King Fahd University of Petroleum and Minerals
College of Computer Sciences & Engineering
Computer Engineering Department

Principles of VLSI – COE 360

Serial to Parallel Data Converter

Phase I

Prepared for:
Dr. Muhammad E. S. Elrabaa

By:

Ahmad Zeyad M. Al-Masri	215769
Mohammed Khaled Hadhrawi	212313

25 March 2006

TABLE OF CONTENTS

ILLUSTRATIONS

FIGURES

Problem Statement and Requirements

Implement one of the following project using AMI's 0.5 µm technology (5V supply) and assume 2 pF load capacitance at all your circuit's outputs. Also, all gates should have symmetrical noise margins.

Design a simple serial-to-parallel data converter. The circuit reads data serially, adds a parity bit to it and then output it in parallel (8-bits) along with a strobe signal. The data format is as follows; 2 start bits (two 1s), a single stop bit (0) and the data packets (each is 7-bits long) are separated by a single 1 (continuation bit). When there is no data on the input it is kept low (i.e. 0). The circuit should be fully synchronous with an external clock and have a master asynchronous reset input. The diagram below shows the block diagram of the circuit and the data format.

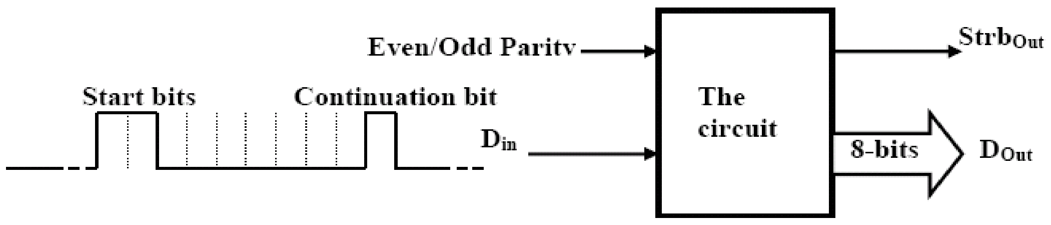


Figure 1: Block diagram of the circuit and data format

Implementation Phases

The deliverables for this project are as follows:

1. ***Phase I: Logic Design, Due Saturday 25/3/2006***

    This is the gate level implementation of the converter. This part should include logic verification (e.g. using Logic Works or HDL).

2. ***Phase II: Circuit Design, Due Saturday 29/4/2006***

    This is the transistor level implementation of the project. This part includes all the SPICE files simulation results.

3. ***Phase III: Mask Design (layout), Due Sunday 28/5/2006***

    This is the physical mask level (i.e. layout) implementation of the project. It includes the post-layout verification using IRsim and SPICE simulations. All layouts should be DRC clean and clearly labeled. A short final report documenting the whole project should be submitted and an exit interview shall be conducted.

Introduction to Phase (1)

In phase, and as stated in the problem statement, it is required to design a serial to parallel converter and provide the logic design of this converter in the gate level. In this report, the logic design and the verification using simulation are implemented using Xilinx Foundation Software and all schematics and simulation graphs are provided.

Data Path (DP) and Control Unit (CU)

The serial to parallel converter design is divided into two major blocks; the first is the data path witch deals with data and manipulate or process that data. The other major block is the control unit witch generate controlling signal to manage the functionality and the behavior of the data path

The inputs for the converter:

Master clock
Master asynchronous reset
Serial data (sequence of bits)
Odd/Even parity bit

The outputs for the converter:

Parallel 8 bits
Strobe signal bit

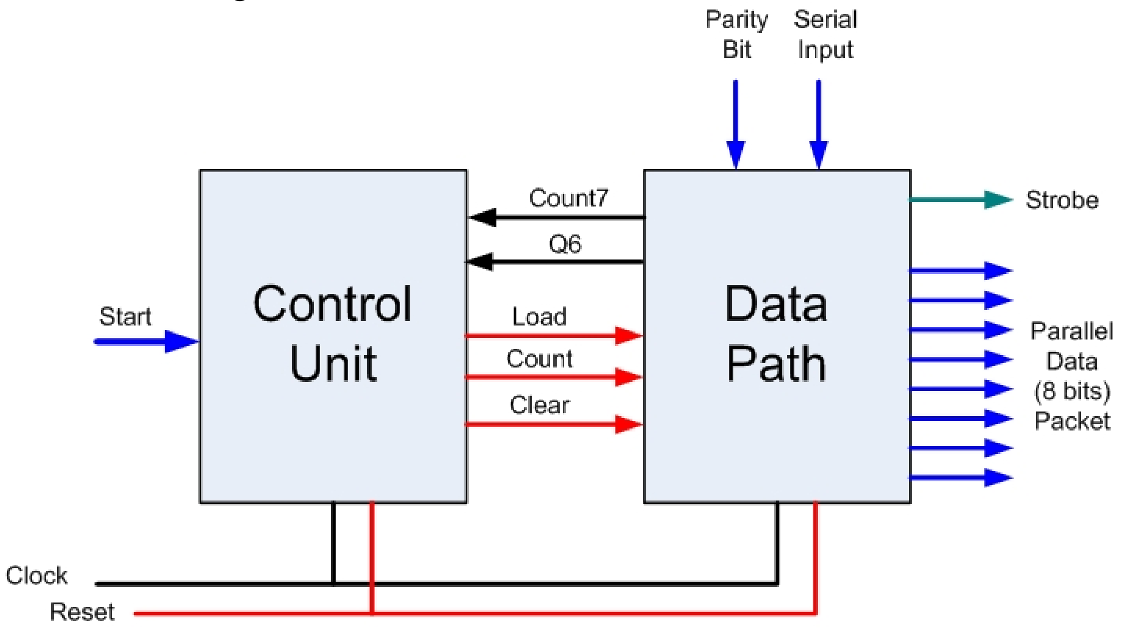


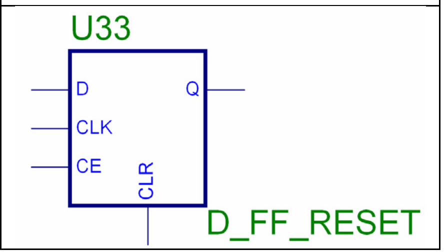
Figure 2: Data Path and Control Unit, inputs and outputs

General Logic:

Before we move on to the data path and the control unit, let us first define some basic logics.

D Flip Flop w/ reset & CE:

The D Flip Flop with reset and chip enable used in this project is consisting of two main parts: The simple D Flip Flop and a multiplexer.

