Part IV

SEQUENTIAL CIRCUITS

SEQUENTIAL LOGIC circuits develop the concepts of clock-driven logic while creating several practical counters and memory circuits. These labs also introduce the *Logisim-Evolution Chronogram*, which builds timing diagrams for sequential logic circuits.

6 COUNTERS

Counters are perhaps the most commonly-used circuits in electronic devices. They are found in virtually all electronics systems, from the simplest embedded computers to massive mainframes. Counters are designed to cycle through a specific predefined sequence of binary numbers when an input pulse is applied. Typically, counters simply count up or down from given start and end numbers, but they can be designed to produce unique output patterns for special uses.

Counters, though, are used for more than simple counting. They can measure time so devices like alarm clocks and watches include counters. They are used as frequency dividers so a fast input frequency can be output at a slower rate. In devices with memory they are used to increment memory addresses as a program steps through some process. They can activate a series of subcircuits in sequence as part of a complex process. They are, in short, one of the most important workhorses of the digital logic world.

6.1 PURPOSE

This lab has two goals:

1. Develop several different common counters using *D* flip-flops. Because there are two main families of counters, asynchronous and synchronous, this lab includes examples of both.

2. Introduce the *Logisim-Evolution* chronogram feature that generates a timing diagram as a sequential circuit functions.

6.2 PROCEDURE

6.2.1 *Asynchronous Up Counter*

A counter is built from a series of flip-flops and where the output from each flip-flop is combined to create the counter output, trigger the next flip-flop, or both. Each flip-flop is considered a "stage" of the counter. A counter is triggered by a clock signal that is typically supplied by a timer with a regularly-recurring pattern of high/low levels, but it can also be triggered by an event of some sort, like the press of a button or the completion of a process.

In all Counter circuits in this manual flip-flop U0 provides the Least Significant Bit to the output and U3 provides the Most Significant Bit.

One of the simplest counters is illustrated in Figure 6.1. This is an asynchronous four-stage up counter. A counter is is considered "asynchronous" if the input clock signal is applied to only the first stage and then that signal ripples through each flip-flop in turn. Thus, an asynchronous counter is frequently called a "ripple" counter.

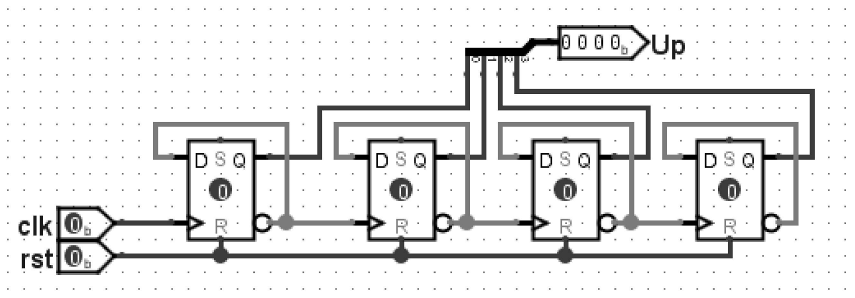


Figure 6.1: Asynchronous Up Counter

The following list describes the operation of the counter in Figure 6.1. Students should open the counter circuit with *Logisim-Evolution* then use the "poke" tool to set the clock high then low (one complete clock cycle) as they follow the description below.

RESET IS ACTIVATED All flip-flops are reset so Q is low and Q' is high.

TICK 1 $U0$ clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 2 $U0$ clocked: $Q0 \downarrow$ — $Q'0 \uparrow$
$U1$ clocked: $Q1 \uparrow$ — $Q'1 \downarrow$

TICK 3 $U0$ clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 4 $U0$ clocked: $Q0 \downarrow$ — $Q'0 \uparrow$
$U1$ clocked: $Q1 \downarrow$ — $Q'1 \uparrow$
$U2$ clocked: $Q2 \uparrow$ — $Q'2 \downarrow$

TICK 5 $U0$ clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 6 $U0$ clocked: $Q0 \downarrow$ — $Q'0 \uparrow$
$U1$ clocked: $Q1 \uparrow$ — $Q'1 \downarrow$

TICK 7 $U0$ clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 8 $U0$ clocked: $Q0 \downarrow$ — $Q'0 \uparrow$
$U1$ clocked: $Q1 \downarrow$ — $Q'1 \uparrow$
$U2$ clocked: $Q2 \downarrow$ — $Q'2 \uparrow$
$U3$ clocked: $Q3 \uparrow$ — $Q'3 \downarrow$

As the clock continues the counter would cycle through the binary values 1001 - 1111. The following table lists the *Up* counter output as indicated by the Q values at each tick listed above.

