**PROBLEM STATEMENT**

This experiment examines the hierarchical construction and functionality of a basic eight bit register using Verilog primitive gates and basic logical constructs. The constructed register is made from eight instances of a D flip-flop module which in turn is made from multiple instances of a SR latch.

**KEYWORDS:** D flip-flop, SR, latch, NAND, AND, Logical High, Logical Low, Intrinsic Delay, fanin, fanout, Capacitive loading, register, Multiplexer, NAND, AND, OR, NOT.

1. **INTRODUCTION**

A register is a collection of multiple flip-flops designed to retain a binary value in synchronicity with a generated clock signal. As such, each of a register’s flip-flops are sequential circuits which utilize latch modules that possess feedback between the Q and Q~ terminals. This feedback enables the latching circuit to retain a passed binary value which can be read or set depending on the input values passed to the latch circuit.

Latch circuits are constructed from fundamental logic gates and possess an associated propagation delay between their input and output signals. This delay can be modeled as a sum of two types of component delay, the Intrinsic delay of the latch’s gate and the capacitive loading delay associated with the amount of gates driven by one gate’s output. A gate’s intrinsic delay is associated with the time it takes for a signal to traverse the internal transistors of that gate. As inputs increase the number of transistors within a gate must also rise to switch between the intermediate logical values. The modeled intrinsic gate delays as a linear approximation of their inputs are displayed below in table 3.1 for all gates used in this experiment.

Table 3.1 Intrinsic Gate Delay

|  |  |
| --- | --- |
| Inputs | Intrinsic Delay |
| 1 | 1 ns |
| 2 | 2ns |
| 3 | 3ns |

Beyond the intrinsic delay, there is also a delay caused by the inherit capacitance between gates within the latch. This capacitive loading delay is a result of the increased time needed to charge or discharge inferred capacitance between gates. As a gate whose output is tied to multiple inputs will in turn form multiple capacitances with those inputs, the delay time increases as a function of the number of outputs driven by a gate. The associated delay between a gate’s fanout and capacitive loading are shown in Table 3.2. No input, or fanin, capacitive loading was accounted in this experiment.

Table 3.2 Fin/Fout Gate Delay

|  |  |
| --- | --- |
| Gate fanout | Capacitive loading |
| 1 | 0.4 ns |
| 2 | 0.7 ns |
| 3 | 1.0 ns |
| Primary Output | 5 ns |

The basic latching circuit consists of two logical gates whose outputs can control the input of the opposing gate. While a latch may be constructed from multiple types of gates, the NAND gate latch is utilized in this experiment. Specifically, a two input SR NAND latch is used to construct all sequential logic within the register. Figure 3.1 below shows the basic latch construct.

Figure 3.1 2-input SR Latch Schematic



From this schematic, a truth table was constructed. Due to the properties of NAND logic, only cases where s0,s1 and r0,r1 are like values are shown. Any input that is logical low within a NAND gate results in an output which is logical high. . A truth table for the meaningful cases of the SR latch is shown below in Table 3.3.

Table 3.3 2-input SR Latch Truth Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Op | S0+S1 | R0+R1 | - | - |  |  |
| IND | 0 | 0 | 0 | 0 | 1?0 | 1?0 |
| IND | 0 | 0 | 0 | 1 | 1?0 | 1?0 |
| IND | 0 | 0 | 1 | 0 | 1?0 | 1?0 |
| IND | 0 | 0 | 1 | 1 | 1?0 | 1?0 |
| Set | 0 | 1 | 0 | 0 | 1 | 0 |
| Set | 0 | 1 | 0 | 1 | 1 | 0 |
| Set | 0 | 1 | 1 | 0 | 1 | 0 |
| Set | 0 | 1 | 1 | 1 | 1 | 0 |
| Reset | 1 | 0 | 0 | 0 | 0 | 1 |
| Reset | 1 | 0 | 0 | 1 | 0 | 1 |
| Reset | 1 | 0 | 1 | 0 | 0 | 1 |
| Reset | 1 | 0 | 1 | 1 | 0 | 1 |
| Hold | 1 | 1 | 0 | 0 | - | - |
| Hold | 1 | 1 | 0 | 1 | - | - |
| Hold | 1 | 1 | 1 | 0 | - | - |
| Hold | 1 | 1 | 1 | 1 | - | - |

Using multiple instances of the above latch, it is possible to construct a sequential synchronous logic circuit that acts as a D FF. The inputs considered consist of data input Data, reset signal Clear and a clock signal Clock. The outputs of the d ff are denoted Qd and Qd~. The intended functionality of the basic D flip-flop is described in table 3.4 below.

Table 3.4 D flip-flop Truth Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data | Clear | Clock |  |  |
| X | 0 | X | 0 | 1 |
|  | 1 | ↑ |  |  |
|  | 1 |  | - | - |
| X = Don’t Care | ↑ = Positive Edge |  | | |

From this table, the flip-flop is constructed using three SR latches as described in Figure 3.1, table 3.3. Each input signal passes through a buffer circuit to account for any delay associated with latch feedback. The constructed D flip-flop module is shown below in figure 3.2. The unused inputs of the S~ and Q gates are tied to logical high to preserve input values.

