**PROBLEM STATEMENT**

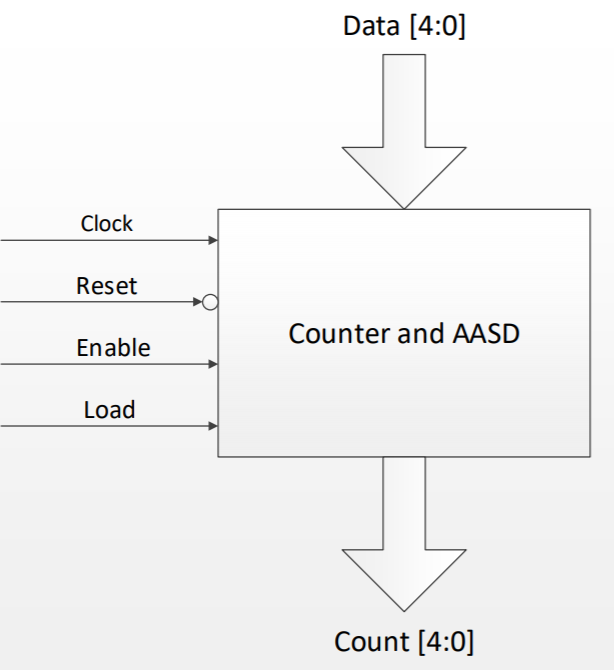
This experiment explores the behavioral modeling of circuits using Verilog through the construction of a simple five bit binary up counter. A simple counter is constructed with multiple functions via behavioral descriptions, as opposed to gate level descriptions, of the modules. The counter is simulated to observe its behavior.

**KEYWORDS:** Binary Up Counter, AASD, Multiplexer, Data Bus, Binary Adder, D flip-flop Set up time, hold time, release time.

1. **INTRODUCTION**

A binary up counter is a digital circuit which continuously increments its values to count from 0 to 2n-1 where n is the number of bits. In this experiment, the five bit binary up counter will count from 0 to 31 and then reset to 0. A binary up counter uses multiple modules to achieve this function including a sequential memory module in the form of a D flip-flop and a data bus multiplexer. The high level block for the 5-bit is shown below in Figure 4.1

Figure 4.1 5-bit Up Counter High Level Block



The additional inputs of enable, reset and load provide additional functions to the binary counter. The enable line can be de-asserted to retain the current value of the counter or enabled to allow the counter value to be changed. The reset is an active low input which resets the value of the counter to 0 when asserted. Finally the load line possesses two functions: when it is 0, the counter will act as a binary up counter and count from the current value up towards 31. If the load line is high, then the counter will assume the binary value passed in on the data lines. Table 4.1 below summarizes the functionality of the binary up counter.

Table 4.1 5-bit Binary Up Counter Truth Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data | L | EN | R | CLK | Count(*i*) |
|  | X | 0 | X | X | Count(*i-1*) |
|  | 0 | 1 | 1 | ↑ | Data+1 |
|  | 1 | 1 | 1 | ↑ |  |
|  | X | 1 | 0 |  | 0 |
| ↑ = Positive Edge  X = Don’t Care | |  | |  | |
| L=Load  R=reset  Count(*i*)=i-th clock of the counter | |  | | EN=enable  CLK=clock | |

The schematic for a single binary up counter bit is shown as an iterative module below in Figure 4.2

Figure 4.2 5-bit Binary Up Counter Scheamtic



While no explicit delays are modeled in this counter’s behavioral expression, the simulation must takes into account the respective parameters of the counter’s flip-flops. Specifically these include set up time and release time. If a value changes while the clock transitioning, unpredictable outputs can be observed. To circumvent these undesirable transitions, a reset signal is used such that the counter is reset and set without violating the intrinsic D-FF hold and set-up times. To do this the reset signal remains asynchronous in its assertion but can only be de-asserted on the rising edge of the clock pulse following its assignment. An active low assertion is modeled. This reset module is called the AASD or Asynchronous Assert Synchronous Deassert module and is described by table 4.2 below

Table 4.2 AASD Truth Table

|  |  |  |  |
| --- | --- | --- | --- |
| Reset | Clock | AASD\_R- | AASD\_R |
| 0 | X | X | 0 |
| 1 | ↑ | 0 | 1 |
| 1 |  | 0 |  |

The schematic for this module is shown below in Figure 4.2, To minimize the risk of a release violation, two D\_FFs are used to provide a signal redundancy and delay.

