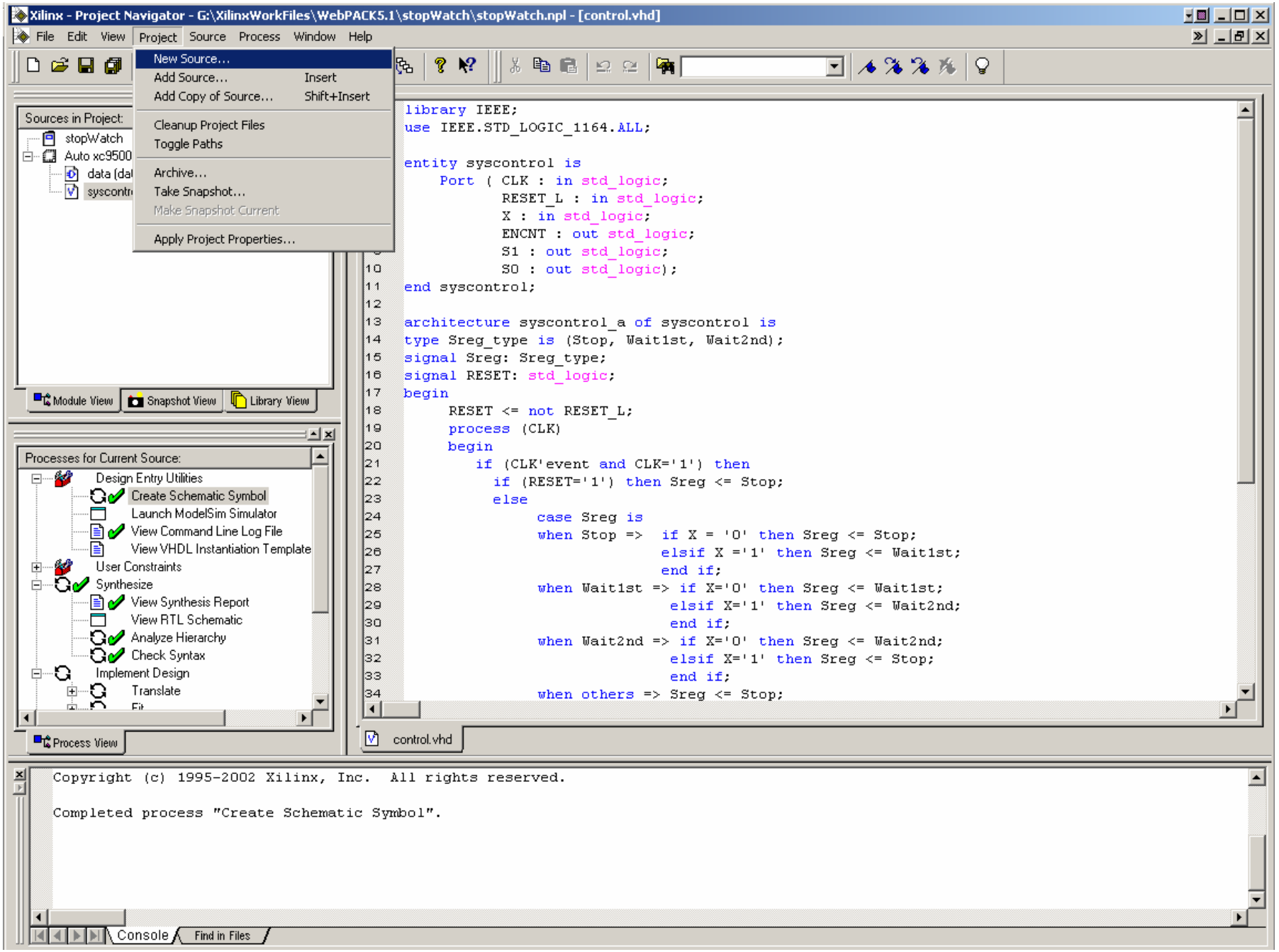
S0=

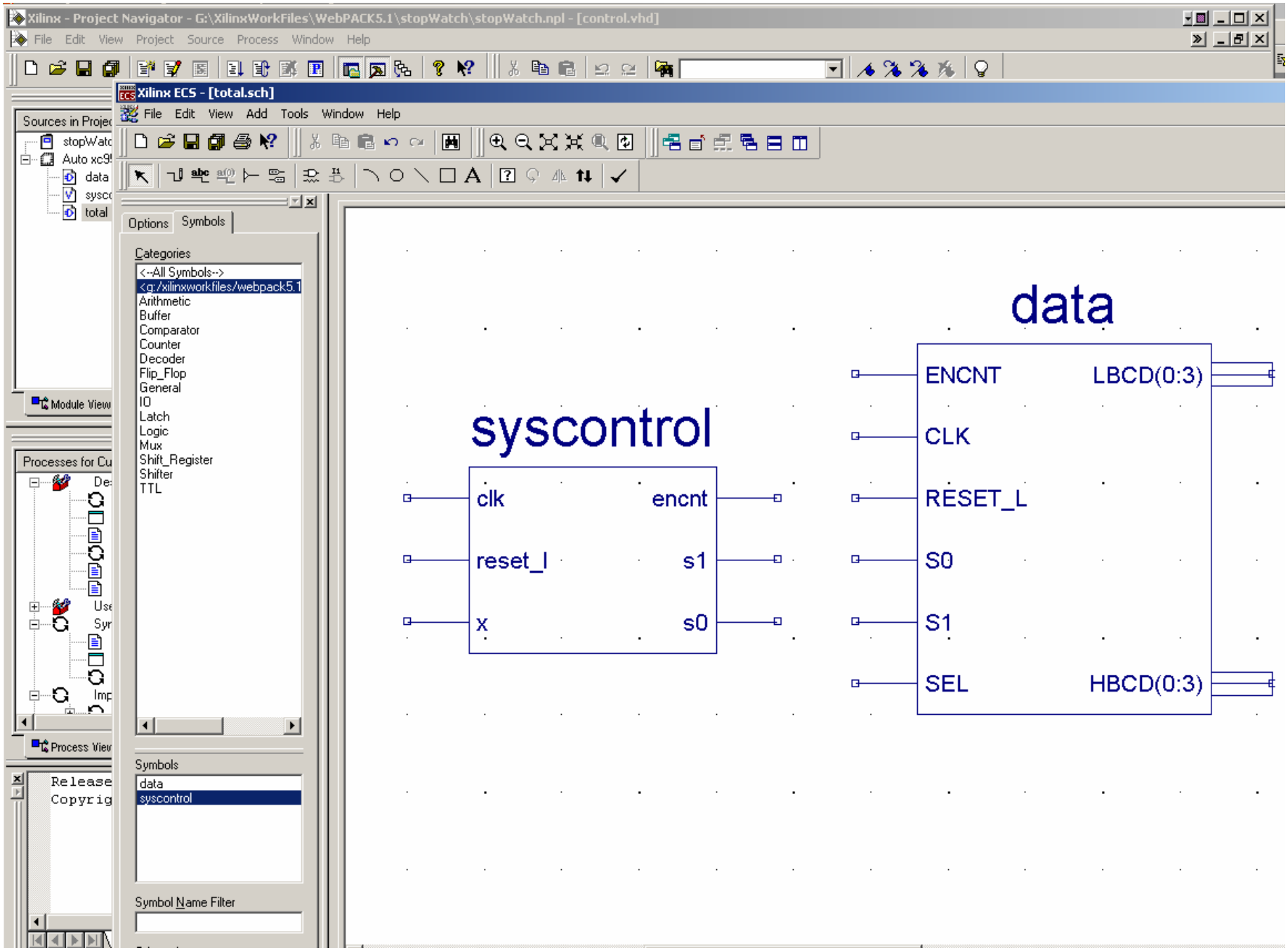


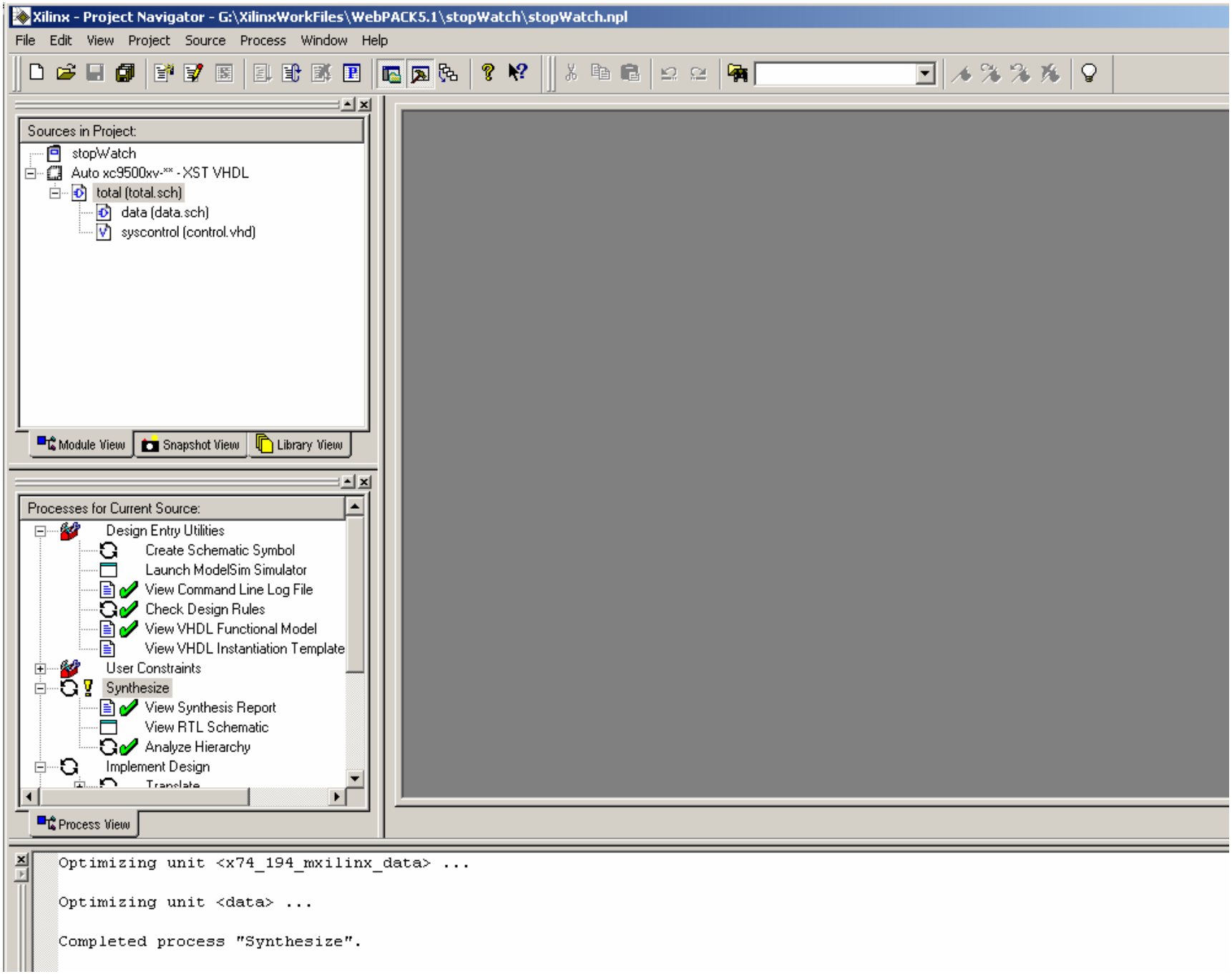
ENCNT=

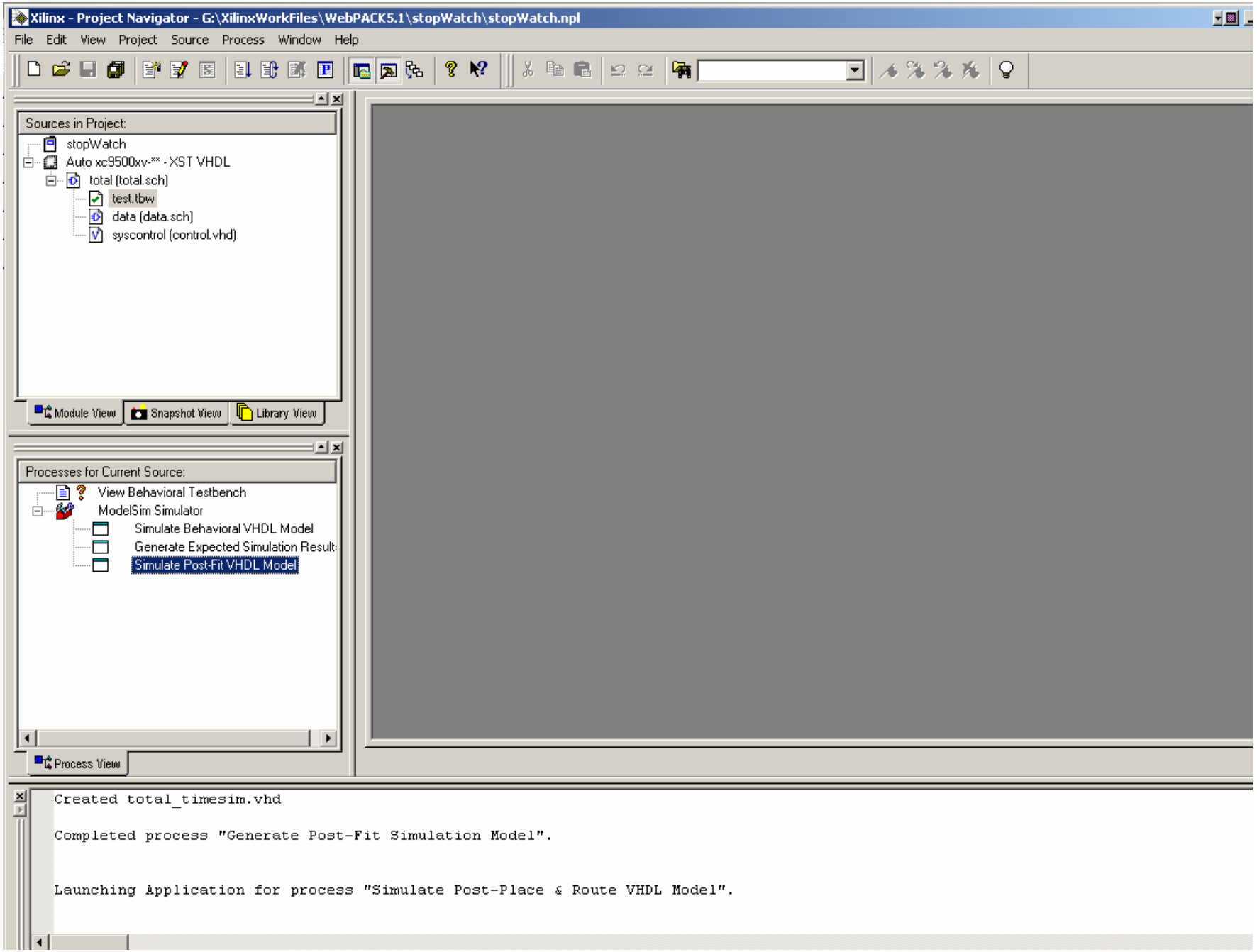
Alternative (VHDL program)


Xilinx - Project Navigator - G:\XilinxWorkFiles\WebPACK5.1\stopWatch\stopWatch.npl - [control.vhd]
File Edit View Project Source Process Window Help
Sources in Project:
stopWatch
Auto xc9500xv-** - XST VHDL
data (data.sch)
syscontrol (control.vhd)
Module View
Snapshot View
Library View
Processes for Current Source:
Design Entry Utilities
Create Schematic Symbol
Launch ModelSim Simulator
View Command Line Log File
Check Design Rules
View VHDL Functional Model
View VHDL Instantiation Template
User Constraints
Synthesize
View Synthesis Report
View RTL Schematic
Analyze Hierarchy
Implement Design
Process View
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity syscontrol is
5 Port (CLK : in std_logic;
6 RESET_L : in std_logic;
7 X : in std_logic;
8 ENCNT : out std_logic;
9 S1 : out std_logic;
10 S0 : out std_logic);
11 end syscontrol;
12
13 architecture syscontrol_a of syscontrol is
14 type Sreg_type is (Stop, Wait1st, Wait2nd);
15 signal Sreg: Sreg_type;
16 signal RESET: std_logic;
17 begin
18 RESET <= not RESET_L;
19 process (CLK)
20 begin
21 if (CLK'event and CLK='1') then
22 if (RESET='1') then Sreg <= Stop;
23 else
24 case Sreg is
25 when Stop => if X = '0' then Sreg <= Stop;
26 elsif X ='1' then Sreg <= Wait1st;
27 end if;
28 when Wait1st => if X='0' then Sreg <= Wait1st;
29 elsif X='1' then Sreg <= Wait2nd;
30 end if;
31 when Wait2nd => if X='0' then Sreg <= Wait2nd;
32 elsif X='1' then Sreg <= Stop;
33 end if;
34 when others => Sreg <= Stop;
control.vhd
Library "C:/Xilinx/xc9500xl/data/lib.xst" Consulted
Library "C:/Xilinx/data/librtl.xst" Consulted
Optimizing unit <syscontrol> ...
Completed process "Synthesize".


Xilinx - Project Navigator - G:\XilinxWorkFiles\WebPACK5.1\stopWatch\stopWatch.npl - [control.vhd]
File Edit View Project Source Process Window Help
New Source...
Add Source... Insert
Add Copy of Source... Shift+Insert
Cleanup Project Files
Toggle Paths
Archive...
Take Snapshot...
Make Snapshot Current
Apply Project Properties...
Sources in Project:
stopWatch
Auto xc9500
Module View
Snapshot View
Library View
Processes for Current Source:
Design Entry Utilities
Create Schematic Symbol
Launch ModelSim Simulator
View Command Line Log File
View VHDL Instantiation Template