Figure 3.2 D flip-flop Schematic



From the D flip-flop described above, a sequential eight bit register is to be built. To handle the register functions, multiplexing logic external to the sequential flip-flop components must be implemented to allow for storing a value. A simple multiplexing circuit to select between two inputs is used to handle register input logic independently of the flip-flops. This module is described in Figure 3.3 and Table 3.5 respectively

Figure 3.3 Gate Schematic for 2-Input MUX



*Table 3.5* Truth Table for 2-Input Multiplexer

|  |  |  |  |
| --- | --- | --- | --- |
| *A* | *B* | *SEL* | OUT |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

From these two modules, a register is constructed. The intended functionality of the register is shown below in Table 3.6. The bus data input and register outputs are each eight bits wide while all other inputs are a single bit.

Table 3.6 8-bit Register Truth Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data | Enable | Reset | Clock | Reg\_out [7:0] |
|  | 0 | X | X |  |
|  | 0 | 0 | X | 0 |
|  | 1 | 0 | X | 0 |
|  | 1 | 1 | ↑ |  |
|  | 1 | 1 | ↓,0,1 |  |

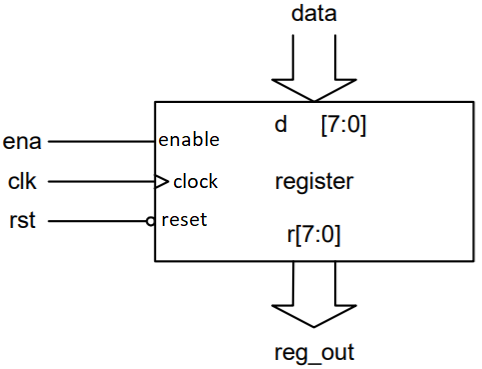
To represent each bit, an iterative module was constructed from a D flip-flop and a simple 2-1 multiplexer. This module is shown in figure 3.4 below. Eight instances of this module were cascaded to construct the register. The input *d[i]* represents a bit of the data input while the output *r[i]* represents a bit of the register output.

Figure 3.4 Iterative Register Module



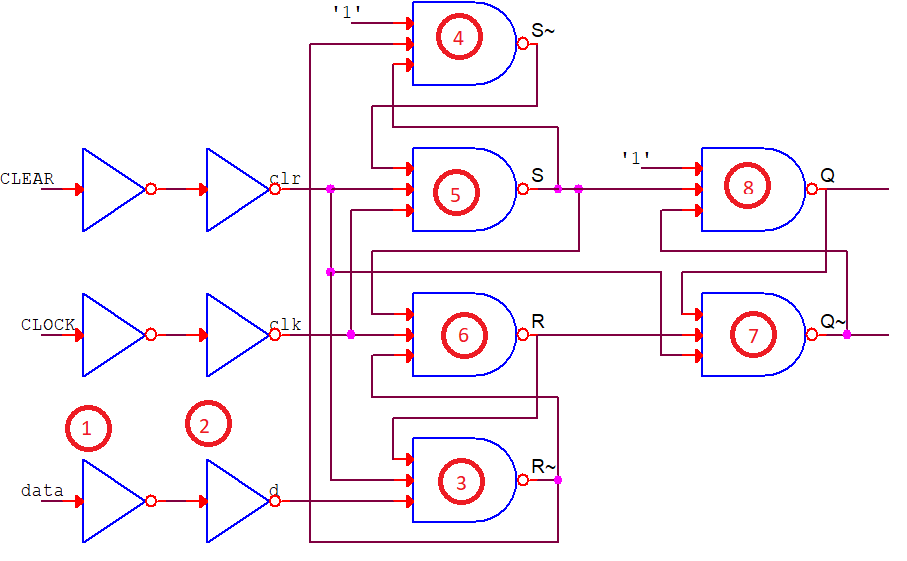
The high level symbol for the constructed register is shown below in Figure 3.5. This block consists of eight instances of the iterative module described above in Fig 3.

Figure 3.5 High Level Register Block



The described register’s theoretical operating frequency is considered as limited by its component modules. An examination of schematics in figures 3.3 and 3.2 show that the slowest path rests in how a signal traverses the D flip-flop. This path is shown below in Figure 3.5.

Figure 3.6 Critical Path for 8-bit Register



The critical path of the circuit’s combinational multiplexer logic is inspected to be significantly less than that of the flip-flop and is not considered in this experiment.

This frequency is derived from the associated delay models used in Tables 3.1 and 3.2. the path delay is represented in Eq 3.1(1) below.

(1)

The associated critical path period is calculated using each gates intrinsic and capacitive loading delays.

(1)

This TCP is representative of the minimum delay necessary to operate the circuit with integrity. An expression for the maximum operating frequency is then found using the inverse relationship described by Eq 3.1(2).

(2)

The calculated fmax is then determined for the flip-flop and subsequently the register.

(4)

1. **METHODOLOGY**

To begin the experiment, the data requirements for this register were first considered. Using the register functions described in Table 3.6. The data burden of the register’s behavior under different testing conditions was formulated.

To test the register module, a heuristic approach is adopted. The register operation for each of the three inputs is observed for all meaningful cases. This includes when the register is loaded with a value, when it is reset, and when it is disabled. To observe register operation while disabled, data and reset are altered while the enable input is low. To observe the behavior of the register when a value is loaded, the register is enabled and the reset is disabled (set to 1) while the data is toggled to different values. Finally to observe the behavior of the register when it is reset, the register is enabled and the reset input is enabled (set to 0) while the data input is changed. To determine the maximum operating frequency, the clock period was decreased from the default frequency until a fault was observed.