Tick	Output
Reset	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

Table 6.1: Up Counter Output

6.2.2 *Asynchronous Down Counter*

The asynchronous down counter illustrated in Figure 6.2 is very similar to the up counter in Figure 6.1 except the stages are triggered from the *Q* output of the preceding stage rather than *Q′* and the *Reset* signal is applied to the flip-flop *S* input rather than *R*.

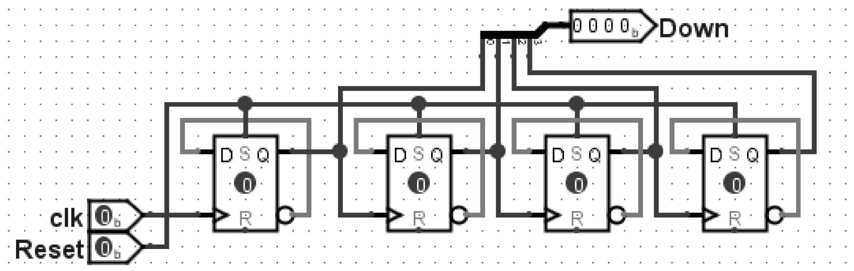


Figure 6.2: Asynchronous Down Counter

The following list describes the operation of the counter in Figure 6.2. Students should open the counter circuit with *Logisim-Evolution* then use the "poke" tool to set the clock high then low (one complete clock cycle) as they follow the description below.

RESET IS ACTIVATED All flip-flops are set so *Q* is high and *Q′* is low.

TICK 1 *U0* clocked: *Q0* ↓ — *Q′0* ↑

TICK 2 *U0* clocked: *Q0* ↑ — *Q′0* ↓
U1 clocked: *Q1* ↓ — *Q′1* ↑

TICK 3 *U0* clocked: *Q0* ↓ — *Q′0* ↑

TICK 4 *U0* clocked: *Q0* ↑ — *Q′0* ↓

U1 clocked: *Q1* ↑ — *Q'1* ↓
U2 clocked: *Q2* ↓ — *Q'2* ↑

TICK 5 *U0* clocked: *Q0* ↓ — *Q'0* ↑

TICK 6 *U0* clocked: *Q0* ↑ — *Q'0* ↓
U1 clocked: *Q1* ↓ — *Q'1* ↑

TICK 7 *U0* clocked: *Q0* ↓ — *Q'0* ↑

TICK 8 *U0* clocked: *Q0* ↑ — *Q'0* ↓
U1 clocked: *Q1* ↑ — *Q'1* ↓
U2 clocked: *Q2* ↑ — *Q'2* ↓
U3 clocked: *Q3* ↓ — *Q'3* ↑

As the clock continues the counter would cycle through the binary values 0110 - 0000. The following table lists the *Down* counter output as indicated by the *Q* values at each tick listed above.

Tick	Output
Reset	1111
1	1110
2	1101
3	1100
4	1011
5	1010
6	1001
7	1000
8	0111

Table 6.2: Down Counter Output

6.2.3 *Asynchronous Decade Counter*

Binary counters, like those considered in Figure 6.1 and Figure 6.2 are only able to count to a value that is a power of two but it is often necessary to build a counter that stops at some other value. These types of counters are called "mod" counters (short for "modulus") since they count up to a preset value then reset and start over, like modulus math. One of the most common mod counters is one that has ten states (it counts from zero to nine) and then resets, and that type of counter is generally referred to as a decade counter. Decade counters are found in any application that has to count in decimal for easy human interpretation.

The logic of a mod counter is to add an `AND` gate on the flip-flop outputs such that the output of the `AND` gate is high when the flip-flop outputs equal the mod number. For example, the `AND` gate for a decade counter would go high when the count reaches ten and that signal would immediately reset the counter back to zero.