User Constraints
Synthesize
View Synthesis Report
View RTL Schematic
Analyze Hierarchy
Check Syntax
Implement Design
Translate
Process View
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity syscontrol is
Port (CLK : in std_logic;
RESET_L : in std_logic;
X : in std_logic;
ENCNT : out std_logic;
S1 : out std_logic;
S0 : out std_logic);
end syscontrol;
architecture syscontrol_a of syscontrol is
type Sreg_type is (Stop, Wait1st, Wait2nd);
signal Sreg: Sreg_type;
signal RESET: std_logic;
begin
RESET <= not RESET_L;
process (CLK)
begin
if (CLK'event and CLK='1') then
if (RESET='1') then Sreg <= Stop;
else
case Sreg is
when Stop => if X = '0' then Sreg <= Stop;
elsif X ='1' then Sreg <= Wait1st;
end if;
when Wait1st => if X='0' then Sreg <= Wait1st;
elsif X='1' then Sreg <= Wait2nd;
end if;
when Wait2nd => if X='0' then Sreg <= Wait2nd;
elsif X='1' then Sreg <= Stop;
end if;
when others => Sreg <= Stop;
control.vhd
Copyright (c) 1995-2002 Xilinx, Inc. All rights reserved.
Completed process "Create Schematic Symbol".
Console
Find in Files


Xilinx - Project Navigator - G:\XilinxWorkFiles\WebPACK5.1\stopWatch\stopWatch.npl - [control.vhd]
File Edit View Project Source Process Window Help
Xilinx ECS - [total.sch]
File Edit View Add Tools Window Help
Sources in Projec
stopWatc
Auto xc9
data
sysc
total
Module View
Processes for Cu
Process View
Release
Copyrig
Options
Symbols
Categories
<--All Symbols-->
<g:/xilinxworkfiles/webpack5.1
Arithmetic
Buffer
Comparator
Counter
Decoder
Flip_Flop
General
IO
Latch
Logic
Mux
Shift_Register
Shifter
TTL
Symbols
data
syscontrol
Symbol Name Filter
syscontrol
clk
reset_l
x
encnt
s1
s0
data
ENCNT
CLK
RESET_L
S0
S1
SEL
LBCD(0:3)
HBCD(0:3)


Xilinx - Project Navigator - G:\XilinxWorkFiles\WebPACK5.1\stopWatch\stopWatch.npl
File Edit View Project Source Process Window Help
Sources in Project:
stopWatch
Auto xc9500xv-** - XST VHDL
total (total.sch)
data (data.sch)
syscontrol (control.vhd)
Module View
Snapshot View
Library View
Processes for Current Source:
Design Entry Utilities
Create Schematic Symbol
Launch ModelSim Simulator
View Command Line Log File
Check Design Rules
View VHDL Functional Model
View VHDL Instantiation Template
User Constraints
Synthesize
View Synthesis Report
View RTL Schematic
Analyze Hierarchy
Implement Design
Process View
Optimizing unit <x74_194_mxilinx_data> ...
Optimizing unit <data> ...
Completed process "Synthesize".