Each hierarchal component of the register was created and instantiated in the design. The SR latch described in Figure 3.1 and Table 3.3 was modeled using primitive Verilog gates. This module is shown blow in section 3.2 as Module 3.1. Next, the D flip-flop was constructed using three instantiations of this latch module in addition and primitive inverter gates to behave as described in Figure 3.2 and Table 3.4. This module is shown blow in section 3.2 as Module 3.2. A multiplexing module as described in Figure 3.3 and table 3.5 was constructed using primitive gates. This module is shown blow as Module 3.3. Finally the register itself was constructed from eight respective iterations of the D flip-flop and multiplexer modules. This register module is shown below Module 3.4. To test these modules, a testbench module was constructed containing a clock generator module, testbench code, default delay values and simulation constraints. The testbench module can be seen below as Module 3.5.

The testbench and behavioral modules must then be compiled using the Verilog Compile Simulator tool (VCS). If compiled with no warnings or errors, the behavioral simulation will be run and the output recorded. During simulation, the default frequency was increased beyond the maximum theoretical frequency to test for circuit faults. Figure 3.6 shows the captured behavioral waveforms of all recorded experimental values under this case during simulation. This output is tabulated in Table 3.6. Figure 3.7 and 3.8 below show the circuit behavior at the maximum possible operating frequency (to the nearest MHz) and the first inoperable frequency.

1. **MODULE FILES & SIMULATION RESULTS**

*Module 3.1*—SR\_Latch2.v

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* \*\*\*

\*\*\* EE 526 L Experiment #3 Kyle E. Keislar, Spring, 2020 \*\*\*

\*\*\* \*\*\*

\*\*\* Hierarchical Modeling \*\*\*

\*\*\* \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: SR\_Latch2.v \*\*\*

\*\*\* Author: Kyle E. Keislar \*\*\*

\*\*\* Date: 02/22/2020 \*\*\*

\*\*\* Version: 1.0 \*\*\*

\*\*\* Revised: \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Module Description: basic 2-input SR latch \*\*\*

\*\*\* Sets when s0 and s1 are high \*\*\*

\*\*\* Resets when r0 and r1 are high \*\*\*

\*\*\* holds if any two of r0, r1, s0 or s1 are high \*\*\*

\*\*\* Indeterminate behavior when all inputs are high\*\*\*

\*\*\* \*\*\*

\*\*\* \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1ns **/** 10ps

// Module Declaration

**module** SR\_Latch2**(**Q**,** //FF output

Qnot**,** //FF output

s0**,** //set nand gate 0 input

s1**,** //set nand gate 1 input

r0**,** //reset nand gate 0 input

r1**);** //reset nand gate 1 input

//Module Parameters

**parameter** NAND\_DEL\_S **=** 1**;**

**parameter** NAND\_DEL\_R **=** 1**;**

// I/O port assignment

**output** **wire** Q**,** Qnot**;**

**input** **wire** s0**,** s1**,** r0**,** r1**;**

//Latch implementation

**nand** **#**NAND\_DEL\_S **(**Q**,**Qnot**,**s0**,**s1**);** //Q with Qnot gate feedback

**nand** **#**NAND\_DEL\_R **(**Qnot**,**Q**,**r0**,**r1**);** //Qnot with Q gate feedback

**endmodule**

*Module 3.2*—dff.sv

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* \*\*\*

\*\*\* EE 526 L Experiment #3 Kyle E. Keislar, Spring, 2020 \*\*\*

\*\*\* \*\*\*

\*\*\* Hierarchical Modeling \*\*\*

\*\*\* \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: d\_ff.sv \*\*\*

\*\*\* Author: Kyle E. Keislar \*\*\*

\*\*\* Date: 02/15/2020 \*\*\*

\*\*\* Version: 1.0 \*\*\*

\*\*\* Revised: \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Module Description: Basic DFF \*\*\*

\*\*\* D Flip-flop constructed from two input SR Latch \*\*\*

\*\*\* as described by SR\_Latch2.v. Positive edge \*\*\*

\*\*\* triggered device with the following truth table \*\*\*

\*\*\* \*\*\*

\*\*\* Data clear clock Q ~Q \*\*\*

\*\*\* X 0 + 0 1 \*\*\*

\*\*\* 0 1 + 0 1 \*\*\*

\*\*\* 1 1 + 1 0 \*\*\*

\*\*\* X 1 ~+ Q ~Q \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1ns **/** 10ps

