SPH 336
COUMPUTING LAB II REPORT

D-LATCH FLIP-FLOP

GROUP MEMBERS

I39/2219/2013 LINCOLN SIMBA ABUGA
I39/2215/2013 AKELLO LAZARUS OTIENO
I39/2214/2013 OMBAE SETH OKELLO
I39/2220/2013 ABEL EVANS AHENDA

OBJECTIVES

To Design a D-LATCH flip flop using System C code.

INTRODUCTION

I. What it is

A D latch flip flop is an electronic device that can be used to store one bit of information and is used to capture, the logic level which is present on the Data line when the clock input is high.

II. Its application

It can be used in any of the following areas :

a. Data Storage
b. Data Transfer
c. Counter
d. Frequency Division

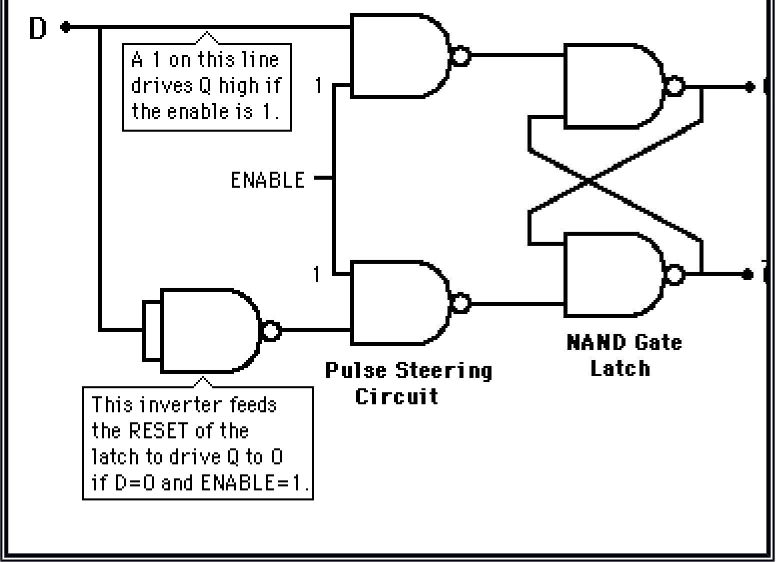
FUNCTIONALITY

I. Implementation

A D Latch flip flop can be implemented in one of the following ways:

D Flip-Flop from NAND Latch

The output Q will track the input D so long as the flip-flop remains enabled.

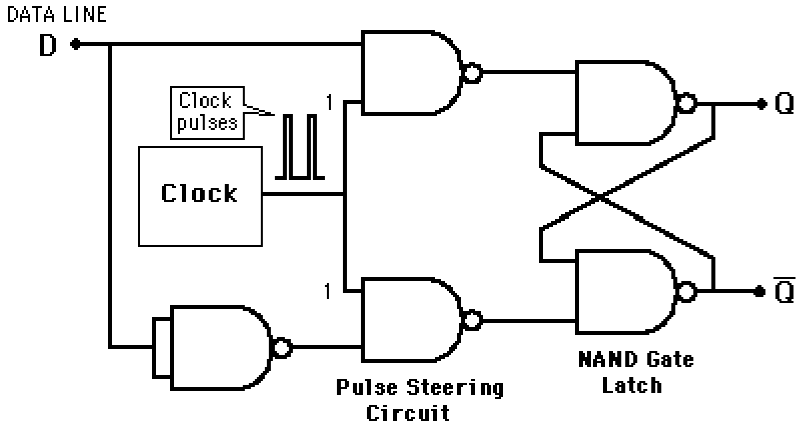


A 1 on this line drives Q high if the enable is 1.

This inverter feeds the RESET of the latch to drive Q to 0 if D=0 and ENABLE=1.

Clocked D Flip-Flop

A D flip flop constructed from a NAND-latch .