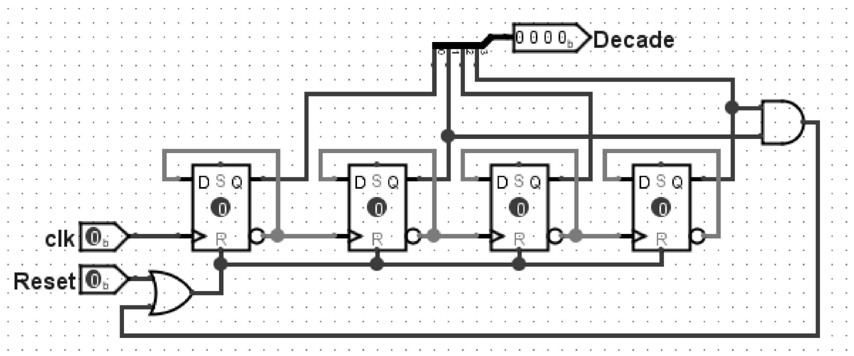


Figure 6.3: Asynchronous Decade Counter

The following list describes the operation of the counter in Figure 6.3:

RESET IS ACTIVATED All flip-flops are reset so Q is low and Q' is high.

TICK 1 *U0* clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 2 *U0* clocked: $Q0 \downarrow$ — $Q'0 \uparrow$
U1 clocked: $Q1 \uparrow$ — $Q'1 \downarrow$

TICK 3 *U0* clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 4 *U0* clocked: $Q0 \downarrow$ — $Q'0 \uparrow$
U1 clocked: $Q1 \downarrow$ — $Q'1 \uparrow$
U2 clocked: $Q2 \uparrow$ — $Q'2 \downarrow$

TICK 5 *U0* clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 6 *U0* clocked: $Q0 \downarrow$ — $Q'0 \uparrow$
U1 clocked: $Q1 \uparrow$ — $Q'1 \downarrow$

TICK 7 *U0* clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 8 *U0* clocked: $Q0 \downarrow$ — $Q'0 \uparrow$
U1 clocked: $Q1 \downarrow$ — $Q'1 \uparrow$
U2 clocked: $Q2 \downarrow$ — $Q'2 \uparrow$
U3 clocked: $Q3 \uparrow$ — $Q'3 \downarrow$

TICK 9 *U0* clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

TICK 10 *U1* clocked: $Q0 \uparrow$ — $Q'0 \downarrow$

Both inputs for the `AND` gate are momentarily high and that sends a reset signal that causes all outputs to go low.

As the clock continues the counter would cycle through the binary values 0000 - 1001. The following table lists the *Decade* counter output as indicated by the *Q* values at each tick listed above.

Tick	Output
Reset	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0000

Table 6.3: Decade Counter Output

6.2.4 *Synchronous Ring Counter*

In a ring counter the high bit is shifted through all of the bits one at a time. This counter is very useful in controlling subcircuits since the high bit in the counter can activate the next subcircuit in the sequence.

The ring counter presented here is also a synchronous circuit; that is, each clock pulse is applied to all of the flip-flops instead of just the first stage. The *Q* output from each flip-flop is used but Q' is not needed at all. Also, there is a feedback line from *U3* to the data input port of *U0* so when the *Q* output of *U3* goes high that is made available to *U0* and loop that value back through the circuit.

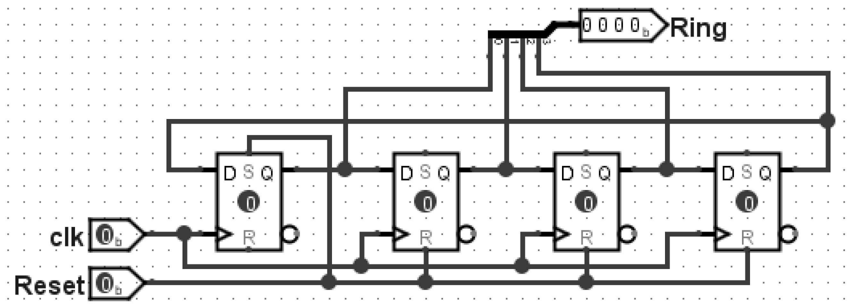


Figure 6.4: Synchronous Ring Counter

The following list describes the operation of the counter in Figure 6.4. Students should open the counter circuit with *Logisim-Evolution* then use the "poke" tool to set the clock high then low (one complete clock cycle) as they follow the description below.