Figure 4.3 AASD Schematic



1. **METHODOLOGY**

To test the five bit up binary counter a significant data burden is placed on the simulation. Each of the 32 states of the counter for values 0-32 must be observed. In addition the synchronous functions of loading, incrementing and holding must be tested under all asynchronous inputs. The asynchronous conditions of resetting the counter, disabling the counter and synchronous de-assertion of reset must also be tested.

To achieve each of these operating states, the values of the enable, load and reset signals in synchronicity with the clock generator module. the data input is only utilized by the counter in the load mode (case 2) but is persistently asserted to illustrate counter function in the other cases In addition, the enable and reset inputs are tested to demonstrate counter behavior when enabled and disabled independently of data values.

To meet this data burden multiple modules are constructed. A bottom level behavioral description of the five bit counter is modeled using behavioral Verilog constructs as a basic up counter when the load line is low. This module is described as a function in table 4.1 and is shown below in Module 4.1. In addition to the counter module an AASD module is also constructed to direct the proper assertion/de-assertion of the reset signal. This module is described by Table 4.2 and Figure 4.3 and is shown below as Module 4.2. To instantiate both of these modules on the same hierarchy, a single top level of the design is also utilized as shown below in Module 4.3. To test these modules, a testbench module was constructed containing a clock generator module and testbench code. The testbench module can be seen below as Module 4.4

The test vectors for this simulation cover a variety of operating states. Each of the 32 states must be traversed by the counter as well as data with load, data without load, reset with enable and load, reset without enable and load, and reset with load. In addition the counter output must be observed when load changes from 1 to 0 after data is loaded.

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. Figure 4.4 shows the captured behavioral waveforms of all recorded experimental values during simulation. This output is tabulated in Table 4.3