CLR	CE	Output (Q)	Logic
0	0	Previous State	
0	1	New State	
1	X	CLear0	

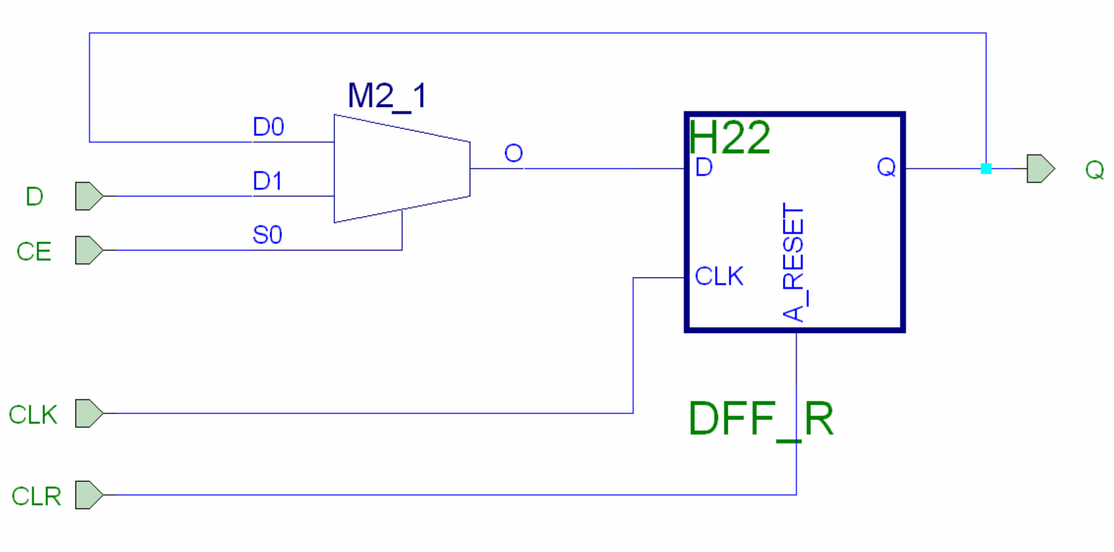


Figure 3: D Flip Flop w/CLR & CE Logic Diagram

Simple D Flip Flop:

In reality it is just two D Latches with master slave design.

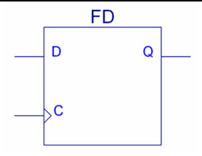
Q	D	Q(t+1)	Logic
0	0	0	
0	1	1	
1	0	0	
1	1	1	

Table 1

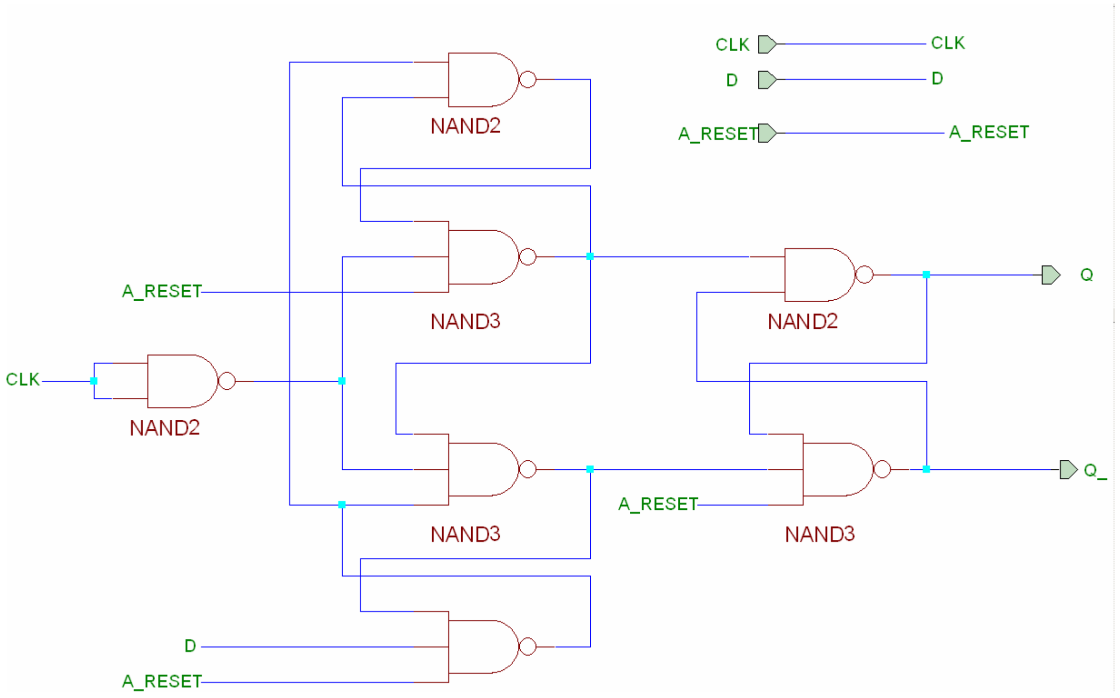


Figure 4: Simple D Flip Flop Logic Diagram

2X1 Multiplexer:

It is only a simple multiplexer.

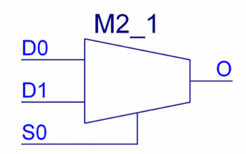
S0	D0	D1	Output	Logic
0	0	X	0	
0	1	X	1	
1	X	0	0	
1	X	1	1	

Table 2

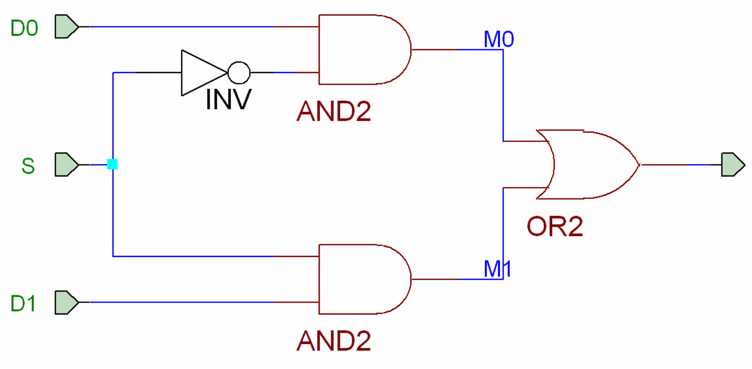


Figure 5: 2X1 Multiplexer Logic Diagram

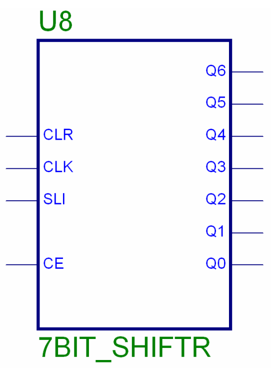
Data Path, Required Logic

To implement the aimed functionality of the data path we need the following components:

1- 7 bit serial shift register.
2- 3 bit counter (since we need to count 7 sequenced bits)
3- 8 bit parallel load register.
4- Parity logic.

