GF2: Second Interim Report

Nicholas Capel (nrjc2) - Team 6

May 26, 2016

1 User Guide

1.1 Opening the Logic Simulator and Definition Files

The program can be run by browsing into the main directory and typing `./logsim` followed by the return key. To open and load a definition file, click `File`, then `Open`. Only *.ge* and *.txt* files can be loaded. If there are any errors, they will be displayed in the console display along with their corresponding error codes.

1.2 Running and Continuing the Simulation

In order to run the simulation, specify the number of cycles you wish to run for, then click `Run`. Once you have started the run, you may then continue it by pressing the `Continue` button, once again specifying the number of cycles.

1.3 Viewing the Outputs

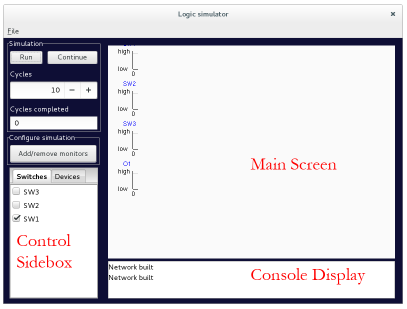
The outputs of the monitored points will then be displayed in the `Main Screen`. You can drag the `Main Screen` window to move around, and use the mouse scroll to zoom in and out.

1.4 Changing Switches and Monitors

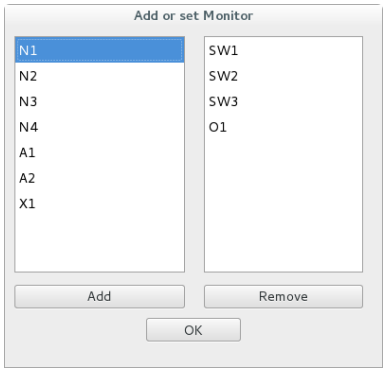
You can set and unset switches by navigating to the `Control Sidebox` and ticking / unticking the dialog boxes. You can add and remove monitors by clicking the `Add/remove monitors` button, selecting the monitors you wish to add or remove, and then pressing either the `Add` or `Remove` button.

1.5 Device List

A list of devices can be found in the `Devices` tab of the `Control Sidebox`.



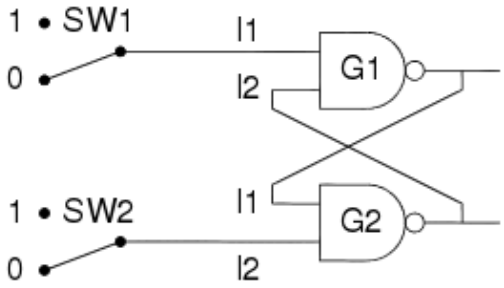
(a) Logic Simulator Main UI



(b) Monitor Modification Display

A Test Definition Files

Figure 1: Bistable Latch

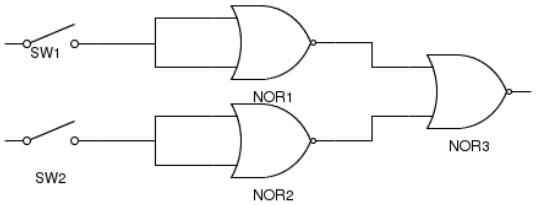



```
DEVICES{
SWITCH SW1:0;
SWITCH SW2:0;
NAND G1:2;
NAND G2:2;
}

CONNECTIONS{
SW1->G1.I1;
SW2->G2.I2;
G1->G2.I1;
G2->G1.I2;
}

MONITORS{
G1;
G2;
}
```