1. **MODULE FILES & SIMULATION RESULTS**

*Module 4.1*—Fivebcount.v

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

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

\*\*\* EE 526 L Experiment #4 Kyle E. Keislar, Spring, 2020 \*\*\*

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

\*\*\* Behavioral Modeling of a Counter \*\*\*

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

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

\*\*\* Filename: Fivebcount.v \*\*\*

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

\*\*\* Date: 02/20/2020 \*\*\*

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

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

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

\*\*\* Module Description: 5-bit binary up counter with enable, reset and \*\*\*

\*\*\* load capabilities. \*\*\*

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

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

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

`timescale 1ns **/** 10ps

// Module Declaration

**module** Fivebcount**(** COUNT**,** //Counter output; 5 bit bus

DATA**,** //counter input w/ feedback

CLK**,** //clock input

RST**,** //AASD signal; when low && enabled resets to 0

EN**,** //active high enable; when low COUNT holds

L**);** //active high load; when low && enabled COUNT increments t

//Module Parameters

// I/O port assignment

**output** **reg** **[**4**:**0**]** COUNT**;**

**input** **wire** **[**4**:**0**]** DATA**;**

**input** **wire** CLK**,** RST**,** EN**,** L**;**

//Internal Signals

**reg** **[**4**:**0**]** count\_reg**;**//wire from enable logic to output registers,

//Load logic

**always** **@(**DATA **or** **posedge** L **or** **posedge** EN **or** **negedge** CLK**)**

**begin**

**if(**RST**==**1'b1 **&&** EN**==**1'b1**)begin**

**if(**L**==**1'b0**)** //increment when load is low

count\_reg**=**count\_reg**+**1**;**

**else**

count\_reg**=**DATA**;** //load data when load is asserted

**end**

**end**

//Reset logic

**always** **@** **(negedge** RST**)**//when active low Reset

**begin**

**if(**EN**==**1'b1**)begin**

COUNT**=**0**;**

count\_reg**=**0**;**

**end**

**end**

//enable logic

**always** **@** **(negedge** EN**)**

count\_reg**=**count\_reg**;**//hold the count

//clock count

**always** **@** **(posedge** CLK**)**

**begin**

COUNT**=**count\_reg**;**

**end**

**endmodule**

*Module 4.2*—AASD.v

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

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

\*\*\* EE 526 L Experiment #4 Kyle E. Keislar, Spring, 2020 \*\*\*

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

\*\*\* Behavioral Modeling of a Counter \*\*\*

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

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

\*\*\* Filename: AASD.v \*\*\*

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

\*\*\* Date: 02/20/2020 \*\*\*

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

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

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

\*\*\* Module Description: Asynchronous reset, Synchronous Deassert \*\*\*

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

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

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

`timescale 1ns **/** 10ps

// Module Declaration

**module** AASD**(** AASD\_R**,** //Output reset signal, low when RST is low

RST**,** //Counter output; 5 bit bus

CLK**);** //clock input

//Module Parameters

// I/O port assignment

**output** **reg** AASD\_R**;**

**input** **wire** RST**,** CLK**;**

//Internal Signals

**wire** dr0**,**dr1**;**

//Asynchronous Reset

**always** **@** **(**RST**)begin**

**if(**RST**==**1'b0**)** AASD\_R**=**1'b0**;**

**end**

//Synchronous Deassert

**always** **@** **(posedge** CLK**)** **begin**

**if(** RST**==**1'b1**)** **begin** //if not being reset reassert on next clock

AASD\_R **=** 1'b1**;**

**end**

**end**

**endmodule**

*Module 4.3*—top\_Fivebcount.v

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

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

\*\*\* EE 526 L Experiment #4 Kyle E. Keislar, Spring, 2020 \*\*\*

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

\*\*\* Behavioral Modeling of a Counter \*\*\*

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

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

\*\*\* Filename: top\_Fivebcount.v \*\*\*

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

\*\*\* Date: 02/20/2020 \*\*\*

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

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

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

\*\*\* Module Description: Top design level for 5-bit counter with AASD \*\*\*

\*\*\* module \*\*\*

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

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

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

`timescale 1ns **/** 10ps

// Module Declaration

**module** top\_Fivebcount**(** COUNT**,** //Counter output; 5 bit bus

DATA**,** //counter input w/ feedback

CLK**,** //clock input

RST**,** //AASD signal; when low && enabled resets to0

EN**,** //active high enable; when low COUNT holds

L**);** //active high load; when low && enabled COUNT increments t

//Module Parameters

// I/O port assignment

**output** **wire** **[**4**:**0**]** COUNT**;**

**input** **wire** **[**4**:**0**]** DATA**;**

**input** **wire** CLK**,** RST**,** EN**,** L**;**

//Internal Signals

**wire** aasd\_r**;**

//Top Level Instnatiations

AASD aasd**(**aasd\_r**,**RST**,**CLK**);**

Fivebcount FBC**(**COUNT**,**DATA**,**CLK**,**aasd\_r**,**EN**,**L**);**

**Endmodule**

*Module 4.4*—tb\_top\_Fivebcount.v

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

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

\*\*\* EE 526 L Experiment #4 Kyle E. Keislar, Spring, 2020 \*\*\*

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

\*\*\* Experiment #4: \*\*\*

\*\*\* Behavioral Modeling of a Counter \*\*\*

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

\*\*\* Filename: tb\_top\_5bcount.v \*\*\*

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

\*\*\* Date: 02/20/2020 \*\*\*

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

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

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

\*\*\* Test Modules: top\_5bcount.v aasd.v Fivebcount.v \*\*\*

\*\*\* & Hierarchy : \*\*\*

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

\*\*\* Test Strategy: The behavior of the counter is observed under each of\*\*\*

its four operating states: \*\*\*

1) Enabled, Set and Incrementing. \*\*\*

2) Enabled, Set and Loading. \*\*\*

3) Enabled and Reset \*\*\*

4) Disabled \*\*\*

To achieve each of these operating states, the \*\*\*

values of the enable, load and reset signals in \*\*\*

synchronicity with the clock generator module. \*\*\*

the data input is only utilized by the counter in the\*\*\*

load mode (case 2) but is persistently asserted to \*\*\*

illustrate counter function in the other cases \*\*\*

In addition, the enable and reset inputs are tested \*\*\*

to demonstrate counter behavior when enabled and \*\*\*

disabled independly of data values. \*\*\*

\*\*\*

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

//Timescale

`timescale 1ns **/** 10ps

`define period 10**;** //clock period

//Macro Definitions

`define pulse 5**;** //clock pulse halv of period

`define runtime 1000**;** //simulation run time for 100 periods

// Module Instantiation

**module** tb\_top\_Fivebcount**();**

//reg inputs

**reg** **[**4**:**0**]** data**;**

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

//wire outputs

**wire** **[**4**:**0**]** count**;**

top\_Fivebcount UUT**(**count**,**data**,**clock**,**reset**,**enable**,**load**);**