RESET IS ACTIVATED *U0* is set and *U1-U3* are reset so the counter is seeded with a single high bit to shift.

TICK 1 *Q0* ↓ — *Q1* ↑

TICK 2 *Q1* ↓ — *Q2* ↑

TICK 3 *Q2* ↓ — *Q3* ↑

TICK 4 *Q3* ↓ — *Q1* ↑

As the clock continues the counter would cycle through the binary values 0001 - 1000. The following table lists the *ring* counter output as indicated by the *Q* values at each tick listed above.

Tick	Output
Reset	0001
1	0010
2	0100
3	1000
4	0001
5	0010
6	0100
7	1000
8	0001

Table 6.4: Ring Counter Output

6.2.5 *Synchronous Johnson Counter*

A Johnson Counter is similar to a ring counter in that a high bit value is shifted through the entire binary word. The difference is that the feedback loop comes from the Q' output of the last stage rather than the Q output. This type of counter is sometimes called a "twisted tail" counter since the Q' output is fedback.

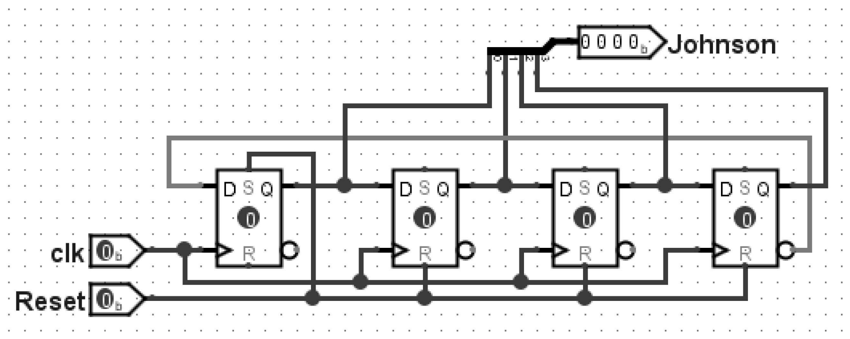


Figure 6.5: Synchronous Johnson Counter

The following list describes the operation of the counter in Figure 6.5. Students should open the counter circuit with *Logisim-Evolution* then use the "poke" tool to set the clock high then low (one complete clock cycle) as they follow the description below.

RESET IS ACTIVATED *U0* is set and *U1*-*U3* are reset so the counter is seeded with a single high bit to shift.

TICK 1 $Q1 \uparrow$

TICK 2 $Q2 \uparrow$

TICK 3 $Q3 \uparrow$

TICK 4 $Q0 \downarrow$

TICK 5 $Q1 \downarrow$

TICK 6 $Q2 \downarrow$

TICK 7 $Q3 \downarrow$

TICK 8 $Q0 \uparrow$

As the clock continues the counter would cycle through the binary values 0000 - 1111. The following table lists the *Johnson* counter output as indicated by the Q values at each tick listed above.

Tick	Output
Reset	0001
1	0011
2	0111
3	1111
4	1110
5	1100
6	1000
7	0000
8	0001

Table 6.5: Johnson Counter Output

6.2.6 *Main*

The `main` circuit provides a human interface to try out each of the counters by dropping them in place of the *Up* counter.

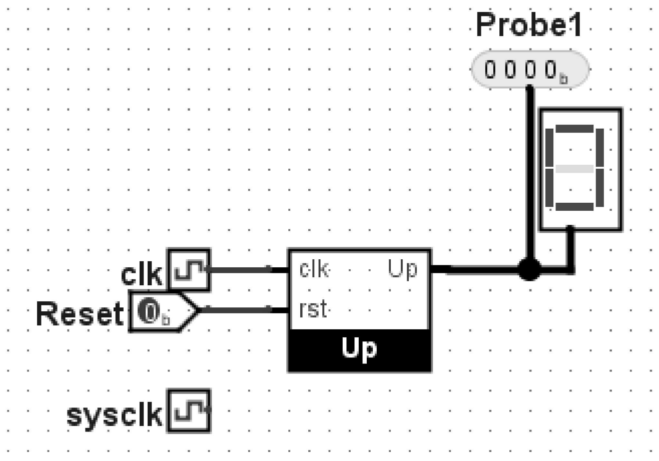


Figure 6.6: Main Circuit

Notice that there are two clocks in the `main` circuit. *Clk* is linked to the counter being tested and is used within the counter circuit to advance the count. *Sysclk* is used by the *Logisim-Evolution* chronogram as described in the next section of this document.