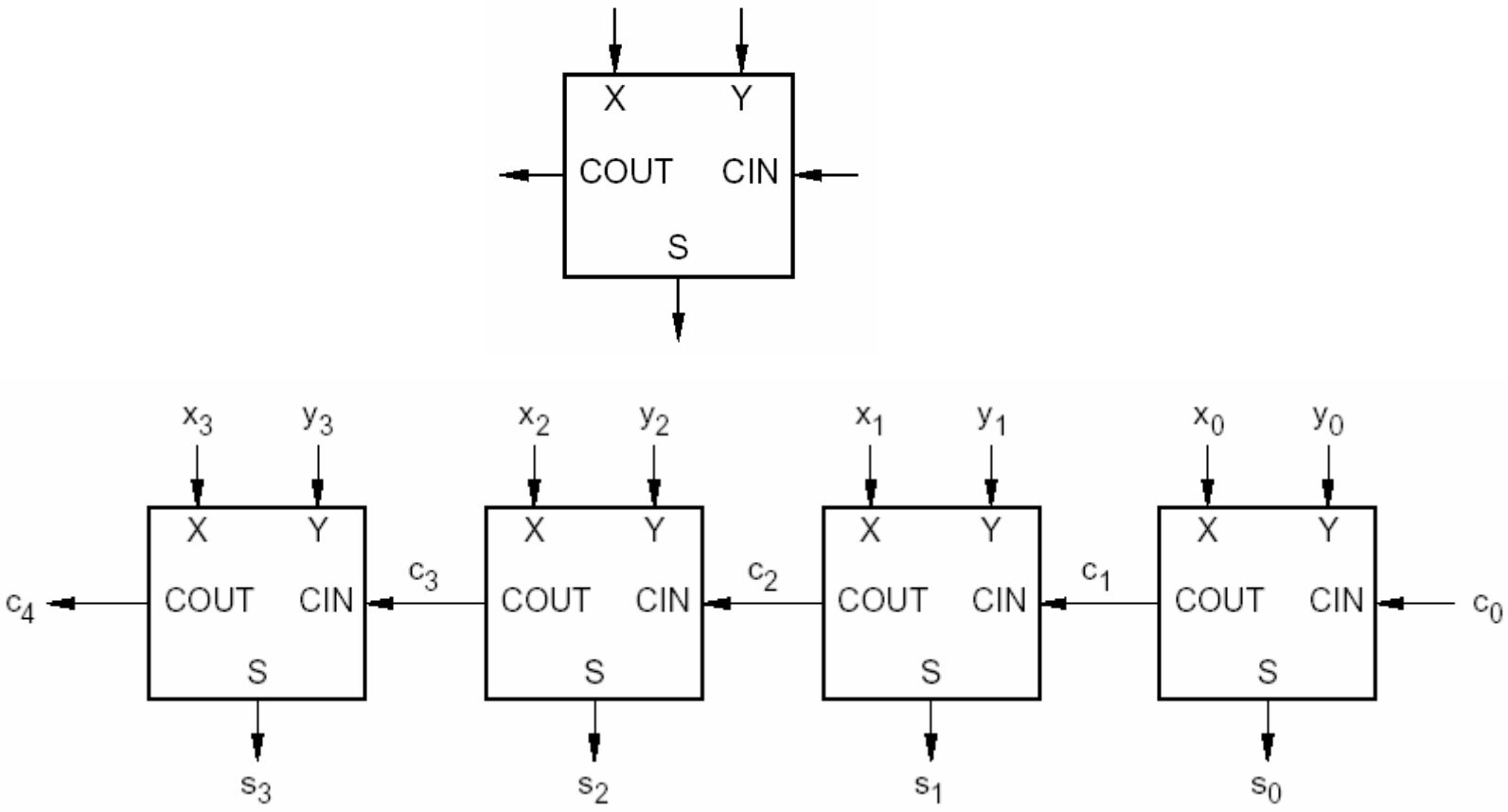

Xilinx - Project Navigator - G:\XilinxWorkFiles\WebPACK5.1\stopWatch\stopWatch.npl
File Edit View Project Source Process Window Help
Sources in Project:
stopWatch
Auto xc9500xv-** - XST VHDL
total (total.sch)
test.tbw
data (data.sch)
syscontrol (control.vhd)
Module View
Snapshot View
Library View
Processes for Current Source:
View Behavioral Testbench
ModelSim Simulator
Simulate Behavioral VHDL Model
Generate Expected Simulation Result
Simulate Post-Fit VHDL Model
Process View
Created total_timesim.vhd
Completed process "Generate Post-Fit Simulation Model".
Launching Application for process "Simulate Post-Place & Route VHDL Model".

4-bit Serial Adder

(Decomposing state machines, synchronous design methodology, system controller design)

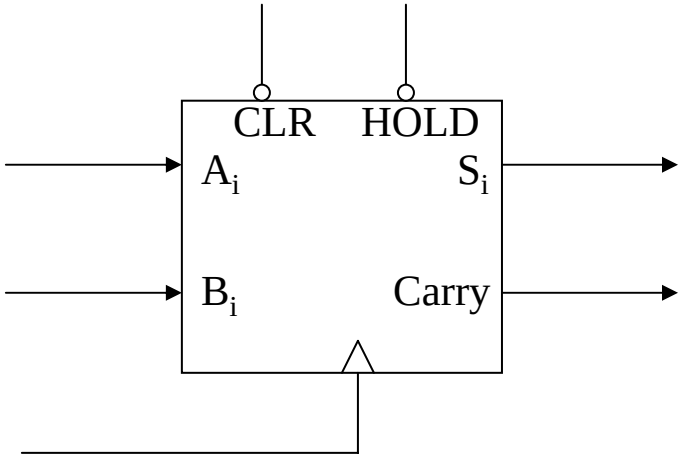
Review of combinational adders

1-bit full adder and 4-bit ripple adder



Binary serial adder

- Serial input feed (two data streams A and B)
- Generate serial output

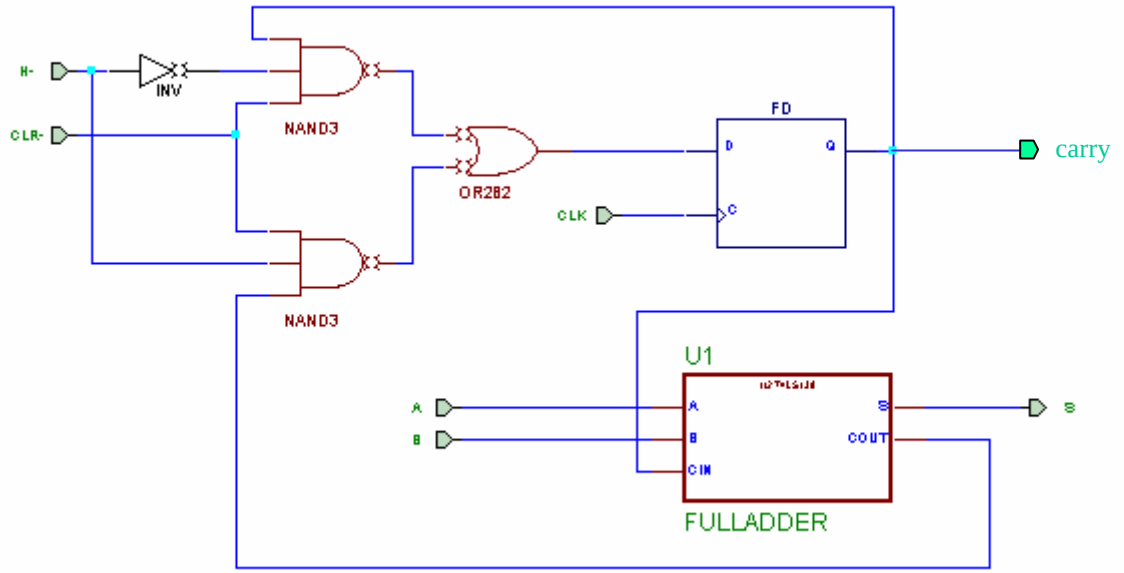


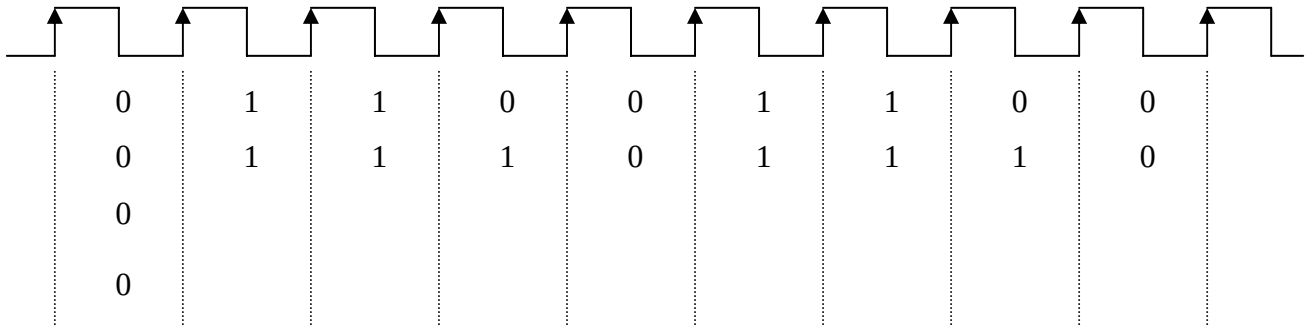
CLR – clear the previous carry synchronously