//Initial Conditions

**initial**

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

**initial** **begin**

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

//Initilization (reset to 0)

enable**=**1'b1**;** data**=**16**;**

**#**`period

**#**`period

**#**`period

**#**`period

**#**`period

load**=**1'b0**;**

**#**`period //AASD\_R de asserted //On De-assert of reset, begin Incrementing to 5

**#**`period //AASD\_R propagation delay

**#**`period //Increment to 1

**#**`period //2

**#**`period //3

**#**`period //4

load**=**1'd1**;** data**=**28**;** //Load 28 when @ 5

**#**`period //5

load**=**1'b0**;**

**#**`period //load delay

**#**`period //28

**#**`period //29

**#**`period //30

**#**`period //31

**#**`period //0

**#**`period //1

**#**`period //2

**#**`period //3

**#**`period //4

reset**=**1'b0**;** //Increment to 4 Set rst to 0

**#**`period

load**=**1'b1**;** data**=**31**;**

//show 0 for 4 periods w/ reset @ 0 and Load toggling from 0-->1

**#**`period

load**=**1'b0**;** data**=**1**;**

**#**`period

load**=**1'b1**;** data**=**12**;**

//show 0 for 4 periods w/ reset @ 0 and Load toggling from 0-->1

**#**`period

load**=**1'b0**;** data**=**1**;**

**#**`period //0

reset**=**1'b1**;** data**=**1'b0**;**

**#**`period //1

**#**`period //2

**#**`period //3

**#**`period //4

**#**`period //5

load**=**1'b1**;** data**=**13**;**//Set En to 0

**#**`period //6

enable**=**1'b0**;** load**=**1'b0**;**

**#**`period //show 6 for 4 periods w/ reset,load @{00,01,10,11}

reset**=**1'b0**;**

**#**`period

**#**`period

reset**=**1'b0**;** load**=**1'b1**;** data**=**11**;**

**#**`period

**#**`period

reset**=**1'b1**;** load**=**1'b0**;** data**=**11**;**

**#**`period

**#**`period

reset**=**1'b1**;** load**=**1'b1**;** data**=**11**;**

**#**`period

**#**`period

enable**=**1'b1**;** load**=**1'b1**;** data**=**20**;** //Set Rst to 1, load to 0 and en to 1

**#**`period

load**=**1'b0**;** //increment from 20 10; //0

**end**

//Asynchronous Behavior

**always**

**begin**

**#**31.9 reset**=**1'b0**;** //Asynchronous Reset

**#**0 $display**(**"RESET @ %d"**,**$time**);**

**#**21 reset**=**1'b1**;** //Synchronous Deassert

**#**1000**;**

**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, Cur\_count= %d\tData\_in=%d\ten=%b\trst=%b\tload=%b"**,** $time**,**count**,**data**,**enable**,**reset**,**load**);**