Here, we will discuss each of these components providing their logic and the circuit implementation.

7 Bit Shift Register



The data are injected to the serial to parallel converter serially and then outputted in parallel fashion, so, the received data should be saved in a register; and since the packet structure are of (7 bits + parity bit), we need to save only 7 bit in the serial shift register.

The data is propagated through the D-flip flop that is used as a memory element. The Serial input at the first clock cycle is propagated and being stored in the first D- flip flop, in the next clock cycle, that bit would be stored in the next connected D- flip flop. The D- flip flop that is used in this design has an asynchronous reset.

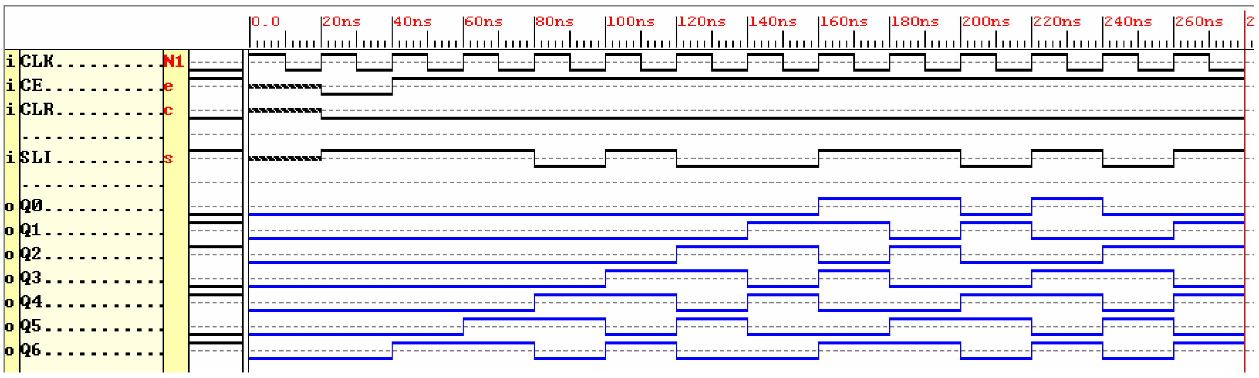


Figure 6: Timing diagram for the operation of the Serial Shift Register

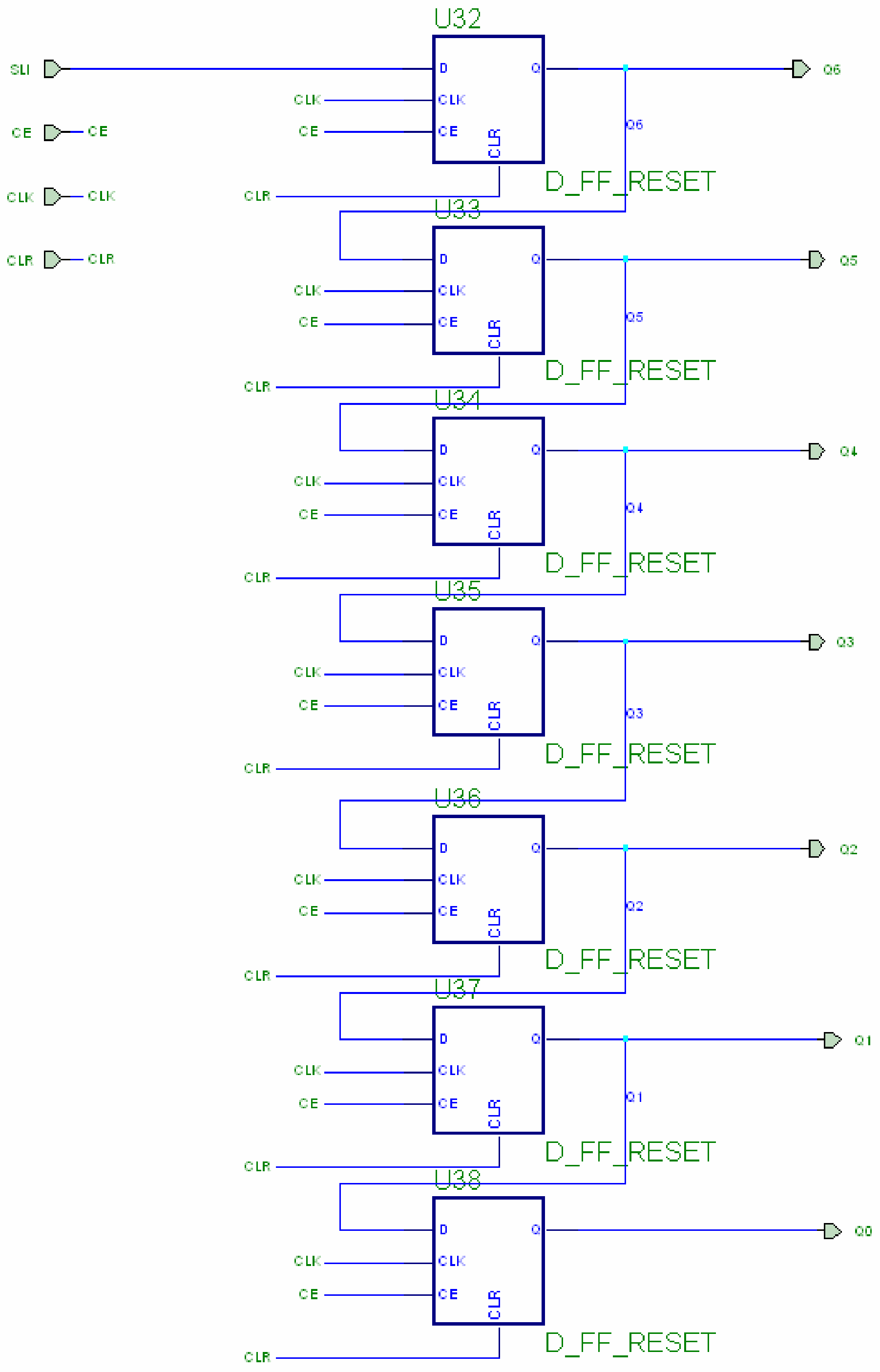
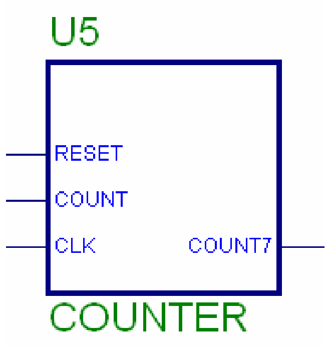


Figure 7: Serial Shift Register Logic Diagram

3 Bit Counter

The counter is used to know exactly when the serial bit inputted are 7 bits, so, we need to count 7 clock cycles after the initiation of the starting bits or the continuation bits. To implement this, three D-flip lops are used.