// Module Declaration

**module** dff**(**q**,** qbar**,** clock**,** data**,** clear**);**

//Module Parameters

//Intrinsic Delay definitions

**parameter** time\_delay\_1**=**1**;**//intrinsic 1 input

**parameter** time\_delay\_2**=**2**;**//intrinsic 2 input

**parameter** time\_delay\_3**=**3**;**//intrinsic 3 input

//Capacitive loading definitions

**parameter** fout\_1**=**0.4**;**//fanout of 1

**parameter** fout\_2**=**0.7**;**//fanout of 2

**parameter** fout\_3**=**1.0**;**//fanout of 3

**parameter** fout\_po**=**5**;**//primary output delay

// I/O port assignment

**output** **reg** q**,** qbar**;**

**input** clock**,** data**,** clear**;**

//Internal Signals

**wire** clr**,** cbar**,** clk**,** clkbar**,** d**,** dbar**,**sbar**,**s**,**rbar**,**r**;**

**wire** Q**,**Qbar**;**

**wire** sbars1**,**qs1**;** //wires for unused inputs of sbar&Q

//Delay Buffer

**not** **#(**time\_delay\_1**+**fout\_1**)** **(**cbar**,**clear**);** //intrinsic 1 fanout 2

**not** **#(**time\_delay\_1**+**fout\_1**)** **(**clkbar**,**clock**);**//intrinsic 1 fanout 1

**not** **#(**time\_delay\_1**+**fout\_1**)** **(**dbar**,**data**);**//intrinsic 1 fanout 1

**not** **#(**time\_delay\_1**+**fout\_3**)** **(**clr**,**cbar**);** //intrinsic 1 fanout 3

**not** **#(**time\_delay\_1**+**fout\_2**)** **(**clk**,**clkbar**);**//intrinsic 1 fanout 2

**not** **#(**time\_delay\_1**+**fout\_1**)** **(**d**,**dbar**);**//intrinsic 1 fanout 2

**assign** sbars1**=** 1'b1**;** //tie unused inputs to 1

**assign** qs1**=** 1'b1**;** //A &~ 1 = A

//Latches

SR\_Latch2 **#(**time\_delay\_3**+**fout\_1**,**time\_delay\_3**+**fout\_3**)** SR1**(**sbar**,**s**,**rbar**,**sbars1**,**clr**,**clk**);** //x connected to s1 of sbar

//Sbar: 3 inputs 1 driven gate

//S: 3 inputs 3 driven gates

SR\_Latch2 **#(**time\_delay\_3**+**fout\_2**,**time\_delay\_3**+**fout\_2**)** SR2**(**r**,**rbar**,**s**,**clk**,**clr**,**d**);**

//Rbar: 3 inputs 2 driven gate

//R: 3 inputs 2 driven gates

SR\_Latch2 **#(**time\_delay\_3**+**fout\_po**,**time\_delay\_3**+**fout\_2**)** SR3**(**Q**,**Qbar**,**s**,**qs1**,**clr**,**r**);** //y connected to s1 of Q

//Qbar: 3 inputs Primary output

//Q: 3 inputs drives 2 gates (qbar unused as output port)

//Synchronous Behavior

**always** **@(posedge** clk**)** **begin**

q**=**Q**;**

qbar**=**Qbar**;**

**end**

**endmodule**

*Module 3.3*—2MUX\_1.v

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* \*\*\*

\*\*\* EE 526 L Experiment #3 Kyle E. Keislar, Spring, 2020 \*\*\*

\*\*\* \*\*\*

\*\*\* Hierarchical Modeling \*\*\*

\*\*\* \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: MUX2\_1.v \*\*\*

\*\*\* Author: Kyle E. Keislar \*\*\*

\*\*\* Date: 02/19/2020 \*\*\*

\*\*\* Version: 1.1 \*\*\*

\*\*\* Revised: 02/19/2019 \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Module Description: 2 Input Multiplexer with Control Line SEL \*\*\*

\*\*\* \*\*\*

\*\*\* A B SEL OUT \*\*\*

\*\*\* 0 0 0 0 \*\*\*

\*\*\* 0 0 1 0 \*\*\*

\*\*\* 0 1 0 0 \*\*\*

\*\*\* 0 1 1 1 \*\*\*

\*\*\* 1 0 0 1 \*\*\*

\*\*\* 1 0 1 0 \*\*\*

\*\*\* 1 1 0 1 \*\*\*

\*\*\* 1 1 1 1 \*\*\*

\*\*\* \*\*\*

\*\*\* Module Limitations: Combinatorial Inputs \*\*\*

\*\*\* Revision Notes: Added in intrinsic & capacitative delays for all\*\*\*

\*\*\* MUX gates \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1ns **/** 10ps

**module** MUX2\_1**(**OUT**,** A**,** B**,** SEL**);**

//Delay Parameters

//Intrinsic Delay definitions

**parameter** time\_delay\_1**=**1**;**//intrinsic 1 input

**parameter** time\_delay\_2**=**2**;**//intrinsic 2 input

**parameter** time\_delay\_3**=**3**;**//intrinsic 3 input

//Capacitive loading definitions

**parameter** fout\_1**=**0.4**;**//fanout of 1

**parameter** fout\_2**=**0.7**;**//fanout of 2

**parameter** fout\_3**=**1.0**;**//fanout of 3

**parameter** fout\_po**=**5**;**//primary output delay

//Port Declarations

**output** OUT**;**

**input** A**,** B**,** SEL**;**

//Internal Signals

**wire** A1**,** B1**,** SEL\_N**;**

//Netlist

**not** **#(**time\_delay\_1**+**fout\_1**)** **(**SEL\_N**,** SEL**);**

**and** **#(**time\_delay\_2**+**fout\_1**)** **(**A1**,** A**,** SEL\_N**);**

**and** **#(**time\_delay\_2**+**fout\_1**)** **(**B1**,** B**,** SEL**);**

**or** **#(**time\_delay\_2**+**fout\_1**)** **(**OUT**,** A1**,** B1**);**

**endmodule**

*Module 3.4*—register.sv

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* \*\*\*

\*\*\* EE 526 L Experiment #3 Kyle E. Keislar, Spring, 2020 \*\*\*

\*\*\* \*\*\*

\*\*\* Hierarchical Modeling \*\*\*

\*\*\* \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: register.sv \*\*\*

\*\*\* Author: Kyle E. Keislar \*\*\*

\*\*\* Date: 02/18/2020 \*\*\*

\*\*\* Version: 1.0 \*\*\*

\*\*\* Revised: \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Module Description: 8-bit Register \*\*\*