**end**

//Check starting values & Finish the simulation at runtime

**always**

**begin**

**#**`runtime

$finish**;**

**end**

**endmodule**

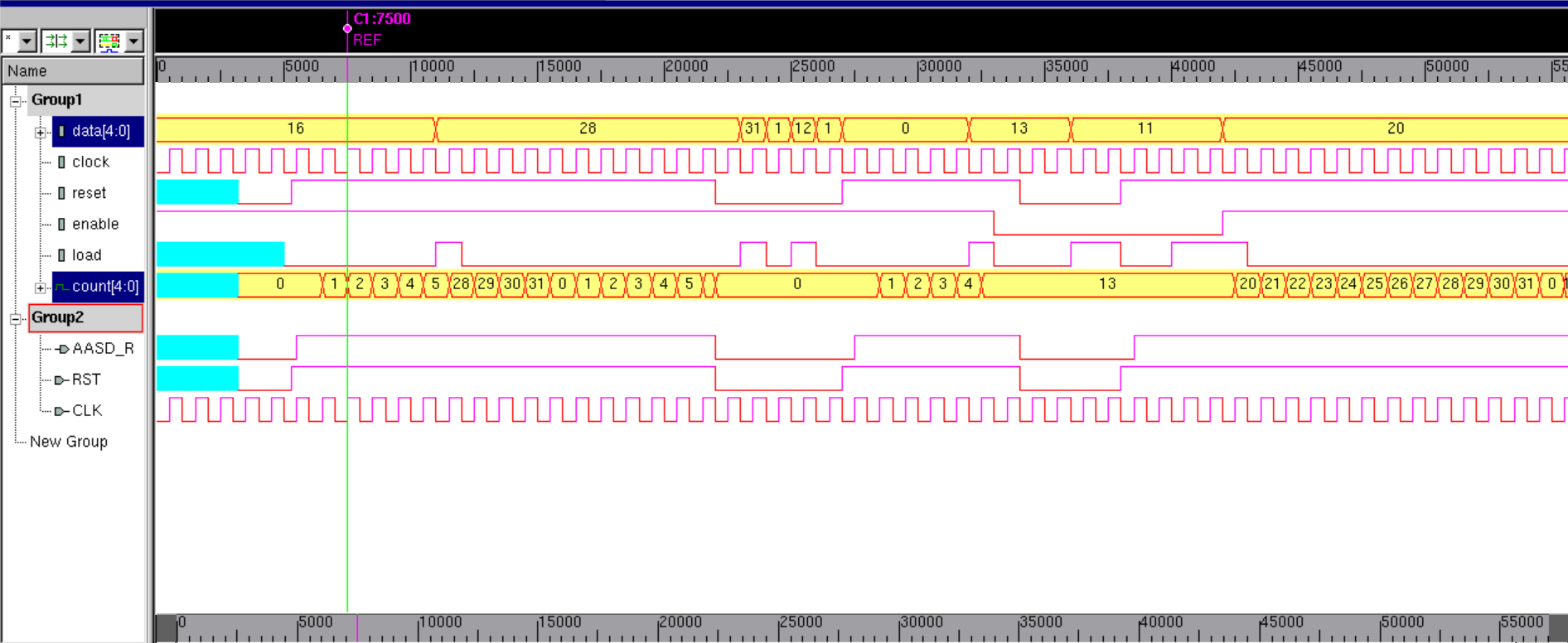


Figure 4.4 Simulation Behavioral Waveforms for Module 4.4

*Table 4.3* Tabulated Experimental Output for the 5-bit Up Counter

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Simulation Time** | **Test Vector #** | **Count** | **Data in** | **enable** | **reset** | **load** |
| 5 | 1 | X | 16 | 1 | X | X |
| 15 | 2 | X | 16 | 1 | X | X |
| 25 | 3 | X | 16 | 1 | X | X |
| 35 | 4 | 0 | 16 | 1 | 0 | X |
| 45 | 5 | 0 | 16 | 1 | 0 | X |
| 55 | 6 | 0 | 16 | 1 | 1 | 0 |
| 65 | 7 | 0 | 16 | 1 | 1 | 0 |
| 75 | 8 | 1 | 16 | 1 | 1 | 0 |
| 85 | 9 | 2 | 16 | 1 | 1 | 0 |
| 95 | 10 | 3 | 16 | 1 | 1 | 0 |
| 105 | 11 | 4 | 16 | 1 | 1 | 0 |
| 115 | 12 | 5 | 28 | 1 | 1 | 1 |
| 125 | 13 | 28 | 28 | 1 | 1 | 0 |
| 135 | 14 | 29 | 28 | 1 | 1 | 0 |
| 145 | 15 | 30 | 28 | 1 | 1 | 0 |
| 155 | 16 | 31 | 28 | 1 | 1 | 0 |
| 165 | 17 | 0 | 28 | 1 | 1 | 0 |
| 175 | 18 | 1 | 28 | 1 | 1 | 0 |
| 185 | 19 | 2 | 28 | 1 | 1 | 0 |
| 195 | 20 | 3 | 28 | 1 | 1 | 0 |
| 205 | 21 | 4 | 28 | 1 | 1 | 0 |
| 215 | 22 | 5 | 28 | 1 | 1 | 0 |
| 225 | 23 | 0 | 28 | 1 | 0 | 0 |
| 235 | 24 | 0 | 31 | 1 | 0 | 1 |
| 245 | 25 | 0 | 1 | 1 | 0 | 0 |
| 255 | 26 | 0 | 12 | 1 | 0 | 1 |
| 265 | 27 | 0 | 1 | 1 | 0 | 0 |
| 275 | 28 | 0 | 0 | 1 | 1 | 0 |
| 285 | 29 | 0 | 0 | 1 | 1 | 0 |
| 295 | 30 | 1 | 0 | 1 | 1 | 0 |
| 305 | 31 | 2 | 0 | 1 | 1 | 0 |
| 315 | 32 | 3 | 0 | 1 | 1 | 0 |
| 325 | 33 | 4 | 13 | 1 | 1 | 1 |
| 335 | 34 | 13 | 13 | 0 | 1 | 0 |
| 345 | 35 | 13 | 13 | 0 | 0 | 0 |
| 355 | 36 | 13 | 13 | 0 | 0 | 0 |
| 365 | 37 | 13 | 11 | 0 | 0 | 1 |
| 375 | 38 | 13 | 11 | 0 | 0 | 1 |
| 385 | 39 | 13 | 11 | 0 | 1 | 0 |
| 395 | 40 | 13 | 11 | 0 | 1 | 0 |
| 405 | 41 | 13 | 11 | 0 | 1 | 1 |
| 415 | 42 | 13 | 11 | 0 | 1 | 1 |
| 425 | 43 | 13 | 20 | 1 | 1 | 1 |
| 435 | 44 | 20 | 20 | 1 | 1 | 0 |
| 445 | 45 | 21 | 20 | 1 | 1 | 0 |
| 455 | 46 | 22 | 20 | 1 | 1 | 0 |
| 465 | 47 | 23 | 20 | 1 | 1 | 0 |
| 475 | 48 | 24 | 20 | 1 | 1 | 0 |
| 485 | 49 | 25 | 20 | 1 | 1 | 0 |
| 495 | 50 | 26 | 20 | 1 | 1 | 0 |
| 505 | 51 | 27 | 20 | 1 | 1 | 0 |
| 515 | 52 | 28 | 20 | 1 | 1 | 0 |
| 525 | 53 | 29 | 20 | 1 | 1 | 0 |
| 535 | 54 | 30 | 20 | 1 | 1 | 0 |
| 545 | 55 | 31 | 20 | 1 | 1 | 0 |