6.2.7 *Chronogram*

Logisim-Evolution can generate a timing diagram, called a *chronogram*, for a sequential circuit. That is a representation of the various signals in a circuit and how those signals change over time. Figure 6.7 is the timing diagram for an Up counter.

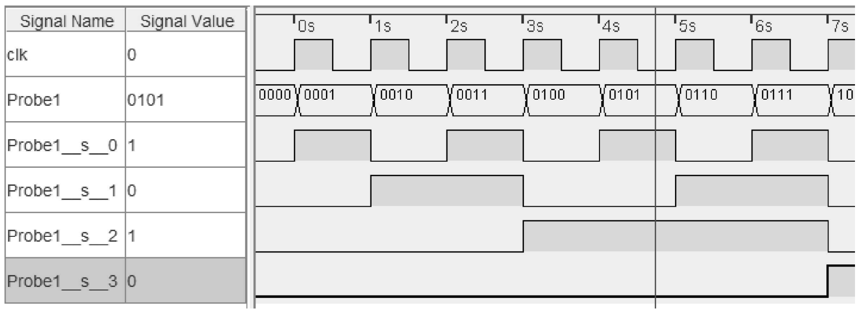


Figure 6.7: Timing Diagram for Up Counter

At the top of Figure 6.7 is a scale that indicates the number of seconds that the counter has been operating. The first trace is the input clk signal. The clock goes high at the start of each second and then goes low at the half-second mark. Under the clock is the "Probe1" signal. Because that is a four-bit number *Logisim-Evolution* displays the number, but under that number is a breakout of the four bits that make up that number. Thus, at time zero "Probe1" is 0001 and "Probe1_s_0" (that stands for "Probe 1, Signal 0") is high while the other bits are low. The *Logisim-Evolution chronogram* includes a cursor indicated by a red line (found just before the five second tick in Figure 6.7) that can be placed anywhere along the diagram. The cursor sets the values of each signal in the area on the left edge of the diagram, so the cursor in Figure 6.7 is pointing to a spot where the *clk* is low, *Probe1* is at 0101, and so forth.

Follow the next steps to use the chronogram. Notes: the chronogram will only check subcircuits that are found on the `main` subcircuit. Therefore, in order to create a timing diagram all subcircuits need to be combined on `main`. The labs completed in this manual have been designed to use the `main` subcircuit as the human interface so the chronogram feature will work well with these circuits.

1. In the `main` subcircuit, add a "sampling clock" labeled *sysclk* (this name is important, do not change it to something else). The sampling clock is only used by the *chronogram* and will not show up in the timing diagram. It should not be connected to any other components and can be placed anywhere on `main`. Set the properties for *sysclk* to a 1 Tick high duration and a 1 Tick low duration (this is the default).

2. Add a circuit master clock labeled *clk*. This is the clock that will be used to trigger all components in the circuit. Set the properties for *clk* to a 4 Tick high duration and a 4 Tick low duration.

3. Set Simulate -> Tick Frequency to 4 Hertz. This will simulate a clock that ticks once per second, as in Figure 6.7. While the

actual tick frequency can be changed later to "speed up" the circuit, a one-second tick is useful for learning how the *chronogram* works.

4. Click SIMULATE -> CHRONOGRAM to set up the *chronogram*. Figure 6.8 illustrates the initial setup screen for the *chronogram*.