HOLD – hold the previous value of the carry

Binary serial adder implementation

- LS183: dual 1 bit full adders

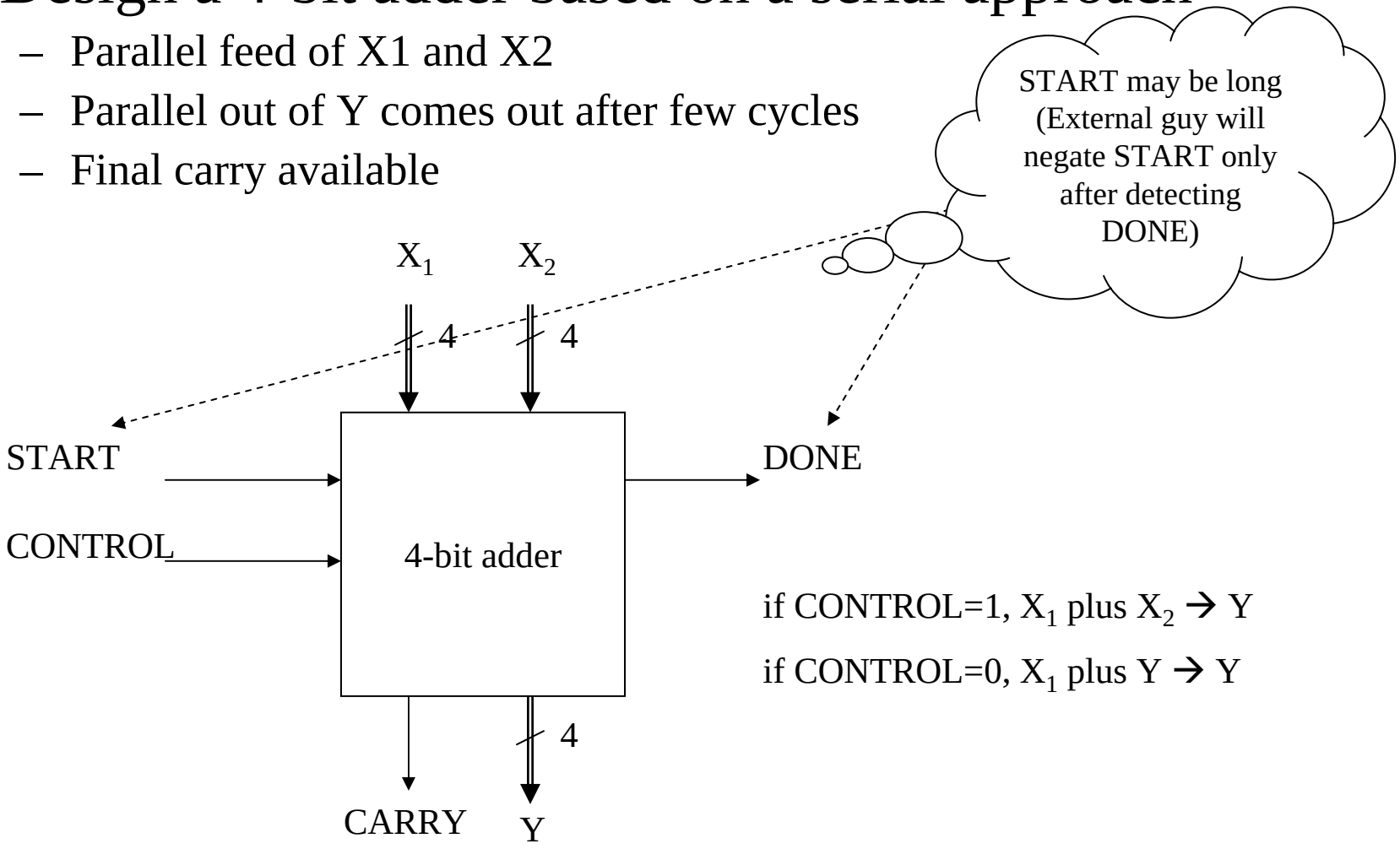




A	0	1	1	0	0	1	1	0	0
B	0	1	1	1	0	1	1	1	0
Q(Carry)	0								
S	0								

Problem Statement

- Design a 4-bit adder based on a serial approach
  - Parallel feed of X1 and X2
  - Parallel out of Y comes out after few cycles
  - Final carry available

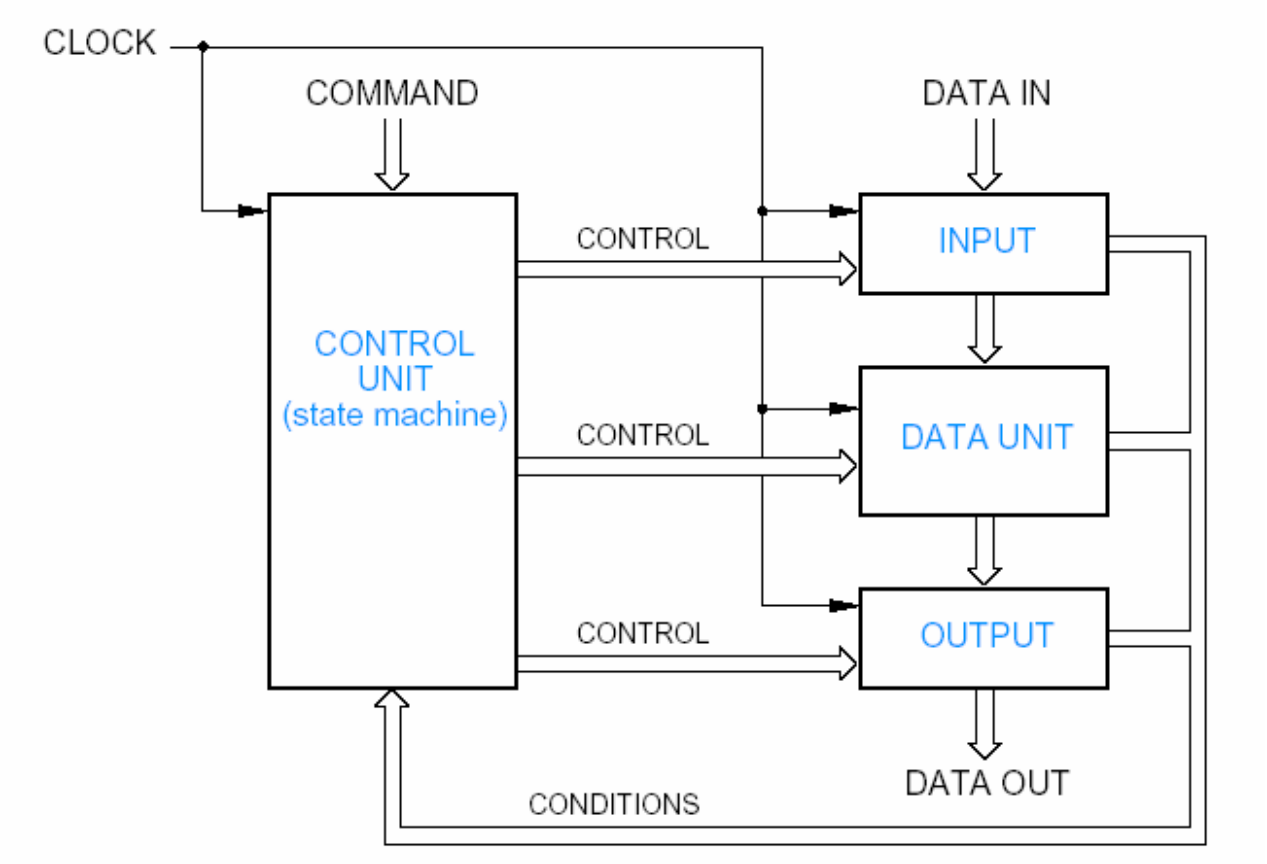


Start from LBB

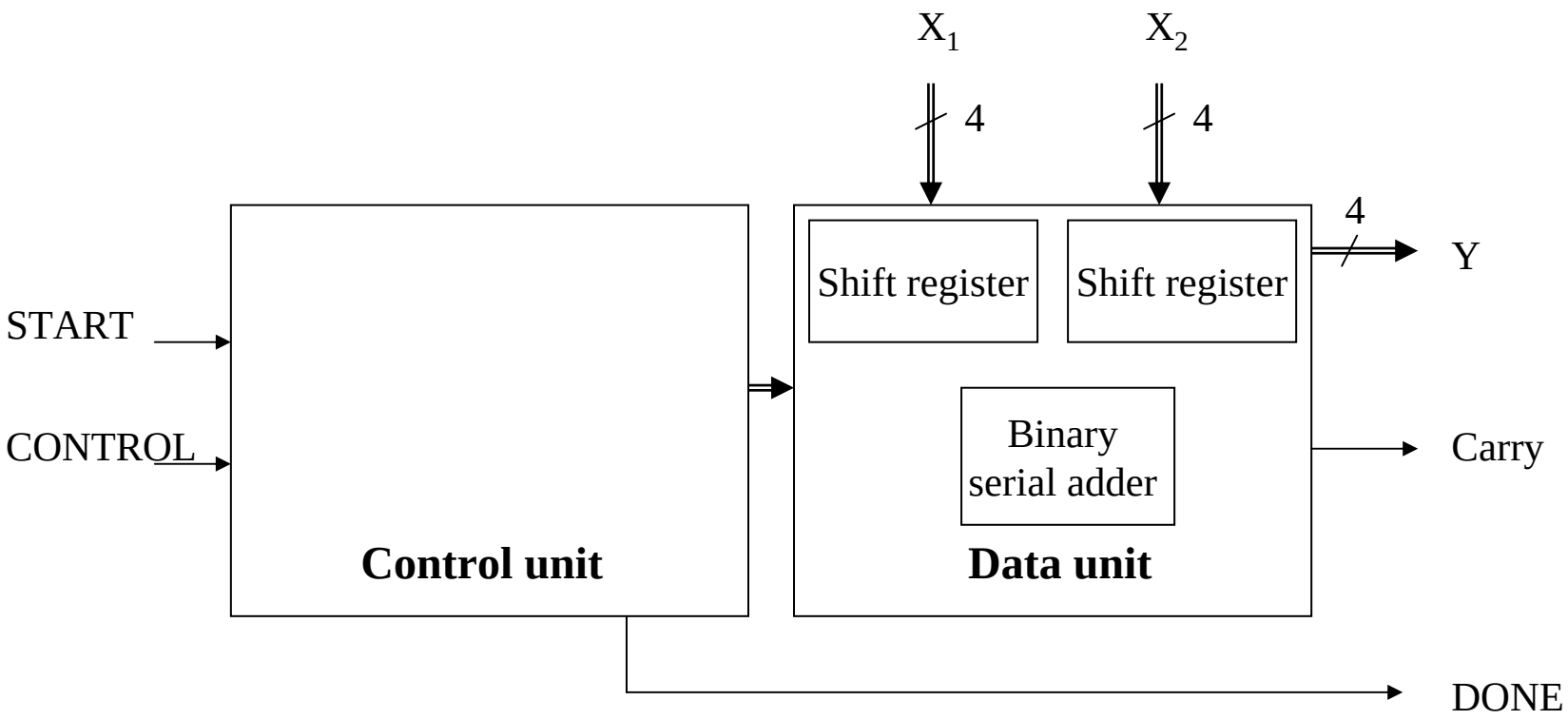
- Here, we have some basic building blocks in mind
  - 4-bit shift registers, binary serial adder, etc
- Tie these elements together and make them controllable from the outside world
- We want to take maximum advantage of common building blocks (MSI chips that are available)