1. **ANALYSIS**

The five-bit binary up counter functioned as intended and successfully demonstrated the behavior of enable/disable, reset, load, incrementation and all other tested forms of operation.

When the counter was enabled (enable=1), each of the other counting functions could be asserted or de-asserted. These functions can be shown to be dependent on enable by contrasting test vectors 1-33, 43-55 with 34-42. When enable is high, the counter is able to be reset, incremented or loaded. However when enable is low the counter’s last value is held until enable is high again.

The reset function of the counter was also demonstrated. This function is shown in operation in test vectors 4,5,23-27, & 35-38. In test vectors 4 and 5 the counter is reset to initialize it to 0 regardless of the value of load or the data bus. The counter was also shown to reset to 0 during the counting operation in vectors 23-27, and when enabled the counter did not begin incrementing until the following clock pulse. Finally the reset functionality when the counter was disabled is observed in vectors 35-38:while enable is low, the counter retains its value even as reset is changed.

The load functionality of the counter is demonstrated thoroughly under the states of the previous two conditions. In test vectors 1-5, load is not defined and does not cause the counter’s value to change until input conditions are suitable and the load signal it is defined in test vector 6. When load is low during the simulation, the counter is incremented if both reset and enable are high. If one or both are low, the counter is not incremented. When load, reset and enable are all high, the counter’s value is passed the value on the data line. This can be observed in test vectors 12 and 33. Again if either reset or enable is low, the current count is not dependent on load or the data line.

The waveforms in Figure 4.4 also highlight the functionality of the AASD\_r module described in Module 4.2. The reset signal can be shown to transition from High to Low in an asynchronous and immediate manner. However, when de-asserted the reset signal must propagate through the module, which requires two clock pulses. This can be illustrated in the simulation time from ~250-290ns. It should also be noted that the AASD signal is independent of the counter’s function or value and as such cannot be changed by the enable or load signal.

1. **CONCLUSION**

From this experiment multiple conclusions about the five bit up counter can be drawn. The counter allows for both preset and clear operations through the de-assertion of reset and assertion of load or the assertion of reset respectively. This operation allows for the counter to be synchronously adjusted to represent different values. In addition the counter value can be held by de-asserting enable.

While the counter did function as anticipated, there are various limitations that must be considered during its operation and application. This counter can only assume values between 0 and 31 and is not capable of representing a larger value with the current number of module iterations. In addition, the counter is a synchronous circuit and possesses a maximum operating frequency. The simulated values for this experiment reflect multiple idealized factors not considered in this simulation including: supplied power, operating temperature, undefined inputs, and user operation.

To mitigate some of these limitations, the counter can be made more scalable by altering the upper levels of its design to allow for larger values to be recorded. To do this the counter must generate sequential bit logic as defined by the user. To accomplish this Verilog parameters can be utilized in the top level of the design as described in module 4.3. By parameterizing the width of the data input and register output the data bus can be adjusted as followed. In addition, a loop can be utilized in this module to create a definitive number of flip-flops as dictated by the user. The counter would then only proceed to the defined value before resetting. A possible modification of this module is included in the appendix as item 4.7.