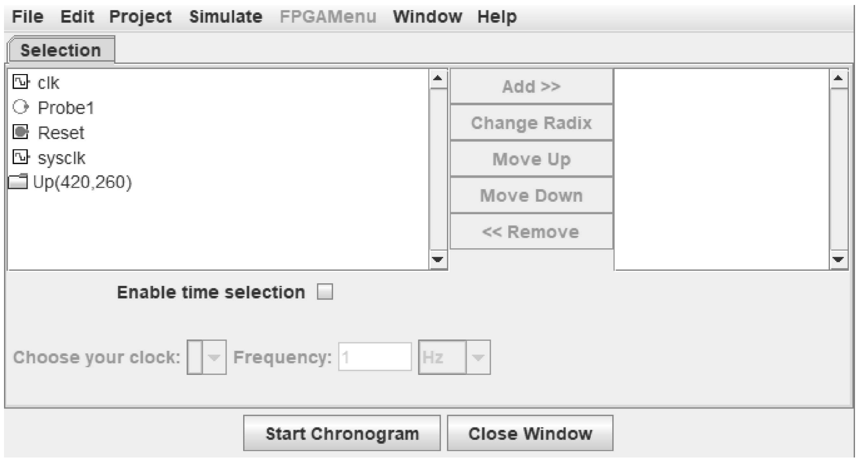


Figure 6.8: Set Up Chronogram

5. Click *sysclk* in the left panel and then click *Add* » to add that signal to the *chronogram*. The "-2" following the *sysclk* name in the right panel indicates that it is a binary signal. It is probably best to add the *sysclk* signal first so it is not overlooked.

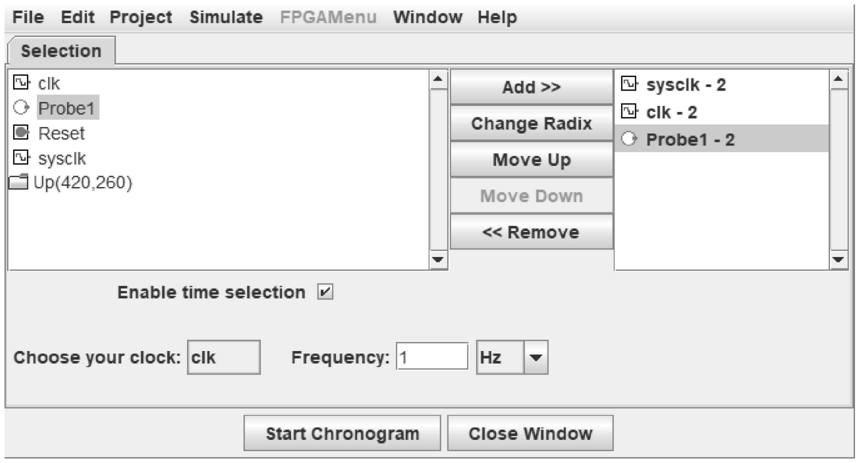
NOTE: sysclk must be added to the chronogram or it will not sample the circuit; however, the sysclk signal will not actually show up in the timing diagram.