Previous State A B C	Next State $A^+ B^+ C^+$	Q2.Q1.Q0
0 0 0	0 0 1	0 0 0
0 0 1	0 1 0	0 0 1
0 1 0	0 1 1	0 1 0
0 1 1	1 0 0	0 1 1
1 0 0	1 0 1	1 0 0
1 0 1	1 1 0	1 0 1
1 1 0	1 1 1	1 1 0
1 1 1	0 0 0	1 1 1

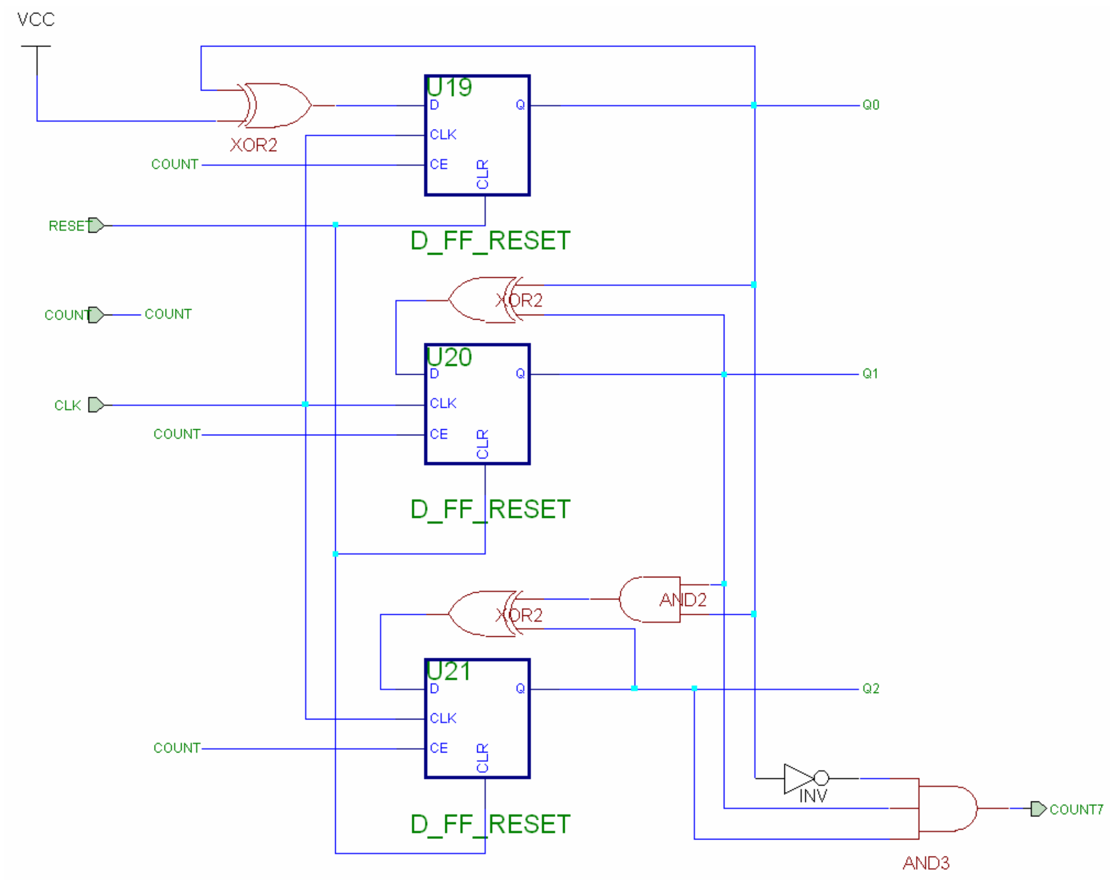


Figure 8: 3-Bit counter

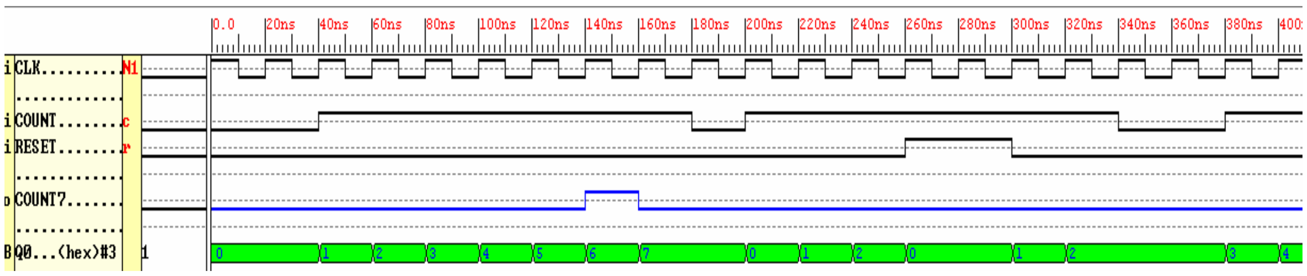
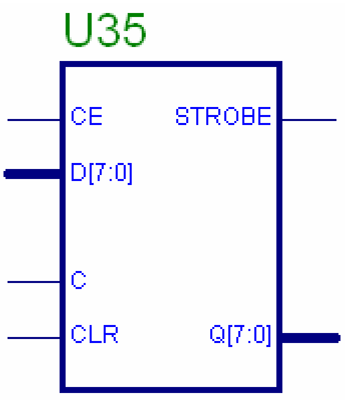


Figure 9: Timing diagram for the 3-bit counter

7 Parallel Load Register

This register will load the saved 7 bits that were inputted serially and the parity computed bit after the counter reaching counting 7 bits. Also it includes the strobe output that tells the user that his new data is now on the bus. It is displayed after one clock cycle using a simple D Flip Flop.