\*\*\* Eight bit wide register constructed from \*\*\*

\*\*\* synchronized leading edge DFFs. Able to hold \*\*\*

\*\*\* unsigned values from 0 to 255 or signed \*\*\*

\*\*\* values from -127 to 120. \*\*\*

\*\*\* \*\*\*

\*\*\* DATA ENA RST CLK R(i+1) \*\*\*

\*\*\* X 0 X X R(i) \*\*\*

\*\*\* X 1 0 X 0 \*\*\*

\*\*\* 0 1 0 + 0 \*\*\*

\*\*\* 1 1 0 + 1 \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1ns **/** 10ps

// Module Declaration

**module** register**(**R**,** CLK**,** DATA**,** ENA**,** RST**);**

//Intrinsic Delay definitions

**parameter** time\_delay\_1**=**1**;**//intrinsic 1 input

**parameter** time\_delay\_2**=**2**;**//intrinsic 2 input

**parameter** time\_delay\_3**=**3**;**//intrinsic 3 input

//Capacitive loading definitions

**parameter** fout\_1**=**0.4**;**//fanout of 1

**parameter** fout\_2**=**0.7**;**//fanout of 2

**parameter** fout\_3**=**1.0**;**//fanout of 3

**parameter** fout\_po**=**5**;**//primary output delay

//Module Parameters

// I/O port assignment

**output** **reg** **[**7**:**0**]** R**;**

**input** **[**7**:**0**]** DATA**;**

**input** CLK**,** ENA**,** RST**;**

**genvar** i**;**//used to generate 1 d ff for each regbit value passed

//Internal Signals

**wire** r\_mux**[**7**:**0**],**d\_in\_FF**[**7**:**0**],**qbar**[**7**:**0**];** //qbar not used as output

//Instantiation Loop

**for** **(**i **=** 0**;** i **<** 8**;** i **=** i **+**1**)** **begin**

//2:1 MUX Instantiations

MUX2\_1 **#(**time\_delay\_1**,**time\_delay\_2**,**time\_delay\_3**,**fout\_1**,**fout\_2**,**fout\_3**,**fout\_po**)**d0\_mux**(**d\_in\_FF**[**i**],**r\_mux**[**i**],**DATA**[**i**],**ENA**);**

//Register Instantiations

dff **#(**time\_delay\_1**,**time\_delay\_2**,**time\_delay\_3**,**fout\_1**,**fout\_2**,**fout\_3**,**fout\_po**)**d0**(**r\_mux**[**i**],**qbar**[**i**],**CLK**,**d\_in\_FF**[**i**],**RST**);**

//Netlist

**assign** R**[**i**]=**r\_mux**[**i**];**

**end**

**endmodule**

*Module 3.5*— tb\_register.v

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* \*\*\*

\*\*\* EE 526 L Experiment #3 Kyle E. Keislar, Spring, 2020 \*\*\*

\*\*\* \*\*\*

\*\*\* Hierarchical Modeling \*\*\*

\*\*\* \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: tb\_SR\_Latch.v \*\*\*

\*\*\* Author: Kyle E. Keislar \*\*\*

\*\*\* Date: 02/19/2020 \*\*\*

\*\*\* Version: 1.0 \*\*\*

\*\*\* Revised: \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Test Modules: register.sv \*\*\*

\*\*\* & Hierarchy : --MUX2\_1.v \*\*\*

\*\*\* --dff.v \*\*\*

\*\*\* ----SR\_Latch2.v (ver 1.1) \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Test Strategy: To test the register module, a heuristic approach is \*\*\*

\*\*\*adopted. The register operation for each of the three inputs is \*\*\*

\*\*\*observed for all meaningful cases. This includes when the register is\*\*\*

\*\*\*loaded with a value, when it is reset, and when it is disabled. To \*\*\*

\*\*\*observe register operation while disabled, data and reset are altered\*\*\*

\*\*\*while the enable input is low. To observe the behavior of the \*\*\*

\*\*\*register when a value is loaded, the register is enabled and the \*\*\*

\*\*\*reset is disabled (set to 1) while the data is toggled to different \*\*\*

\*\*\*values. Finally to observe the behavior of the register when it is \*\*\*

\*\*\*reset, the register is enabled and the reset input is enabled \*\*\*

\*\*\*(set to 0) while the data input is changed. \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Timescale

`timescale 1ns **/** 10ps

//Macro Definitions

`define period 25**;** //clock period

`define pulse 12.5**;** //clock pulse halv of period

`define runtime 800**;** //simulation run time for 50 periods

// Module Instantiation

**module** tb\_register**();**

//Intrinsic Delay definitions

**parameter** time\_delay\_1**=**1**;** //intrinsic 1 input

**parameter** time\_delay\_2**=**2**;** //intrinsic 2 input

**parameter** time\_delay\_3**=**3**;** //intrinsic 3 input

//Capacitive loading definitions

**parameter** fout\_1**=**0.4**;** //fanout of 1

**parameter** fout\_2**=**0.7**;** //fanout of 2

**parameter** fout\_3**=**1.0**;** //fanout of 3

**parameter** fout\_po**=**5**;** //primary output delay

//reg inputs

**reg** **[**7**:**0**]** data**;**

**reg** clock**,** enable**,** reset**;**

//wire outputs

**wire[**7**:**0**]** R**;**

register **#(**time\_delay\_1**,**time\_delay\_2**,**time\_delay\_3**,**fout\_1**,**fout\_2**,**fout\_3**,**fout\_po**)**UUT**(**R**,**clock**,**data**,**enable**,**reset**);**

//Initial Conditions

**initial**

clock**=**1'b0**;**//set clock to 0 at simulation start

**initial** **begin**

$vcdpluson**;** //include waveforms in simulation

//Initilization

data**=**8'hFF**;** enable**=**1'b1**;** reset**=**1'b0**;** //TV #1

**#**`period

**#**`period $display**(**"\n-------------------------------------|| Initial Conditions ||-------------------------------------\n"**);**

//$display(/\*"%d\tdata in = %h\tdata out = %h\tEN = %0d\trst = %0d"\*/,$time,data,R,enable,reset);

$display**(**"\n-------------------------------------|| Initial Conditions ||-------------------------------------\n"**);**

//a. The register resets when RST is zero.