II. Truth Table

D Flip-flop

Symbol

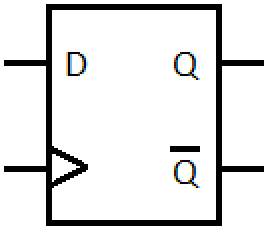


Table of truth:

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

COMPUTATION

With the above knowledge, we did a system c code with the following header files.

a. DLatchCode.h-that does the following
   - Performs the actual D latch operation
   - It changes the value of the output to that of the data input otherwise the output does not change

b. DLatchDriver.h which Drives values to input and enable clock of the D latch

c. DlatchMonitor.h which Probes the input, clock and outputs for values

d. DlatchCode.cc which contains the main folder

The following are the codes

```
/*
 * DLatchCode.h
 *
 *  Created on: Nov 30, 2015
 *      Author: laz
 */

#ifndef DLATCHCODE_H_
#define DLATCHCODE_H_
/**
```

```
 * Performs the actual D latch operation
 * It changes the value of the output to that of the data input otherwise thw output does not
change
 */
#include <systemc>
#include <systemc.h>

SC_MODULE(DLatchCode){
        sc_in <bool> data;
        sc_in <bool> enable;
        sc_out <bool> q;

        SC_CTOR(DLatchCode){
                SC_METHOD(theCode);
                sensitive << data << enable;
        }

        void theCode(){
                //output q is equal to data if enable is 1
                if (enable)q = data;
        }
};



#endif /* DLATCHCODE_H_ */
```

```
/*
 * DLatchDriver.h
 *
 *  Created on: Nov 30, 2015
 *      Author: laz
 */

#ifndef DLATCHDRIVER_H_
#define DLATCHDRIVER_H_
/**
 * Drives values to input and enable clock of the D latch
 */

#include <systemc>
#include <systemc.h>

SC_MODULE(DLatchDriver){
        sc_out <bool> input, clock;
```

```
    SC_CTOR(DLatchDriver){
        SC_THREAD(driveInputPort);
        sensitive<<input;
        SC_THREAD(driveClock);
        sensitive<<clock;
    }

    //drives data to the input port
    void driveInputPort(){
        while(1){
            input = 0;
            wait(5,SC_NS);
            input = 1;
            wait(5,SC_NS);
        }
    }

    //drives clock to the enable port
    void driveClock(){
        while(1){
            clock = 0;
            wait(3,SC_NS);
            clock = 1;
            wait(4,SC_NS);
        }
    }
};


#endif /* DLATCHDRIVER_H_ */







/*
 * DlatchMonitor.h
 *
 *  Created on: Nov 30, 2015
 *      Author: laz
 */

#ifndef DLATCHMONITOR_H_
#define DLATCHMONITOR_H_
/**
 * Probes the input, clock and outputs for values
 */
#include <systemc>
#include <systemc.h>

SC_MODULE(DLatchMonitor){
    sc_in <bool> m_in, m_clock,  m_q;

    SC_CTOR(DLatchMonitor){
        SC_METHOD(monitorDLatch);
        sensitive<<m_in<<m_q;
```

```
			dont_initialize();
		}

		void monitorDLatch(){
			cout<<sc_time_stamp()<<"   input is "<< m_in <<"clock edge "<<m_clock<< "
		output is"<<m_q<<endl;
    }

};


#endif /* DLATCHMONITOR_H_ */
```