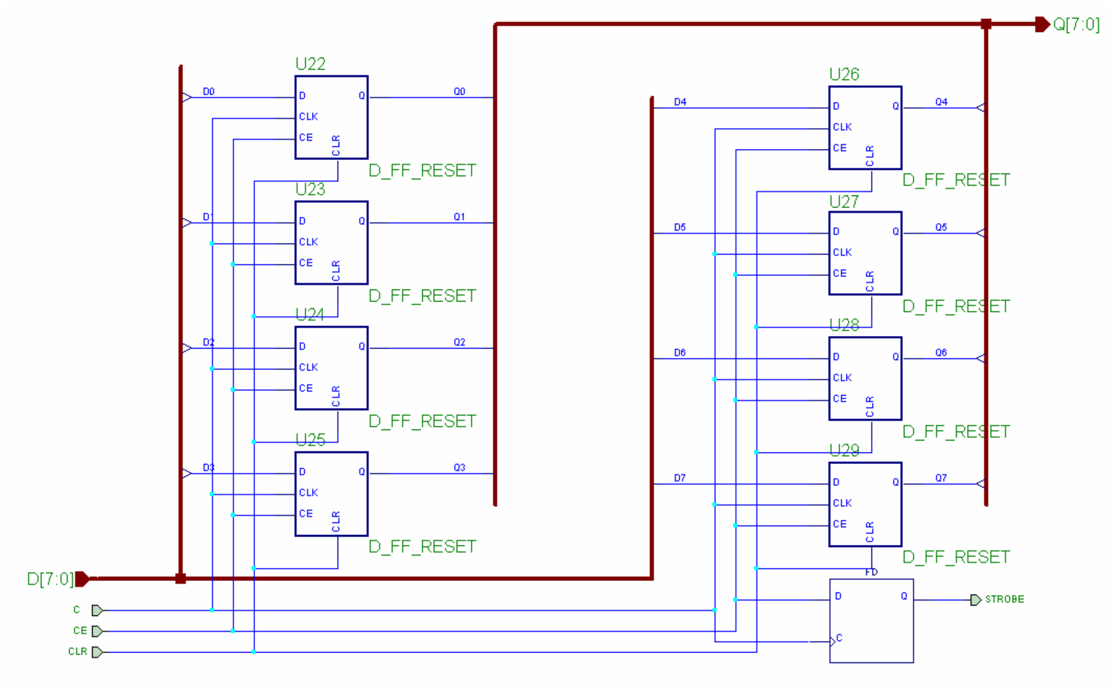


Figure 10: Parallel load register logic circuit

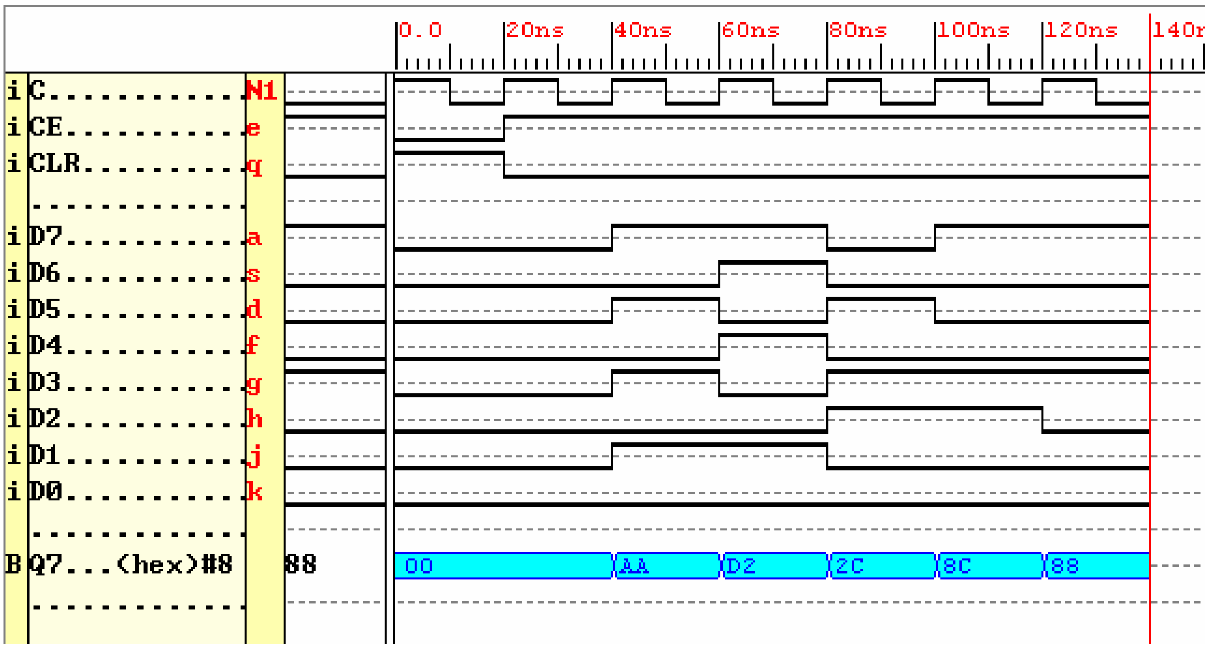
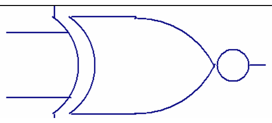


Figure 11: Timing diagram for the parallel load register

Parity Logic

The parity is inputted to the circuit to decide witch party should be used for the serial input packet, for instance, let 0 means choosing the odd parity and 1 means choosing the even parity.