**APPENDIX**

1. **SIMULATION LOG A**

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 27 03:14 2020

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

5 , Cur\_count= x Data\_in=16 en=1 rst=x load=x

15 , Cur\_count= x Data\_in=16 en=1 rst=x load=x

25 , Cur\_count= x Data\_in=16 en=1 rst=x load=x

RESET @ 32

35 , Cur\_count= 0 Data\_in=16 en=1 rst=0 load=x

45 , Cur\_count= 0 Data\_in=16 en=1 rst=0 load=x

55 , Cur\_count= 0 Data\_in=16 en=1 rst=1 load=0

65 , Cur\_count= 0 Data\_in=16 en=1 rst=1 load=0

75 , Cur\_count= 1 Data\_in=16 en=1 rst=1 load=0

85 , Cur\_count= 2 Data\_in=16 en=1 rst=1 load=0

95 , Cur\_count= 3 Data\_in=16 en=1 rst=1 load=0

105 , Cur\_count= 4 Data\_in=16 en=1 rst=1 load=0

115 , Cur\_count= 5 Data\_in=28 en=1 rst=1 load=1

125 , Cur\_count= 28 Data\_in=28 en=1 rst=1 load=0

135 , Cur\_count= 29 Data\_in=28 en=1 rst=1 load=0

145 , Cur\_count= 30 Data\_in=28 en=1 rst=1 load=0

155 , Cur\_count= 31 Data\_in=28 en=1 rst=1 load=0

165 , Cur\_count= 0 Data\_in=28 en=1 rst=1 load=0

175 , Cur\_count= 1 Data\_in=28 en=1 rst=1 load=0

185 , Cur\_count= 2 Data\_in=28 en=1 rst=1 load=0

195 , Cur\_count= 3 Data\_in=28 en=1 rst=1 load=0

205 , Cur\_count= 4 Data\_in=28 en=1 rst=1 load=0

215 , Cur\_count= 5 Data\_in=28 en=1 rst=1 load=0

225 , Cur\_count= 0 Data\_in=28 en=1 rst=0 load=0

235 , Cur\_count= 0 Data\_in=31 en=1 rst=0 load=1

245 , Cur\_count= 0 Data\_in= 1 en=1 rst=0 load=0

255 , Cur\_count= 0 Data\_in=12 en=1 rst=0 load=1

265 , Cur\_count= 0 Data\_in= 1 en=1 rst=0 load=0

275 , Cur\_count= 0 Data\_in= 0 en=1 rst=1 load=0

285 , Cur\_count= 0 Data\_in= 0 en=1 rst=1 load=0

295 , Cur\_count= 1 Data\_in= 0 en=1 rst=1 load=0

305 , Cur\_count= 2 Data\_in= 0 en=1 rst=1 load=0

315 , Cur\_count= 3 Data\_in= 0 en=1 rst=1 load=0

325 , Cur\_count= 4 Data\_in=13 en=1 rst=1 load=1

335 , Cur\_count= 13 Data\_in=13 en=0 rst=1 load=0

345 , Cur\_count= 13 Data\_in=13 en=0 rst=0 load=0

355 , Cur\_count= 13 Data\_in=13 en=0 rst=0 load=0

365 , Cur\_count= 13 Data\_in=11 en=0 rst=0 load=1

375 , Cur\_count= 13 Data\_in=11 en=0 rst=0 load=1

385 , Cur\_count= 13 Data\_in=11 en=0 rst=1 load=0

395 , Cur\_count= 13 Data\_in=11 en=0 rst=1 load=0

405 , Cur\_count= 13 Data\_in=11 en=0 rst=1 load=1

415 , Cur\_count= 13 Data\_in=11 en=0 rst=1 load=1

425 , Cur\_count= 13 Data\_in=20 en=1 rst=1 load=1

435 , Cur\_count= 20 Data\_in=20 en=1 rst=1 load=0

445 , Cur\_count= 21 Data\_in=20 en=1 rst=1 load=0

455 , Cur\_count= 22 Data\_in=20 en=1 rst=1 load=0

465 , Cur\_count= 23 Data\_in=20 en=1 rst=1 load=0

475 , Cur\_count= 24 Data\_in=20 en=1 rst=1 load=0

485 , Cur\_count= 25 Data\_in=20 en=1 rst=1 load=0

495 , Cur\_count= 26 Data\_in=20 en=1 rst=1 load=0

505 , Cur\_count= 27 Data\_in=20 en=1 rst=1 load=0

515 , Cur\_count= 28 Data\_in=20 en=1 rst=1 load=0

525 , Cur\_count= 29 Data\_in=20 en=1 rst=1 load=0

535 , Cur\_count= 30 Data\_in=20 en=1 rst=1 load=0

545 , Cur\_count= 31 Data\_in=20 en=1 rst=1 load=0

555 , Cur\_count= 0 Data\_in=20 en=1 rst=1 load=0

565 , Cur\_count= 1 Data\_in=20 en=1 rst=1 load=0

$finish called from file "tb\_top\_Fivebcount.v", line 159.