$display**(**"a reset==0\n"**);**reset**=**1'b0**;** //TV #2

**#**`period

**#**`period

//change values to observe reset behavior

data**=**8'h77**;** //TV #3

**#**`period

**#**`period

data**=**8'h00**;** //TV #4

**#**`period

**#**`period

data**=**8'h11**;** //TV #5

//b. The contents of the data bus are clocked into the register when ENA is asserted.

$display**(**"b enable==1; data clocked in\n"**);**data**=**8'h22**;**reset**=**1'b1**;**//TV #6

**#**`period

**#**`period

//change values to observe register behavior

data**=**8'hFF**;** //TV #7

**#**`period

**#**`period

data**=**8'hAB**;** //TV #8

**#**`period

**#**`period

data**=**8'h33**;** //TV #9

**#**`period

**#**`period

**#**`period

//c. The contents of the register are preserved and the contents of the data bus

//are ignored when the register is clocked with ENA de-asserted.

$display**(**"c enable==0; data held\n"**);**data**=**8'hAA**;** enable**=**1'b0**;** //TV #10

**#**`period

**#**`period

//change values to observe reset behavior

data**=**8'h10**;** //TV #11

**#**`period

**#**`period

data**=**8'h00**;** //TV #12

**#**`period

**#**`period

data**=**8'hCC**;** //TV #13

**#**`period

**#**`period

$display**(**"reset to 0\n"**);**reset**=**1'b0**;** //reset to 0 //TV #14

**end**

//Synchronous Behavior

**always**

**begin**

**#**`pulse clock**=** 1'b1**;**

**#**`pulse clock**=**1'b0**;**

**end**

//Print Block (Prints every clock)

**always** **@** **(posedge** clock**)**

**begin**

$display**(**"%d\t %h\t %h\t%0d\t%0d"**,**$time**,**data**,**R**,**enable**,**reset**);**

**end**

//Check starting values & Finish the simulation at runtime

**always**

**begin**

**#**`runtime

$finish**;**

**end**

**endmodule**

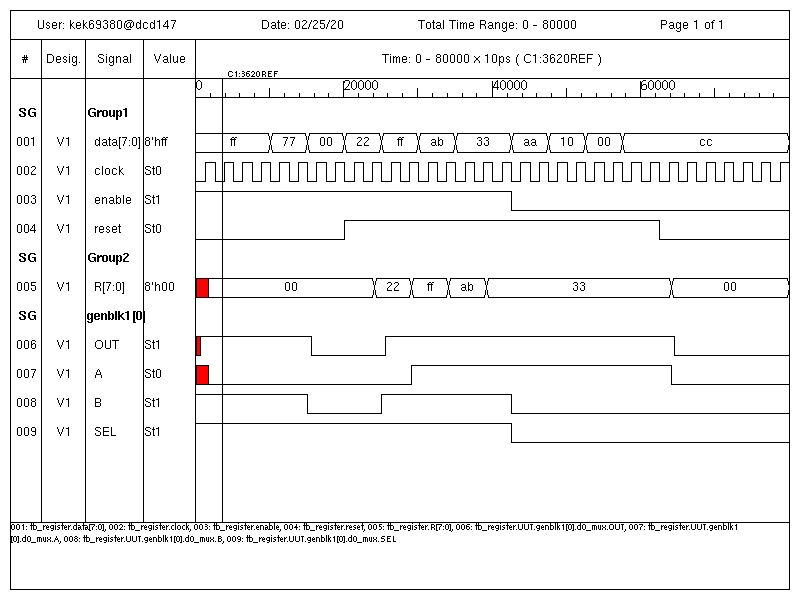


Figure 3.6 Simulation Behavioral Waveforms for Module 3.5 @ Default Frequency; f=40 MHz

*Table 3.6* Tabulated Experimental Output for the D\_FF register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Simulation Time** | **Test Vector #** | **r[7:0]** | **data[7:0]** | **ena** | **rst** |
| 63 | 1 | 0 | FF | 1 | 0 |
| 88 | 2 | 0 | FF | 1 | 0 |
| 113 | 3 | 0 | 77 | 1 | 0 |
| 138 | 4 | 0 | 77 | 1 | 0 |
| 163 | 5 | 0 | 0 | 1 | 0 |
| 188 | 6 | 0 | 0 | 1 | 0 |
| 213 | 7 | 0 | 22 | 1 | 1 |
| 238 | 8 | 0 | 22 | 1 | 1 |
| 263 | 9 | 22 | FF | 1 | 1 |
| 288 | 10 | 22 | FF | 1 | 1 |
| 313 | 11 | FF | AB | 1 | 1 |
| 338 | 12 | FF | AB | 1 | 1 |
| 363 | 13 | AB | 33 | 1 | 1 |
| 388 | 14 | AB | 33 | 1 | 1 |
| 413 | 15 | 33 | 33 | 1 | 1 |
| 438 | 16 | 33 | AA | 0 | 1 |
| 463 | 17 | 33 | AA | 0 | 1 |
| 488 | 18 | 33 | 10 | 0 | 1 |
| 513 | 19 | 33 | 10 | 0 | 1 |
| 538 | 20 | 33 | 0 | 0 | 1 |
| 563 | 21 | 33 | 0 | 0 | 1 |
| 588 | 22 | 33 | CC | 0 | 1 |
| 638 | 23 | 33 | CC | 0 | 0 |
| 663 | 24 | 0 | FF | 0 | 0 |