Parity chosen	Serial Input bit	Parity bit added	Logic
0 (odd)	0	1	
0 (odd)	1	0	
1 (even)	0	0	
1 (even)	1	1	XNOR

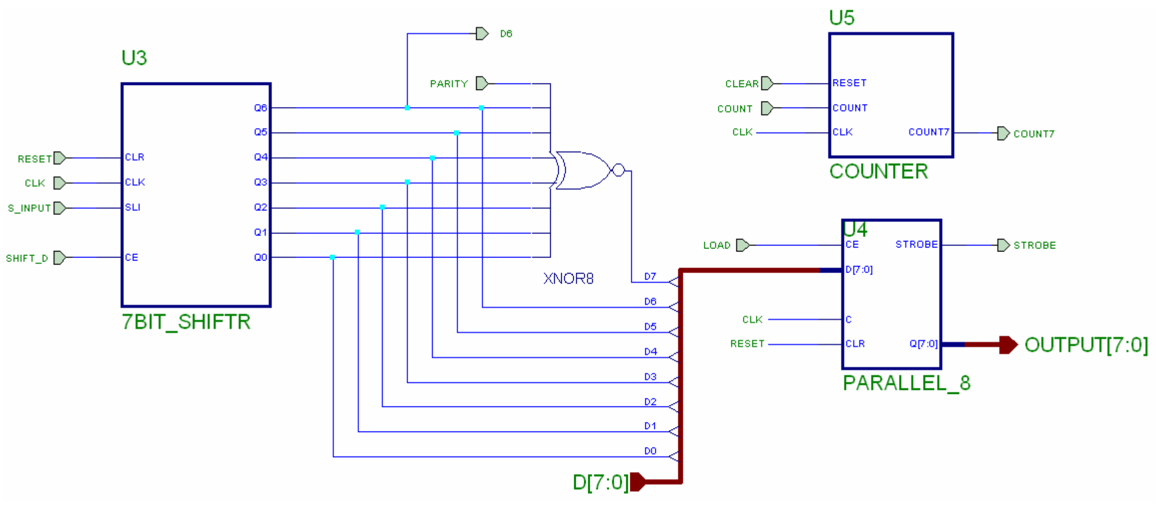


Figure 12: Data Path Block Diagram

Control Unit, Required Logic

To implement this unit, we need to design a state diagram and extract the logic required from the state designed. General AND, OR gates are used here. Besides, a D Flip Flop w/ RESET & CE is used.

States Logics:

The logic used in this project is stated as follow:

$$A^{+} = \overline{A}.B.Q6 + A.B.Q6 + A.\overline{B}.C7 + A.\overline{B}.\overline{C7} = B.Q6 + A.\overline{B}$$

$$B^{+} = \overline{A}.\overline{B}.Q6 + A.\overline{B}.C7$$

$$CLR\_Counter = \overline{A}.B.Q6 + A.B.Q6 = B.Q6$$

$$Count = A.\overline{B}.\overline{C7}$$

$$Load = A.\overline{B}.C7$$

A^{+}
D- Flip Flop
A

State Diagram:

The state diagram is shown below. There are four states in the design. This requires using two D Flip Flops.