Design System Architecture

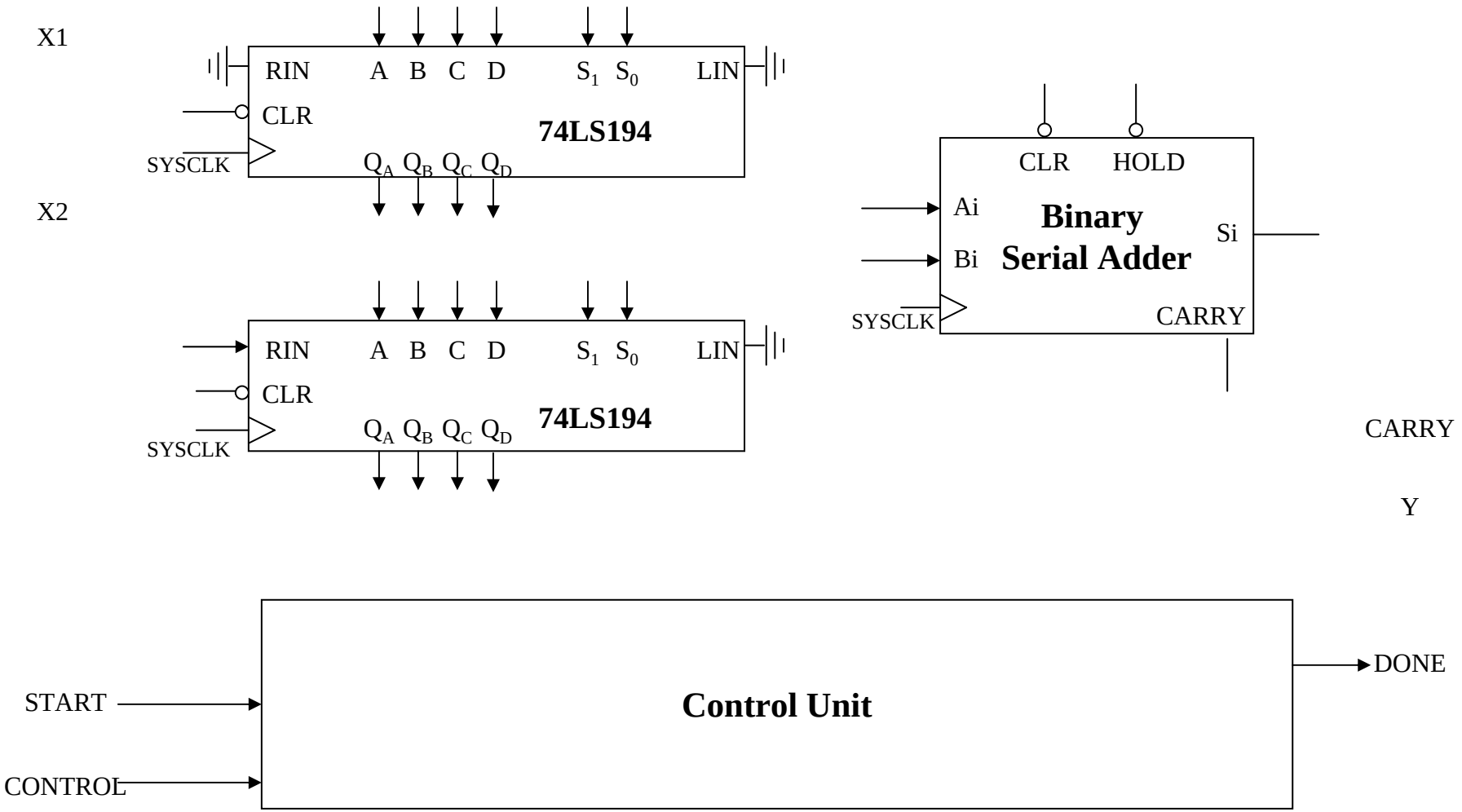
- First step: divide the system into a control unit and data unit
  - Data unit – stores, routes, combines, and generally process data
  - Control unit – starting & stopping the process, testing conditions, and deciding what to do next



Data unit & Control unit



Architecture

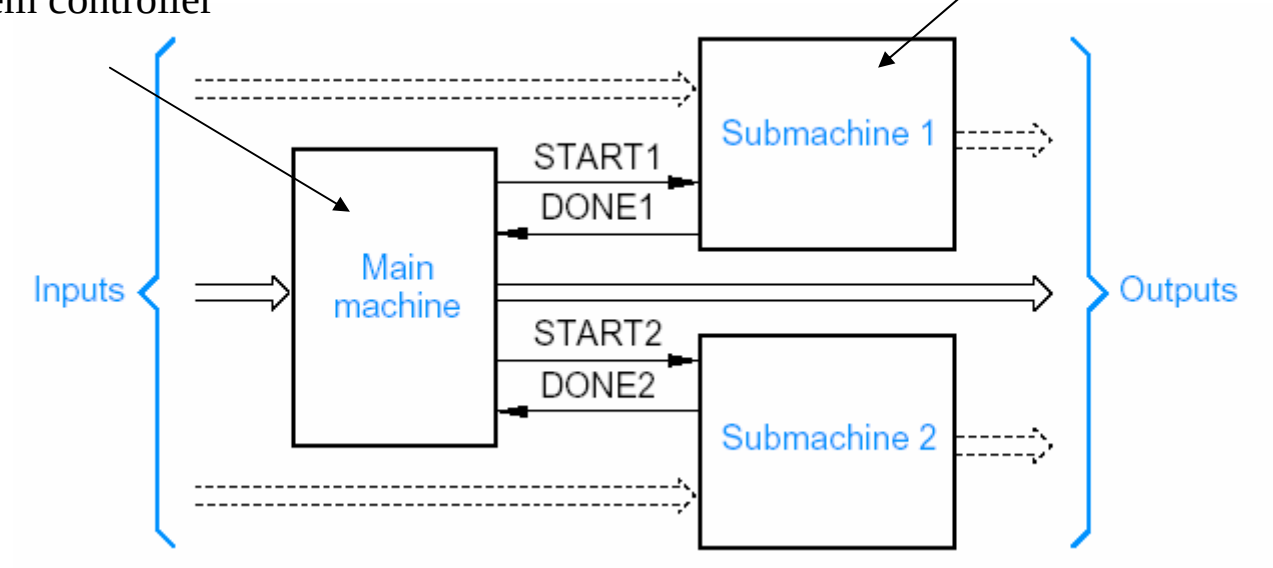


Decomposing state machine

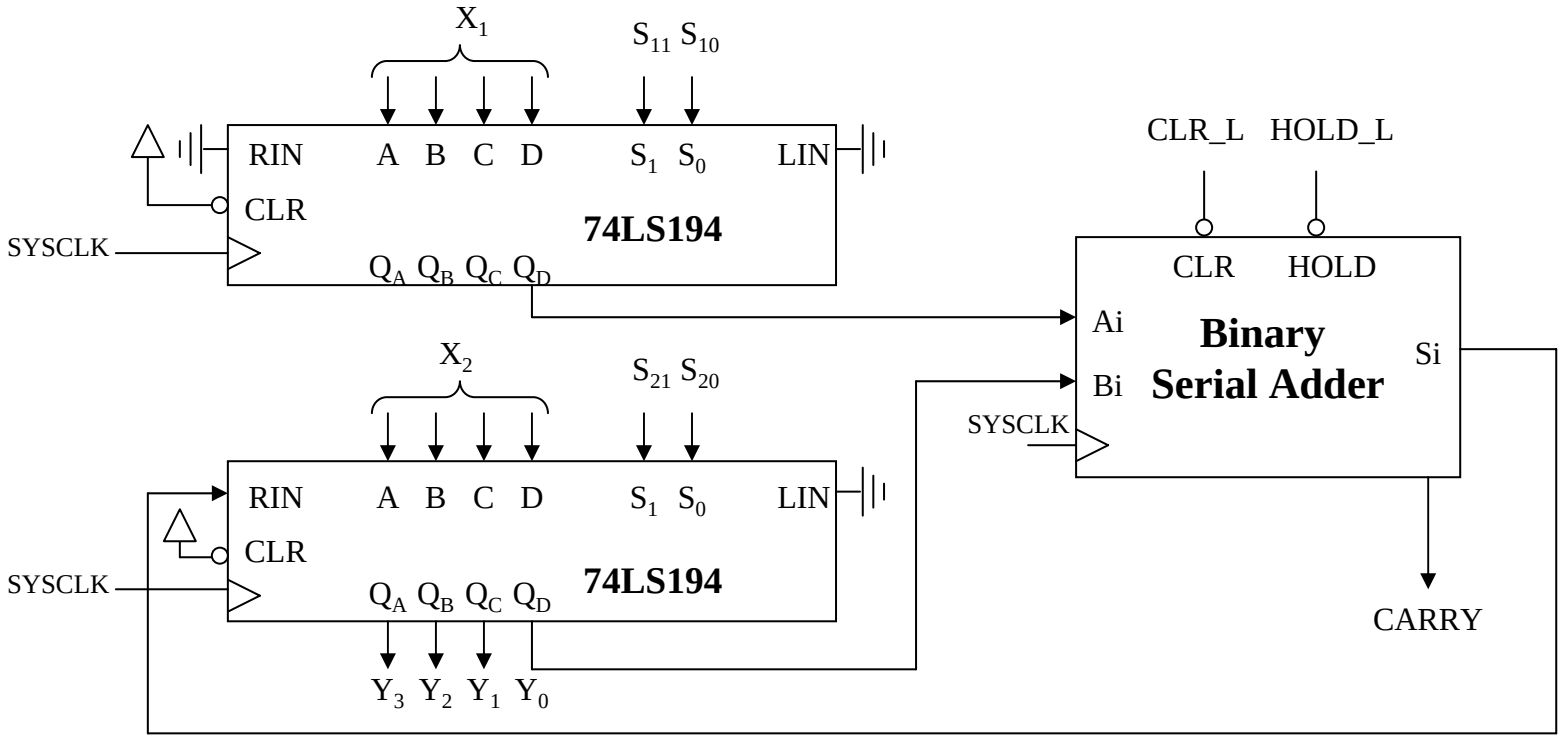
- Second step: the state machine part (control unit) can be decomposed into several parts
  - Main machine (system controller) – provides the primary inputs and outputs and does top level control
  - Submachines – perform lower-level steps under control of main machine
    - Typical submachine – counter
      - Saves 2^n states in main machine
      - Easier to follow the control