6. Click *clk* in the left panel and then click *Add* » to add that signal to the *chronogram*.

7. Click *Probe1* in the left panel and then click *Add* » to add that signal to the *chronogram*.

8. Click "Enable time selection" and chose *clk* as the clock with a frequency of 1 Hertz.

9. The *chronogram* setup should look like Figure 6.9.

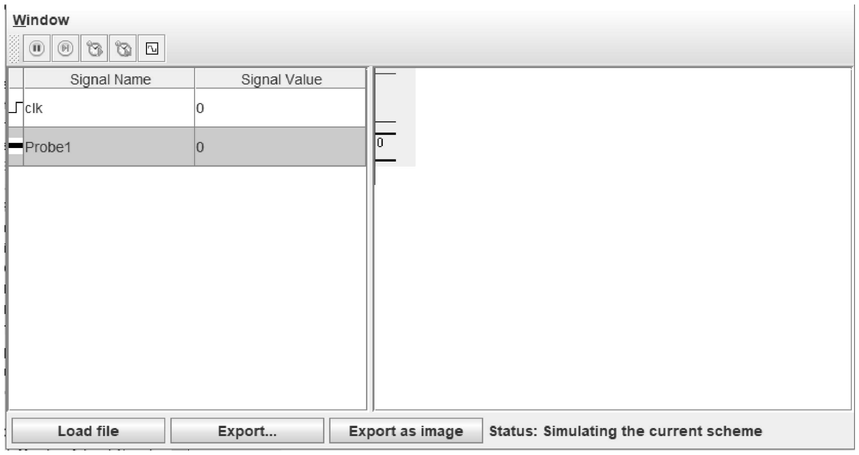


Enable time selection ✓

Choose your clock: clk Frequency: 1 Hz

Figure 6.9: Chronogram Ready

10. Click *Start Chronogram* and the screen illustrated in Figure 6.10 pops up.



Signal Name	Signal Value
clk	0
Probe1	0

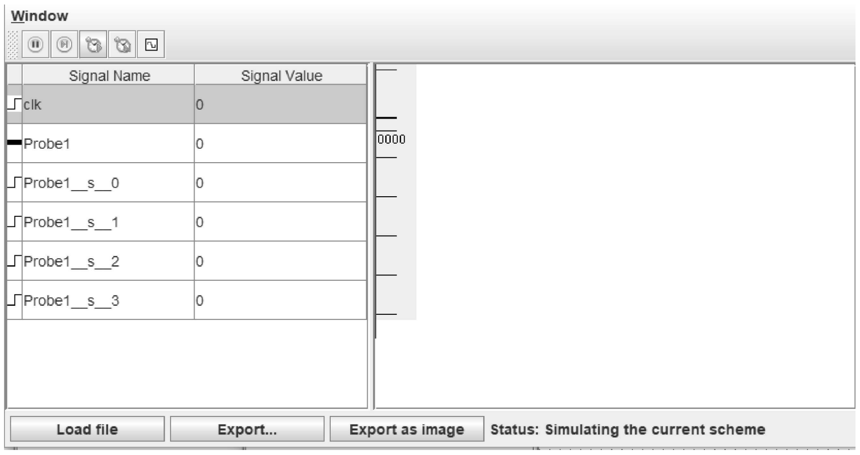
Status: Simulating the current scheme

Figure 6.10: Chronogram Starting

11. Right-click on the *Probe1* signal and set the format for binary. The format can be set for any radix but to match this lab binary numbers should be specified.

12. Right-click on the *Probe1* signal and enable *Expand* to see all four signals that create *Probe1*.

13. At this point, the chronogram should look like Figure 6.11.



Signal Name	Signal Value
clk	0
Probe1	0
Probe1__s__0	0
Probe1__s__1	0
Probe1__s__2	0
Probe1__s__3	0

```
0000
```

Status: Simulating the current scheme

Figure 6.11: Chronogram At Zero Time

14. The *chronogram* has five buttons that control the simulator.

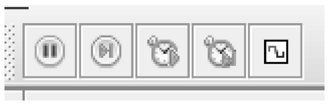


Figure 6.12: Chronogram Controls

- Button One: Start/Stop the simulation.
- Button Two: Simulate one step.
- Button Three: Start/Stop *sysclk*. This will "turn on" the chronogram and begin creating a timing diagram.
- Button Four: Step one *sysclk* tick. This will tick the *sysclk* one time. Since this lab set up the *sysclk* for four ticks per second this button would need to be clicked four times to extend the timing diagram one second.
- Button Five: Step one *clk* tick. This extends the timing diagram by one complete clock tick, or one second in this circuit.

15. Click button three to start the *chronogram* and watch the timing diagram unfold. After a few seconds click that button a second time to stop the *chronogram*.

16. The following can be done once the timing diagram is complete.

- Click on the timing diagram to set the cursor (indicated by a red line). Once the cursor is set the values for each signal at the cursor's location are printed next to the signal's label on the left edge of the timing diagram.
- Hover the mouse over the timing diagram and roll the mouse wheel to zoom the timing diagram appearance.

- Click "Export" to save the timing diagram signal levels in a text file. That file can later be loaded to reevaluate the timing diagram.
- Click "Export as image" to save the timing diagram as a PNG file.

6.3 CHALLENGE

This lab includes several different timers. Place all of them on a single subcircuit named `Universal` that includes an output mux so a user can select the type of counter output desired. Place the `Universal` circuit on `main` and wire appropriate inputs and outputs.

Set up the chronogram for the ring counter and create a ten-second timing diagram for that counter. Save the timing diagram as a PNG image named "RingCounter."

6.4 DELIVERABLE

To receive a grade for this lab, complete the Challenge. Be sure the standard identifying information is at the top left of the `main` circuit:

```
George Self
Lab 06: Counters
March 17, 2018
```

Save the circuit with this name: *Lab07_counter* and submit that along with *RingCounter.PNG* for grading.