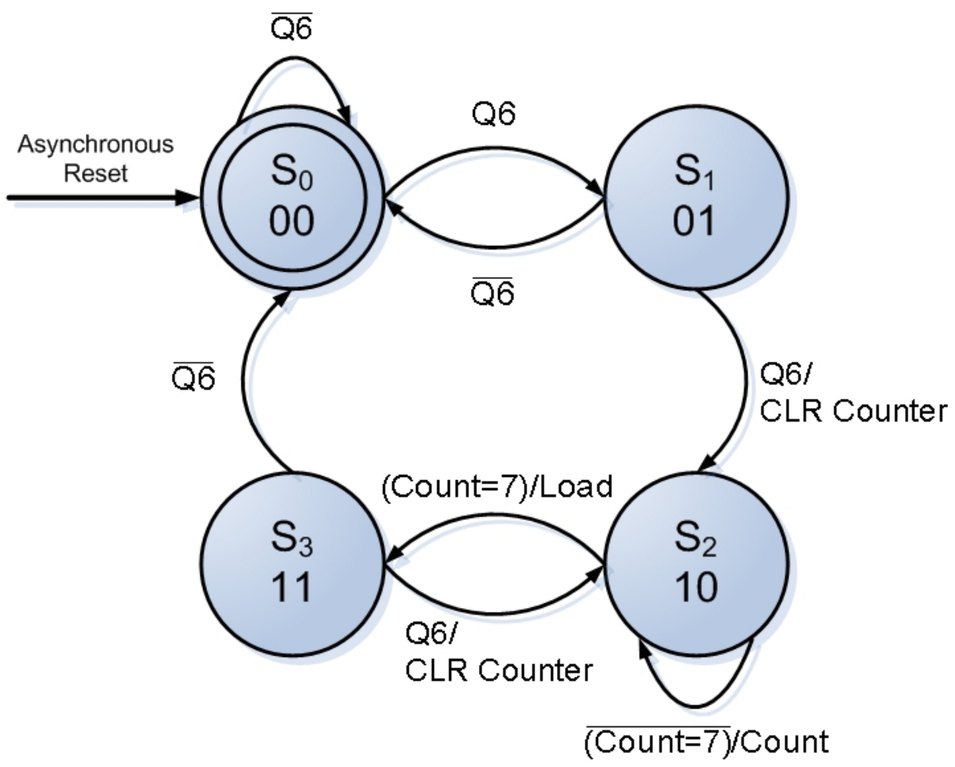


Figure 13: Control Unit State Diagram

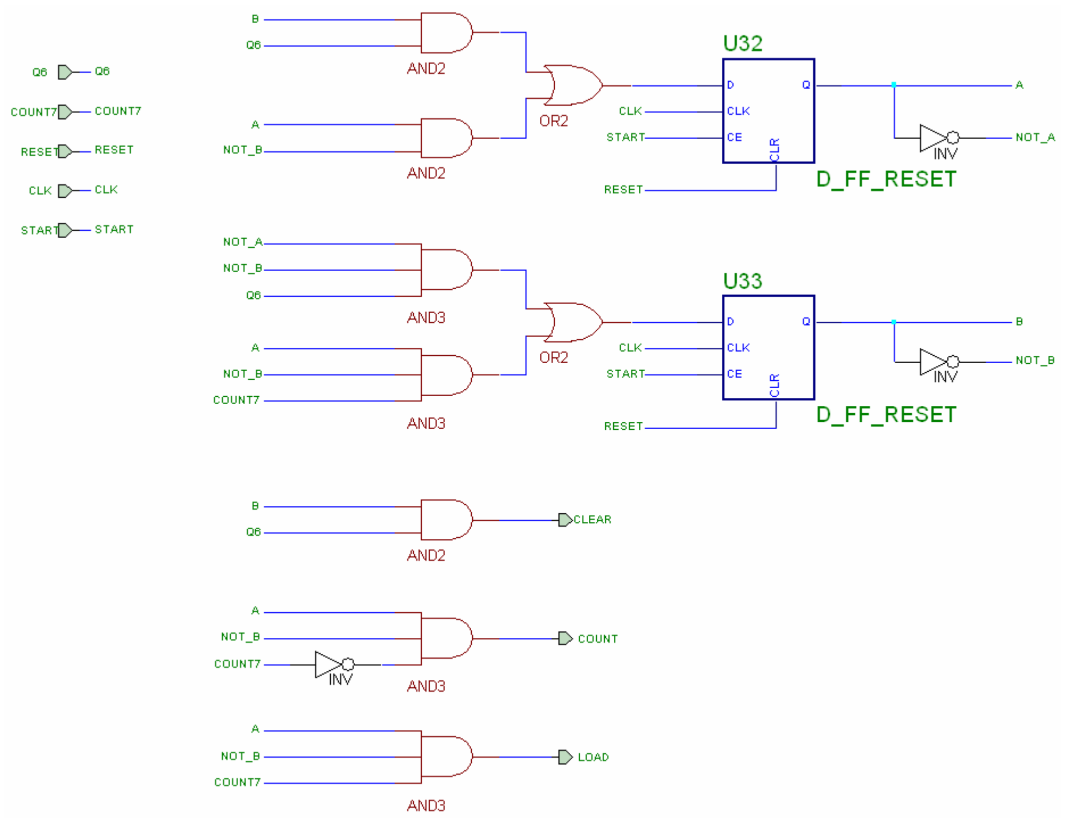


Figure 14: Control Unit Block Diagram

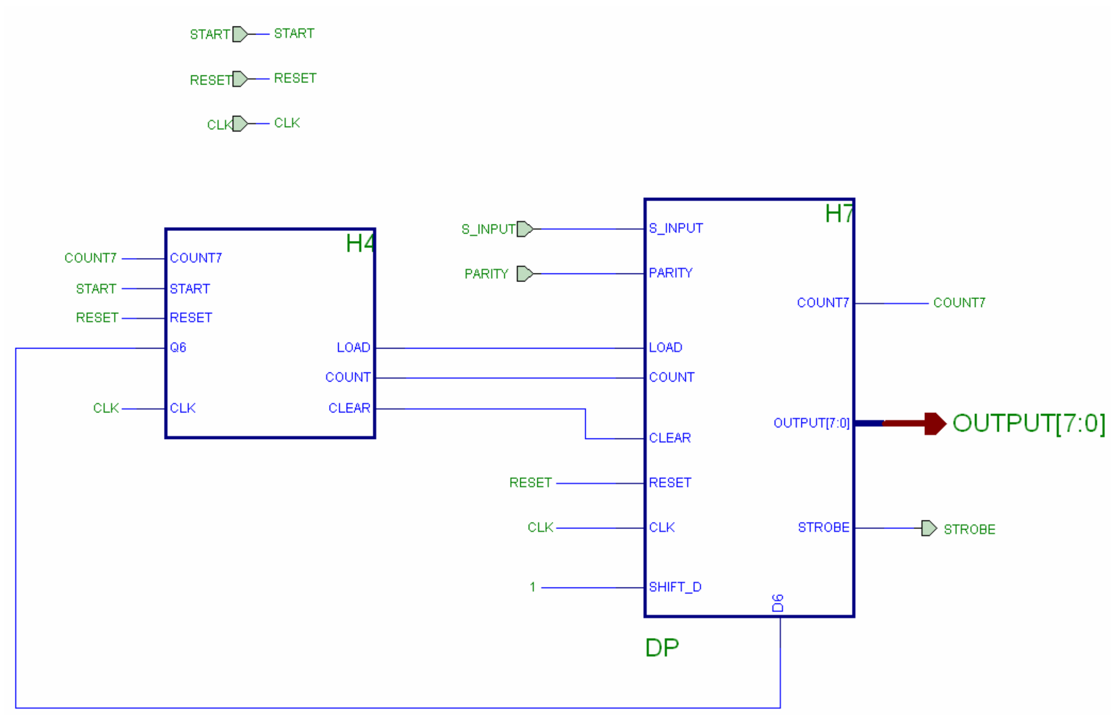


Figure 15: Serial to Parallel Block Diagram

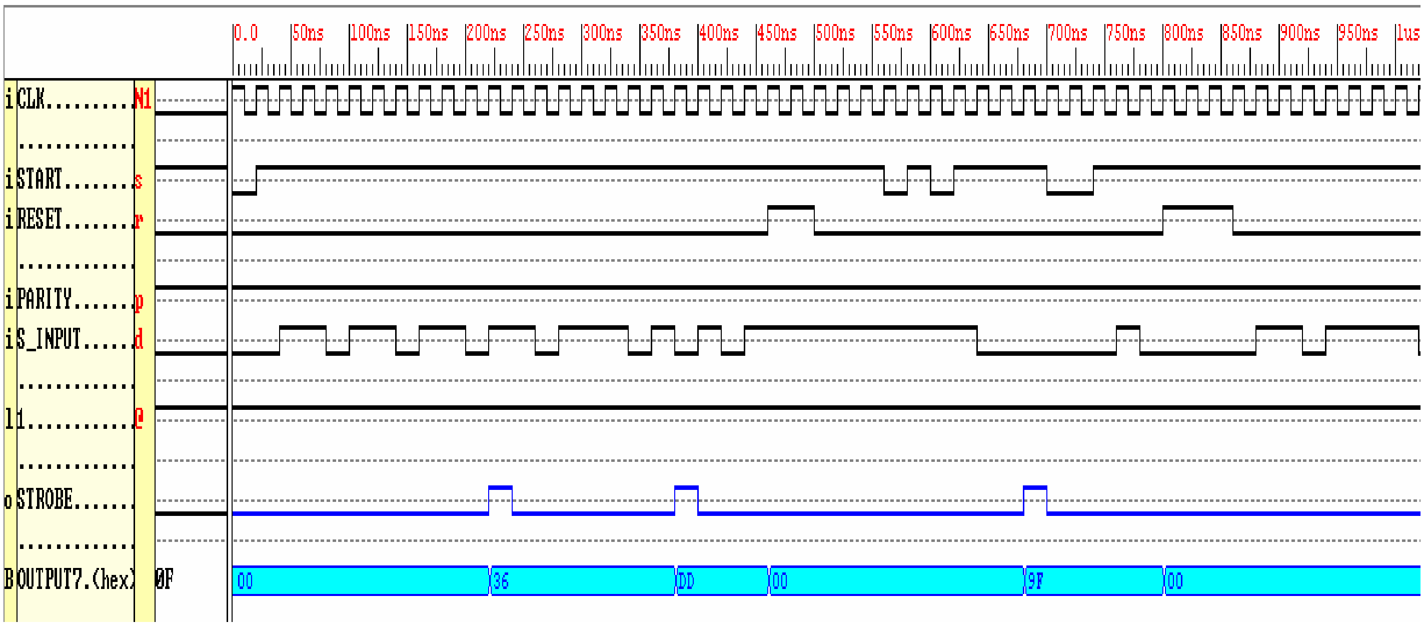


Figure 16: SERIAL_TO_PARRALLEL Final Simulation (1)

Conclusion:

From the above description we can conclude here the final output of the design. The simulation is done by the Xilinx simulator.