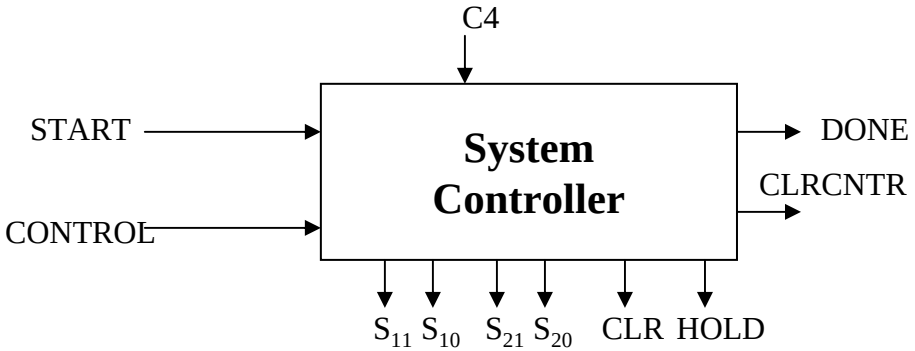
2 bit counter to count # of bits added

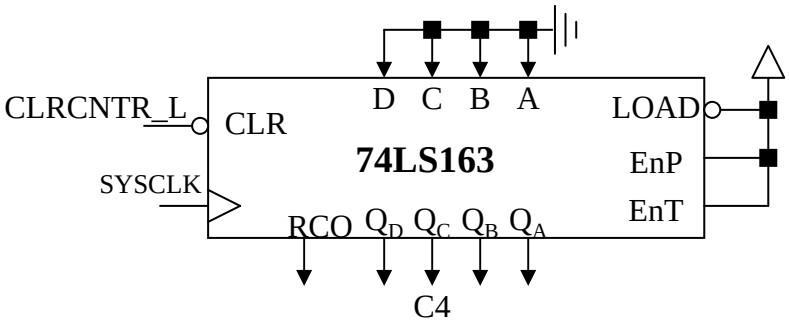
Our system controller



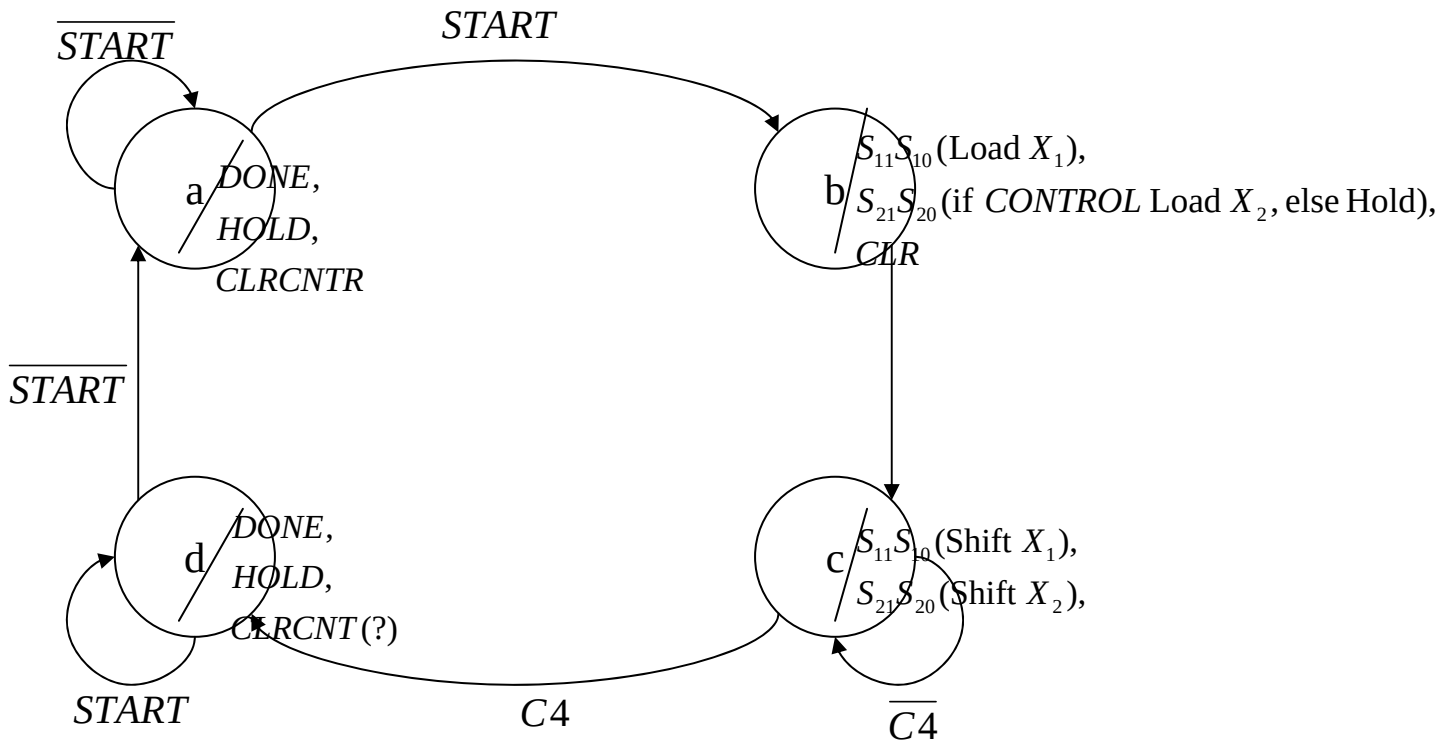
Architecture







System controller design



Example results

0101 (5)

+ 0001 (1)

0110 (6)

Sate	Carry	Reg 1	Reg 2	Counter
a	-	-	-	0
b	-	-	-	0
c	0	0101	0001	1
c	1	0010	0000	2
c	0	0001	1000	3
c	0	0000	1100	4
d	0	0000	0110	5

State assignment and Transition Table

	Q1Q0	Q1Q0 (D1D0)	DONE	CLRCNTR	CLR	HOLD	S11	S10	S21	S20
a	0 0	0 ST	1	1	0	1	x	x	0	0
b	0 1	1 1	0	0	1	x	1	1	CONT'	CONT'
c	1 1	1 $\overline{C4}$	0	0	0	0	0	1	0	1
d	1 0	ST 0	1	x	0	1	x	x	0	0

Excitation and Output Eqs.

Q_1 \ Q_0	0	1
0	0	1
1	ST	1

$$D_1 = Q_0 + ST \cdot Q_1$$

	0	1
0	ST	1
1	0	$\overline{C4}$

$$D_0 = \overline{Q_1} Q_0 + \overline{Q_1} \cdot ST + Q_0 \cdot \overline{C4}$$

Q_1 \ Q_0	0	1
0	1	0
1	1	0

$$DONE = \overline{Q_0}$$

Q_1 \ Q_0	0	1
0	1	0
1	x	0

$$CLRCNTR = \overline{Q_0}$$

Q_1 \ Q_0	0	1
0	0	1
1	0	0

$$CLR = \overline{Q_1} Q_0$$

Q_1 \ Q_0	0	1
0	1	x
1	1	0

$$HOLD = \overline{Q_0}$$

Q_1 \ Q_0	0	1
0	x	1
1	x	0

$$S_{11} = \overline{Q_1}$$

Q_1 \ Q_0	0	1
0	x	1
1	x	1

$$S_{10} = 1$$

Q_1 \ Q_0	0	1
0	0	c
1	0	0

$$S_{21} = \overline{Q_1} Q_0 \cdot CONTROL$$

Q_1 \ Q_0	0	1
0	0	c
1	0	1

$$S_{20} = Q_0 \cdot CONTROL + Q_1 Q_0$$

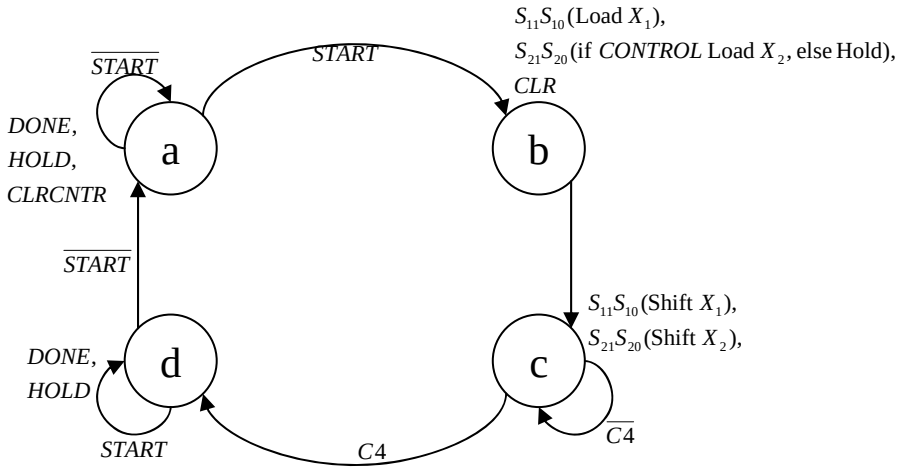
Logic Diagram

Asynchronous Inputs and Output Glitches

- Things to watch out for in synchronous design
  - Clock skew
  - Gating the clock
  - Asynchronous inputs
  - Output glitches