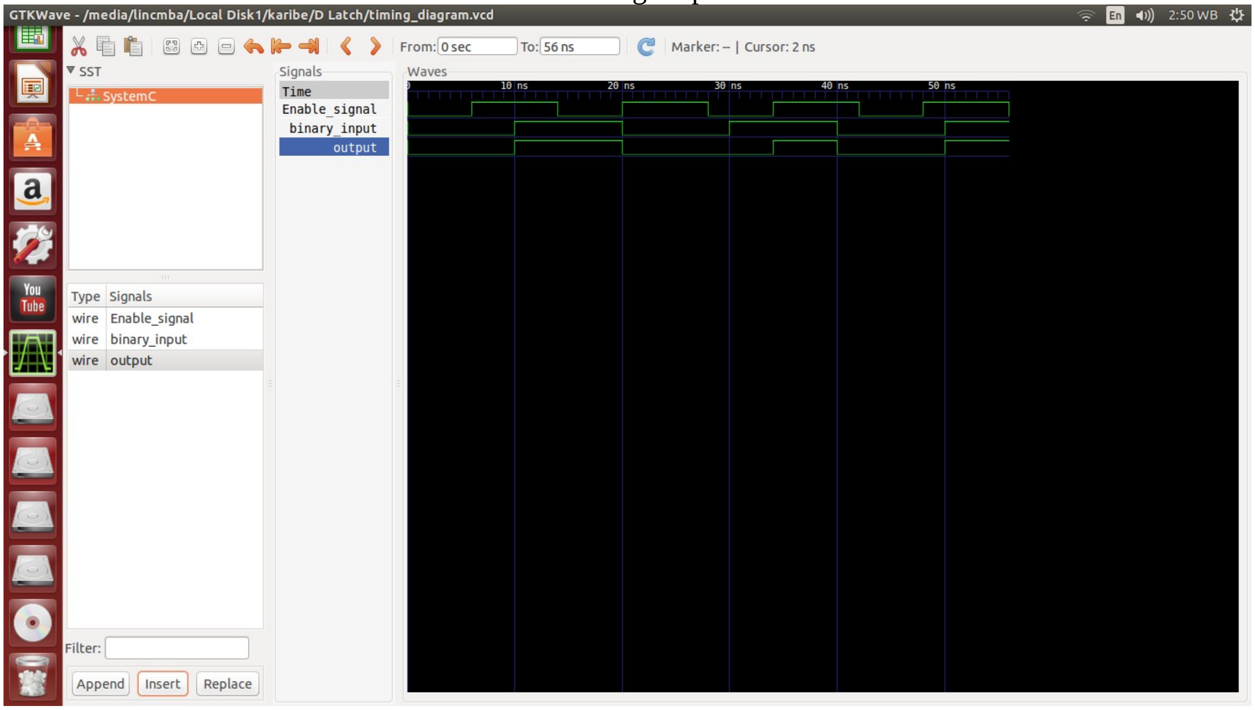
```
/*
 * DLatchCode.cc
 *
 *  Created on: Nov 30, 2015
 *      Author: laz
 */
/**
 * Contains the main file
 */
#include <systemc>
#include <systemc.h>
#include "DLatchMonitor.h"
#include "DLatchCode.h"
#include "DLatchDriver.h"

int sc_main(int argc, char *argv[]){
		//signals at the input and output
		sc_signal <bool> data, clk, q;

		//module instances
		DLatchCode latch("d latch");
		DLatchMonitor mon("monitor instance");
		DLatchDriver dr("driver instance");

		dr.input(data);
		dr.clock(clk);
		mon.m_in(data);
		latch.data(data);

    mon.m_clock(clk);
		latch.enable(clk);

		latch.q(q);
		mon.m_q(q);

		//create a trace file with nanosecond resolution
		sc_trace_file *tf;
		tf = sc_create_vcd_trace_file("timing_diagram");
		tf->set_time_unit(0.5, SC_NS);
```

```
    //trace the signals interconnecting modules
    sc_trace(tf, data, "binary_input"); // signals to be traced
    sc_trace(tf, clk, "Enable_signal");
    sc_trace(tf, q, "output");

    //run a simulation for 20 systemc nano-seconds
    if( !sc_pending_activity() )sc_start(30,SC_NS);
    //close the trace file
    sc_close_vcd_trace_file(tf);

    return 0;
}
```

We built and executed the above code.

RESULTS

We created VCD trace files and obtained the following output



GTKWave - /media/lincmba/Local Disk1/karibe/D Latch/timing_diagram.vcd

From: 0 sec To: 56 ns Marker: -- | Cursor: 2 ns

Type	Signals
wire	Enable_signal
wire	binary_input
wire	output

CONCLUSION

Our objective of designing a D Latch flip flop using system c code was accomplished.

This is evidenced by the modules we created with the header files namely DLatchCode.h, DLatchDriver.h, DlatchMonitor.h, DlatchCode.cc.

Then after we obtained the timing diagrams above hence it was a successful tutorial.