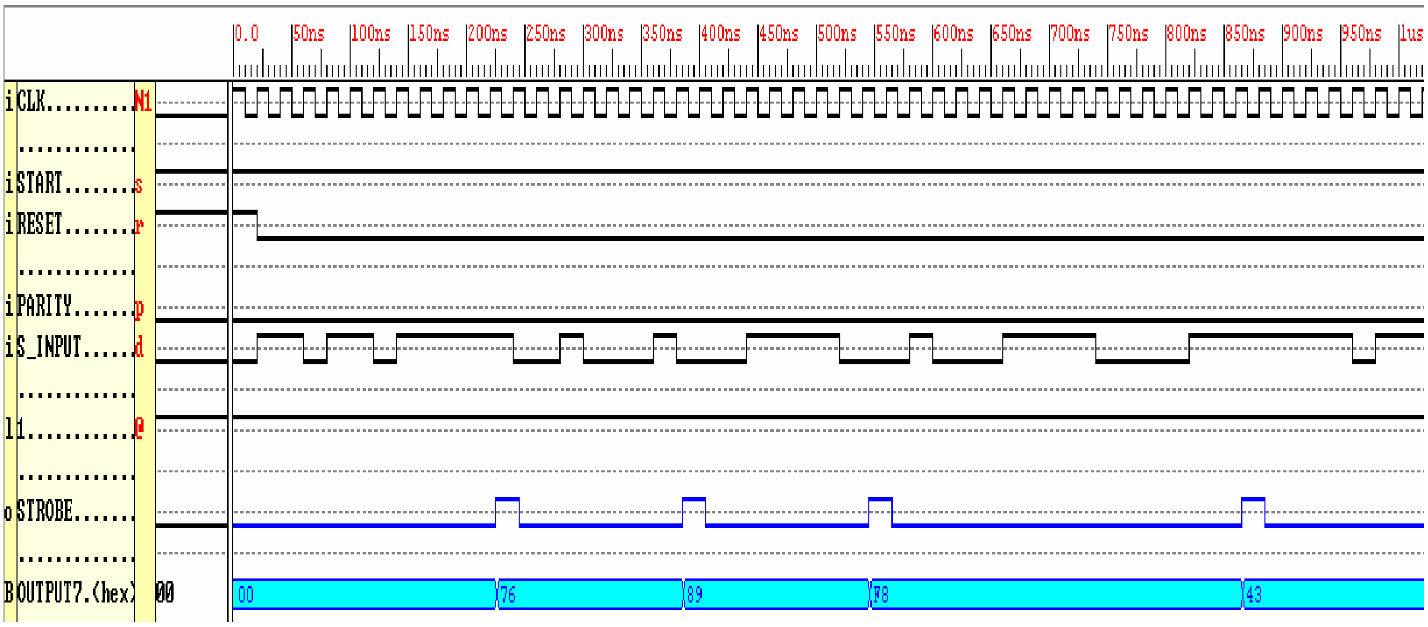


Figure 18: SERIAL_TO_PARRALLEL Final Simulation (2)

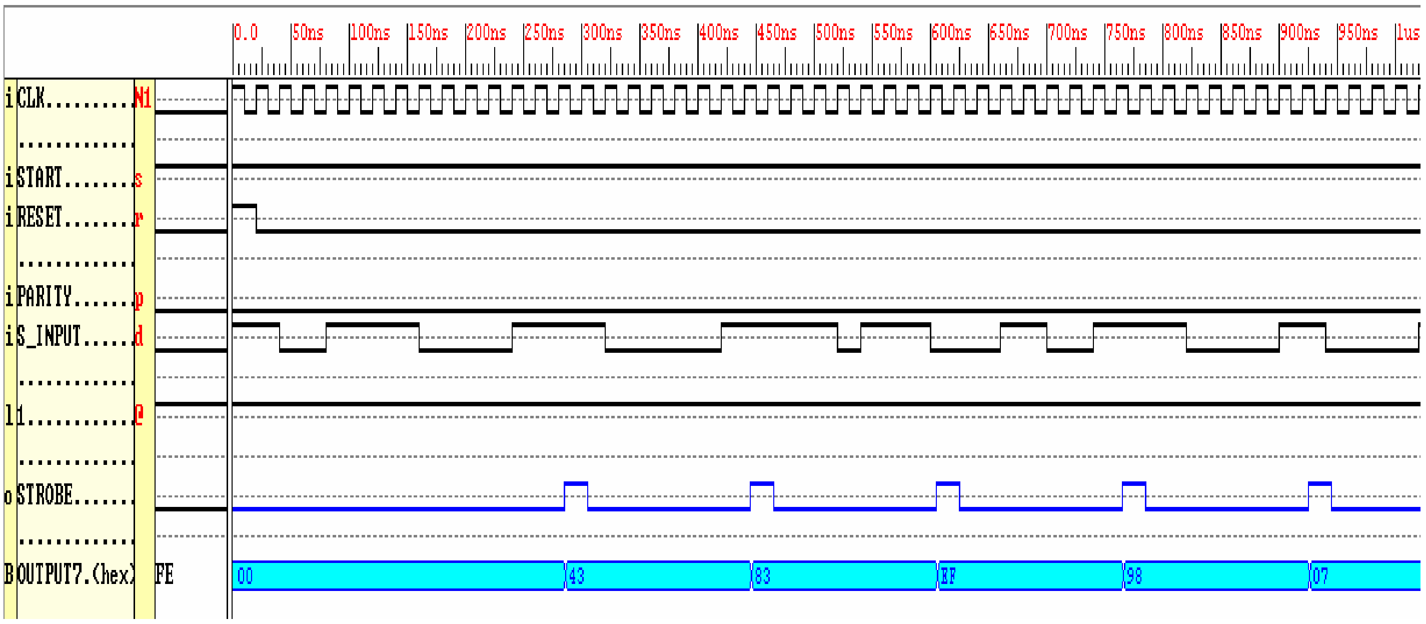


Figure 17: SERIAL_TO_PARRALLEL Final Simulation (3)