$finish at simulation time 57000

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

Time: 570000 ps

CPU Time: 0.250 seconds; Data structure size: 0.0Mb

Thu Feb 27 03:14:57 2020

1. **Parameterized top\_Fivebcount Module**

top\_Fivebcount.v

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

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

\*\*\* EE 526 L Experiment #4 Kyle E. Keislar, Spring, 2020 \*\*\*

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

\*\*\* Behavioral Modeling of a Counter \*\*\*

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

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

\*\*\* Filename: top\_Fivebcount.v \*\*\*

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

\*\*\* Date: 02/20/2020 \*\*\*

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

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

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

\*\*\* Module Description: Top design level for 5-bit counter with AASD \*\*\*

\*\*\* module \*\*\*

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

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

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

`timescale 1ns **/** 10ps

// Module Declaration

**module** top\_Fivebcount**(** COUNT**,** //Counter output; 5 bit bus

DATA**,** //counter input w/ feedback

CLK**,** //clock input

RST**,** //AASD signal; when low && enabled resets to0

EN**,** //active high enable; when low COUNT holds

L**);** //active high load; when low && enabled COUNT increments t

//Module Parameters

Parameter WIDTH=16 //16 bit counter

// I/O port assignment

**output** **wire** **[**WIDTH-1**:**0**]** COUNT**;**

**input** **wire** **[**WIDTH-1**:**0**]** DATA**;**

**input** **wire** CLK**,** RST**,** EN**,** L**;**

//Internal Signals

**wire** aasd\_r**;**

//Top Level Instnatiations

AASD aasd**(**aasd\_r**,**RST**,**CLK**);**

Fivebcount #(WIDTH-1) FBC**(**COUNT**,**DATA**,**CLK**,**aasd\_r**,**EN**,**L**);**

**Endmodule**

Fivebcount.v

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

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

\*\*\* EE 526 L Experiment #4 Kyle E. Keislar, Spring, 2020 \*\*\*

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

\*\*\* Behavioral Modeling of a Counter \*\*\*

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

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

\*\*\* Filename: Fivebcount.v \*\*\*

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

\*\*\* Date: 02/20/2020 \*\*\*

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

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

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

\*\*\* Module Description: 5-bit binary up counter with enable, reset and \*\*\*

\*\*\* load capabilities. \*\*\*

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

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

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

`timescale 1ns **/** 10ps

// Module Declaration

**module** Fivebcount**(** COUNT**,** //Counter output; 5 bit bus

DATA**,** //counter input w/ feedback

CLK**,** //clock input

RST**,** //AASD signal; when low && enabled resets to 0

EN**,** //active high enable; when low COUNT holds

L**);** //active high load; when low && enabled COUNT increments t

//Module Parameters

Parameter WIDTH=16 //16 bit counter

// I/O port assignment

**output** **reg** **[**WIDTH-1**:**0**]** COUNT**;**

**input** **wire** **[**WIDTH-1**:**0**]** DATA**;**

**input** **wire** CLK**,** RST**,** EN**,** L**;**

//Internal Signals

**reg** **[**WIDTH-1**:**0**]** count\_reg**;**//wire from enable logic to output registers,

//Load logic

**always** **@(**DATA **or** **posedge** L **or** **posedge** EN **or** **negedge** CLK**)**

**begin**

**if(**RST**==**1'b1 **&&** EN**==**1'b1**)begin**

**if(**L**==**1'b0**)** //increment when load is low

count\_reg**=**count\_reg**+**1**;**

**else**

count\_reg**=**DATA**;** //load data when load is asserted

**end**

**end**

//Reset logic

**always** **@** **(negedge** RST**)**//when active low Reset

**begin**

**if(**EN**==**1'b1**)begin**

COUNT**=**0**;**

count\_reg**=**0**;**

**end**

**end**

//enable logic

**always** **@** **(negedge** EN**)**

count\_reg**=**count\_reg**;**//hold the count

//clock count

**always** **@** **(posedge** CLK**)**

**begin**

COUNT**=**count\_reg**;**

**end**

**endmodule**

1. **Experiment 4 Force file for VCS**

cd ~/526/lab4

vcs -debug -full64 tb\_top\_Fivebcount.v top\_Fivebcount.v AASD.v Fivebcount.v

simv | tee lab4\_log.log