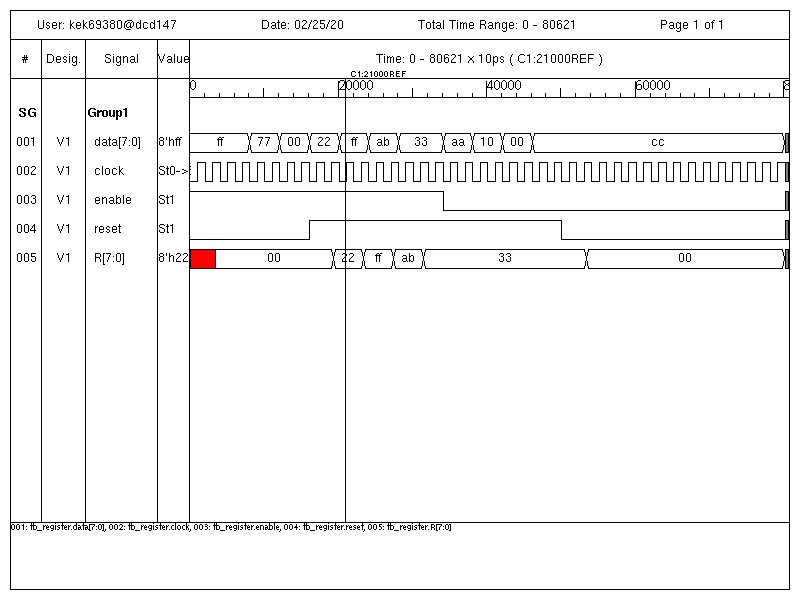
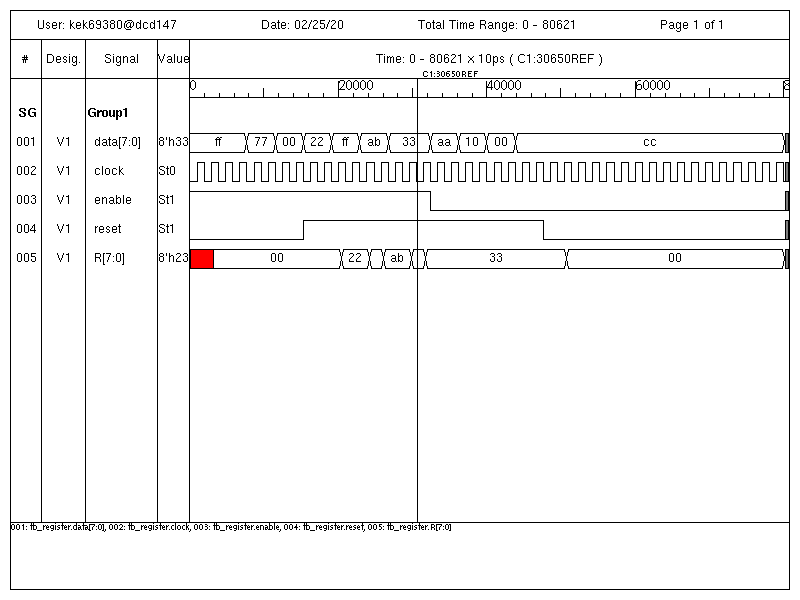


Figure 3.7 Simulation Behavioral Waveforms for Module 3.5 @ Operable Frequency f=50MHz

Figure 3.8 Simulation Behavioral Waveforms for Module 3.5 @ Inoperable Frequency f=52.6 MHz

1. **ANALYSIS**

The register constructed in this experiment functioned as intended, allowing for data to be loaded, stored as well as conforming with the expected functions of reset and clear. However, there were significant discrepancies in the perceived and theoretical operating frequencies accounted for during simulation.

The enable signal of the register as described above functioned as intended. When enable is low in test vectors 16-23, the register retains the last stored value of 33. However when enable is high, the store value can be changed by loading in new data as shown by test vectors 7-15.

The reset signal also performs as described and the interaction between this signal and the enable signal is examined. When reset is low the active low reset signal is asserted on the register. This clears the stored value to 0 as shown by test vectors 1-6, 23 and 24. When the reset signal is high, it is de-asserted and the register values are not reset, this can be observed for all other test vector cases. When enable is low and reset is low, the circuit is still reset. This behavior allows for the register to be cleared, or preset to 0, before use.

The circuit behaves as a sequential, synchronous register with an asynchronous clear. The data bits passed into the register are only propagated to the output on the rising edge of the clock signal.

During operation, the constructed SR latch flip-flops also behave as D flip flipflops. When enabled and not cleared, the flip-flop is set to the value of the input data bit. This is illustrated in test vectors 7-15. The register values only change under loading conditions (enable=1 & reset=1) when the data values also change.

In addition the simple multiplexing logic also acts to preserve values when the register is disabled. This can be seen in test vectors 16-23 where all register bits are retained through the iterative feedback logic when enable is low. When enable is high, the data values are passed.