Example

- Binary counting order for our previous state assignment



	Q1Q0	Q1Q0 (D1D0)	DONE	CLRCNTR	CLR	HOLD	S11	S10	S21	S20
a	0 0	0 ST	1	1	0	1	x	x	0	0
b	0 1	1 0	0	0	1	x	1	1	CONT'	CONT'
c	1 0	1 C4	0	0	0	0	0	1	0	1
d	1 1	ST ST	1	x	0	1	x	x	0	0

Focus on part of solution

Q_1 \ Q_0	0	1
0	0	1
1	1	ST

$D_1 = Q_1\overline{Q_0} + \overline{Q_1}Q_0 + ST \cdot Q_1$

	0	1
0	ST	0
1	C4	ST

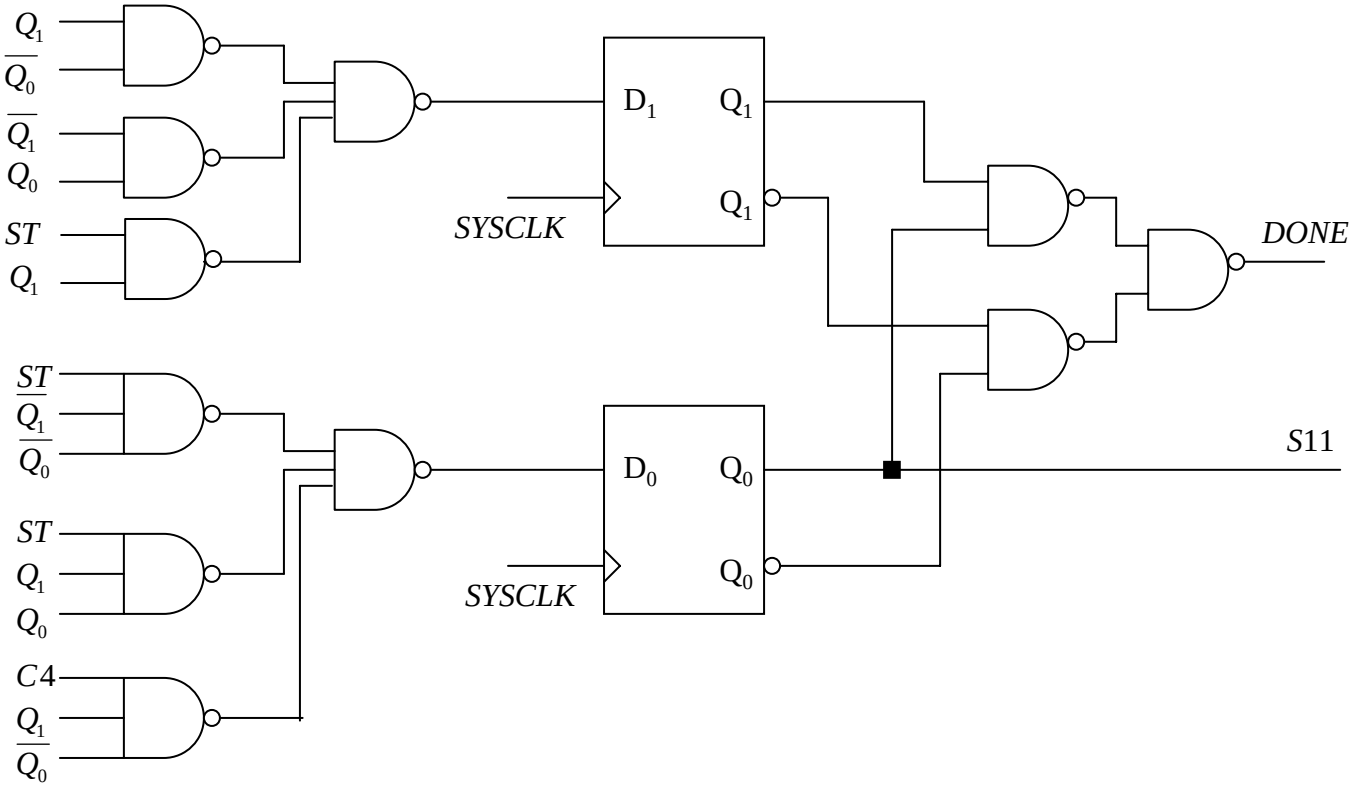
$D_0 = ST \cdot \overline{Q_1}\,\overline{Q_0} + ST \cdot Q_1Q_0 + C4 \cdot Q_1\overline{Q_0}$

Q_1 \ Q_0	0	1
0	1	0
1	0	1

$DONE = \overline{Q_1}\,\overline{Q_0} + Q_1Q_0$

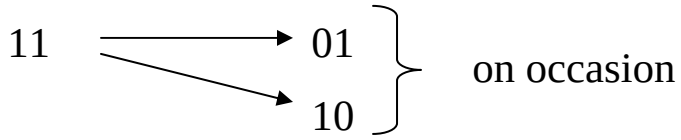
Q_1 \ Q_0	0	1
0	-	1
1	0	-

$S11 = Q_0$

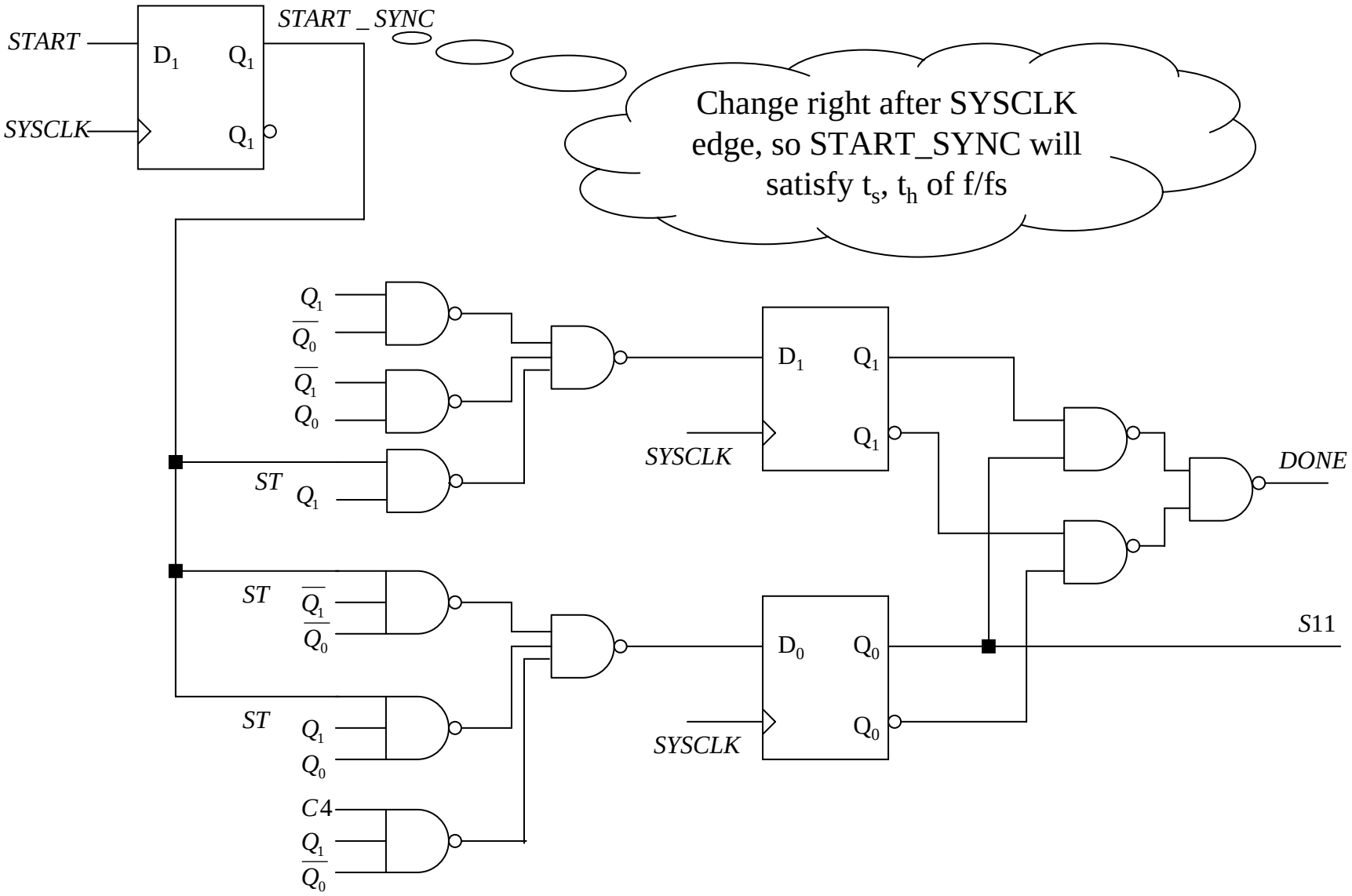


Asynchronous START

- Is START synchronous or asynchronous?
  - Could be either
  - Assume asynchronous (comes from another system not using same SYSCLK)
- Look at transition from “11” to “00” by negating START
- What happens if t_s, t_h of D f/fs are not satisfied due to asynchronous START?
  - Usually stay at 1 or go to 0
  - Problem? – early change to START=0 => “00”
    late change to START=0 => “11”
- Problem: unexpected results can happen
  - Delays not equal, f/fs different
  - Generally – could do either if t_s, t_h not satisfied



Synchronize START (Synchronizer)

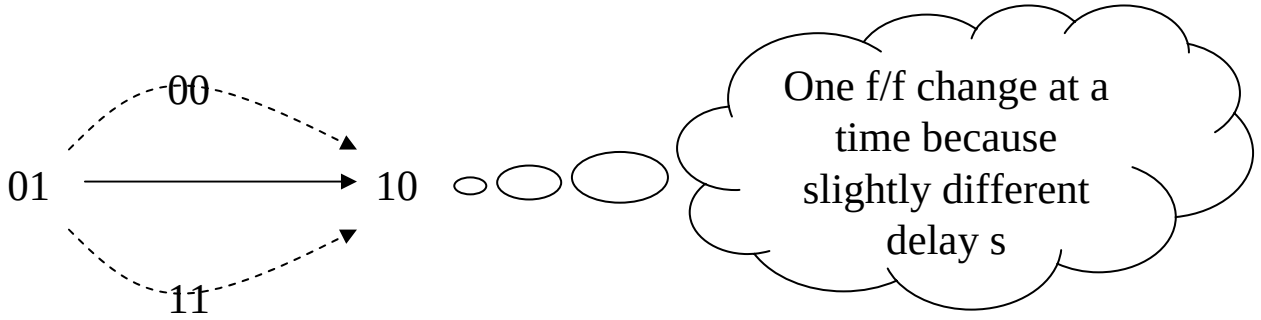


Additional Problem?

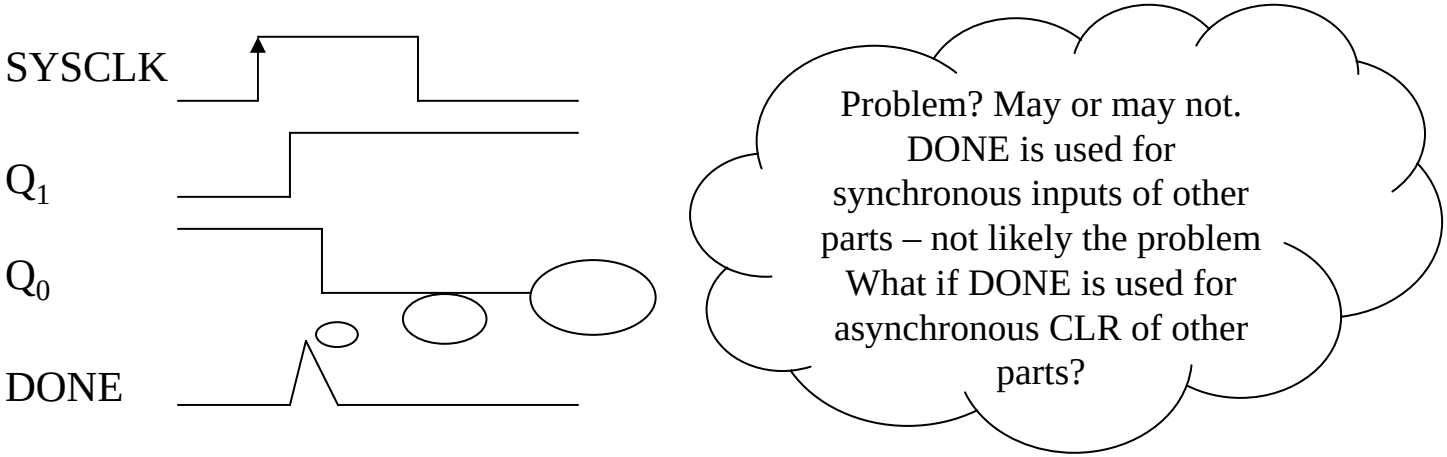
- What else besides START_SYNC=1 or 0?
- Metastable – stuck in middle for a while
  - What happen if START does not satisfy t_s, t_h of "Synchronizer" D f/f
  - START_SYNC not 1 or not 0 for a while
- Metastability – real problem (early versions of several microprocessor chips had this problem!)
- Synchronizer Failure and Metastability
  - Solutions?

Output Glitch on DONE

- Look at transition between b="01" and c="10"

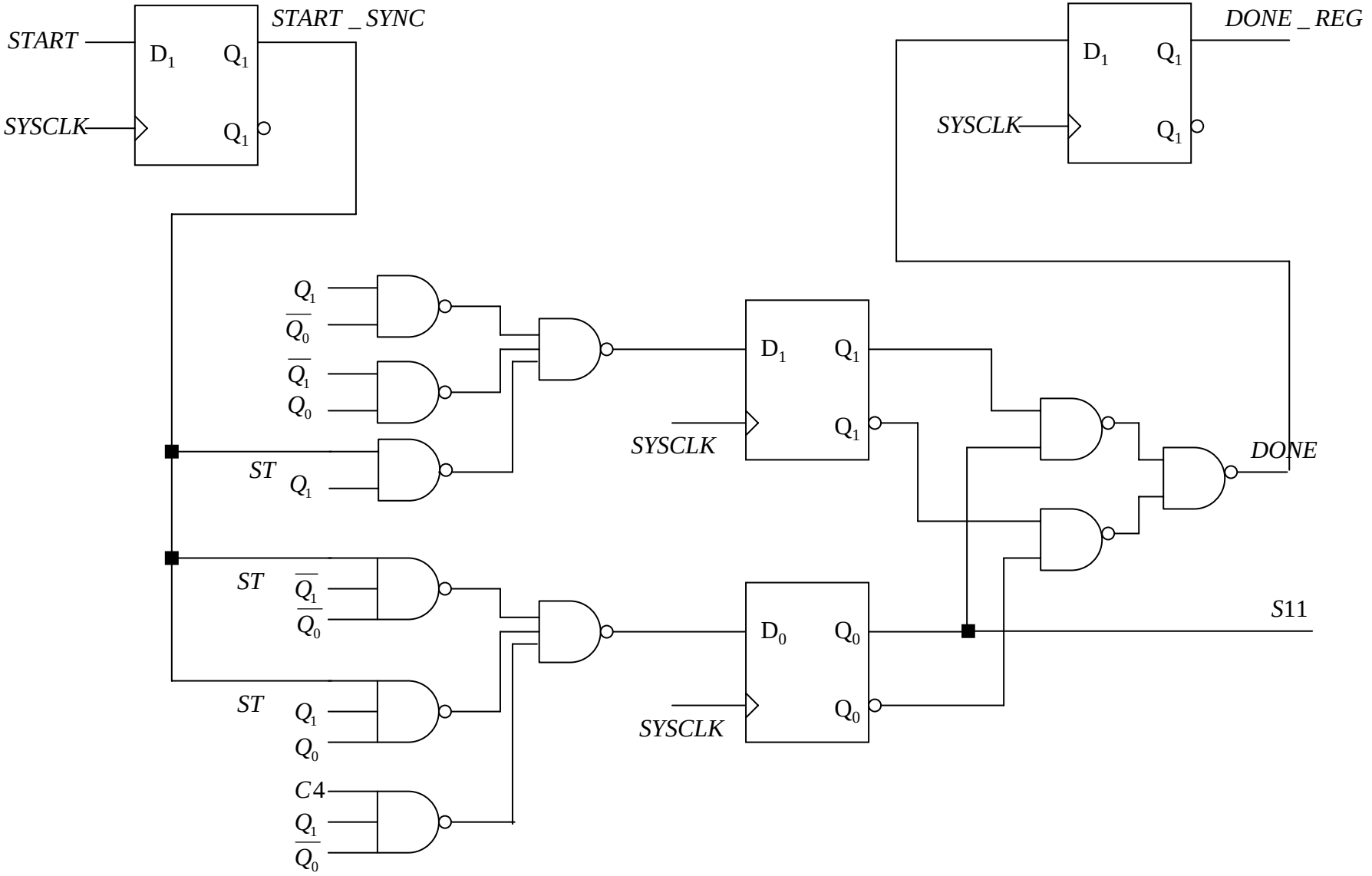


- For a moment in the transition from "01" to "10"
  - Q_1Q_0 = "00" or "11"
  - DONE=1 between states b ("01") and c ("10")



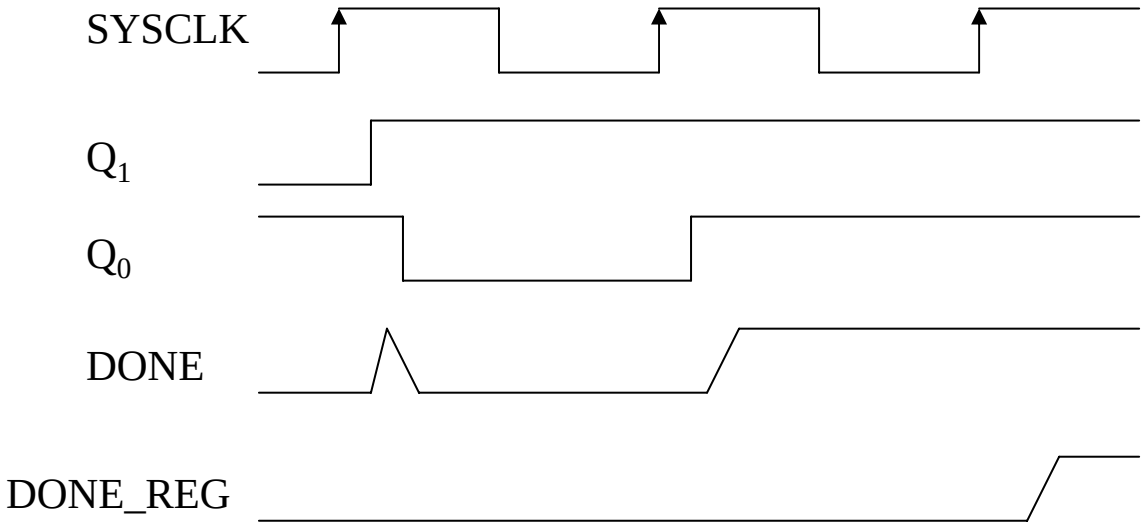
Put a register (Stabilizer) on output

- One D f/f on DONE



Work?

- DONE_REG delayed – usually no problem



- Register output not always needed
  - Good state assignment (compare this with our first state assignment)
  - Some good output logics (e.g., S11)

Synchronous Design Methodology (Summary)

- All LBBs and f/fs are clocked by the same common clock signal
  - We use guaranteed LBBs and f/fs by the manufacturer (critical race free!!)
  - Glitches on combinational circuits connecting LBBs and f/fs have no effect, since the control inputs are sampled only after the glitches have had a chance to settle out
- Three tasks to ensure reliable system operation
  - Minimize and determine the amount of clock skew
  - Ensure that f/fs have positive setup- and hold-time margins
  - Identifying asynchronous inputs, synchronize them with the clock
  - Filter any problematic output glitches with output stabilizers