During simulation, severe discrepancies between the theoretical and experimental operating frequencies were recorded. To find the maximum simulation frequency, the output of the register was analyzed and two faults were found at approximately 250 & 380 ns. In the case of the latter, the output of the register at this time was an erroneous value of 0x23 compared to the correct value of 0xAB. This value is reflected in test vector 14. The erroneous value does not conform to the expected operating conditions under this test vector. Both enable and reset are high, allowing the register to assume new values input on the data bus but the data flip-flops are not provided the clock pulse width to reset their values properly. The result is the input data value of 0xAB is not displayed as in the previous test vector.

At the point of these faults the clock period values were recorded to calculate a maximum operating frequency. This frequency was discovered between a period of 20 and 19 ns. To approximate maximum simulated frequency , these frequencies were averaged together as described by equation 3.4(1) below. This frequency is recorded in item 3.1.

(1)

This frequency is significantly higher than the theoretical frequency observed in Eq. 3.1(4). The simulated frequency possesses multiple sources of error. While the simulated and theoretical frequencies both accounted for the same delays and relative timescales, the simulated frequencies the nature of the chosen test vectors must be considered. This experiment’s simulation software assumed ideal sources of power during circuit operation which may not be reflective of the worst case.

1. **CONCLUSION**

From this experiment multiple conclusions about the hierarchical modeling of a register can be draw. It was shown that designs can be constituted of instantiations of component modules in a top-up methodology. The constructed register possessed 3 levels of design going from the lowest SR latch to the D flip-flop and multiplexer, to the top level of the register. The hierarchical approach required debugging and designs to begin at the lowest level and proceed up to the top level of the design. The approach was both tedious and through, while a behavioral approach was taken during the conceptual stage of the register’s design, each component was described at the gate level and was not abstracted with behavioral logic.

While the register functioned as intended, the notable difference form the expected operating frequency does not support the validity of the tested register. To correct this difference, the experiment should include a more through analysis of the critical path.

The register circuit described in this experiment possesses multiple limitations which must be taken in to account. The circuit’s sequential logic does not permit any any asynchronous setting of the flip-flops and no preset functionality was included in this experiment. In addition, while not reflected with confidence from these results, a maximum operating frequency for this sequential circuit must be considered in its applications. The circuit did not simulate with successful outputs given a constant test vector population at frequencies above ~51 MHz. Finally the circuit is also limited by multiple real world factors not considered in this simulation including: supplied power, operating temperature, undefined inputs, and user operation.

**APPENDIX**

1. **SIMULATION LOG A: Register under Default Period**

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version N-2017.12-SP2-2\_Full64; Runtime version N-2017.12-SP2-2\_Full64; Feb 25 11:16 2020

VCD+ Writer N-2017.12-SP2-2\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

13 data in = ff data out = xx EN = 1 rst = 0

38 data in = ff data out = 00 EN = 1 rst = 0

-------------------------------------|| Initial Conditions ||-------------------------------------

50 data in = ff data out = 00 EN = 1 rst = 0

-------------------------------------|| Initial Conditions ||-------------------------------------

a reset==0

63 data in = ff data out = 00 EN = 1 rst = 0

88 data in = ff data out = 00 EN = 1 rst = 0

113 data in = 77 data out = 00 EN = 1 rst = 0

138 data in = 77 data out = 00 EN = 1 rst = 0

163 data in = 00 data out = 00 EN = 1 rst = 0

188 data in = 00 data out = 00 EN = 1 rst = 0

b enable==1; data clocked in

213 data in = 22 data out = 00 EN = 1 rst = 1

238 data in = 22 data out = 00 EN = 1 rst = 1

263 data in = ff data out = 22 EN = 1 rst = 1

288 data in = ff data out = 22 EN = 1 rst = 1

313 data in = ab data out = ff EN = 1 rst = 1

338 data in = ab data out = ff EN = 1 rst = 1

363 data in = 33 data out = ab EN = 1 rst = 1

388 data in = 33 data out = ab EN = 1 rst = 1

413 data in = 33 data out = 33 EN = 1 rst = 1

c enable==0; data held

438 data in = aa data out = 33 EN = 0 rst = 1

463 data in = aa data out = 33 EN = 0 rst = 1

488 data in = 10 data out = 33 EN = 0 rst = 1

513 data in = 10 data out = 33 EN = 0 rst = 1

538 data in = 00 data out = 33 EN = 0 rst = 1

563 data in = 00 data out = 33 EN = 0 rst = 1

588 data in = cc data out = 33 EN = 0 rst = 1

613 data in = cc data out = 33 EN = 0 rst = 1

reset to 0

638 data in = cc data out = 33 EN = 0 rst = 0

663 data in = cc data out = 00 EN = 0 rst = 0

688 data in = cc data out = 00 EN = 0 rst = 0

713 data in = cc data out = 00 EN = 0 rst = 0

738 data in = cc data out = 00 EN = 0 rst = 0

763 data in = cc data out = 00 EN = 0 rst = 0

788 data in = cc data out = 00 EN = 0 rst = 0

$finish called from file "tb\_register.v", line 146.

$finish at simulation time 80000

V C S S i m u l a t i o n R e p o r t

Time: 800000 ps

CPU Time: 0.320 seconds; Data structure size: 0.0Mb

Tue Feb 25 11:16:54 2020

1. **Experiment 3 Force file for VCS**

*Lab3\_cmds.f*

cd ~/526/lab3

vcs -debug -sverilog -full64 tb\_register.v register.sv MUX2\_1.v dff.sv SR\_Latch2.v

simv | tee lab3\_log.log