Figure 2: NAND gate implementation



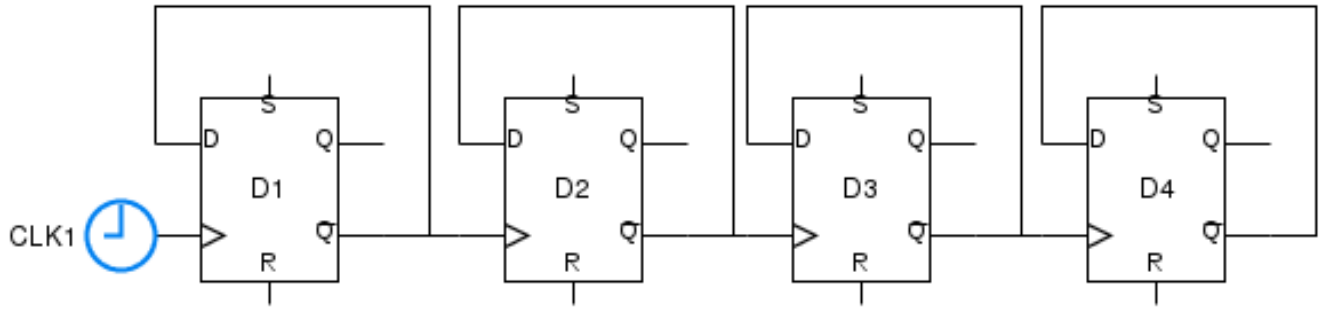

```
// Implementing an AND gate from NOR gates
DEVICES{
SWITCH SW1:0;
SWITCH SW2:0;
NOR NOR1:2;
NOR NOR2:2;
NOR NOR3:2;
}

CONNECTIONS{
SW1->NOR1.I1,NOR1.I2;
SW2->NOR2.I1,NOR2.I2;
```

```
NOR1->NOR3.I1;
NOR2->NOR3.I2;
}

MONITORS{
NOR3;
}
```

Figure 3: Modulo N counter

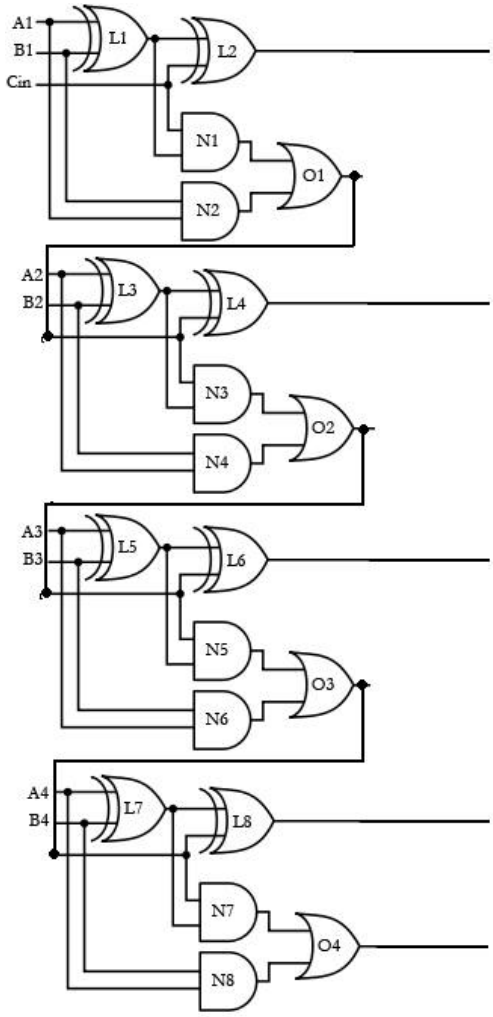



```
// This code implements a modulo n counter.
DEVICES{
SWITCH SW1:0;
DTYPE D1;
DTYPE D2;
DTYPE D3;
DTYPE D4;
CLOCK CLK1:1;
}

CONNECTIONS{
SW1->D1.SET;
SW1->D1.CLEAR;
SW1->D2.SET;
SW1->D2.CLEAR;
SW1->D3.SET;
SW1->D3.CLEAR;
SW1->D4.SET;
SW1->D4.CLEAR;
CLK1->D1.CLK;
D1.QBAR->D2.CLK,D1.DATA;
D2.QBAR->D3.CLK,D2.DATA;
D3.QBAR->D4.CLK,D3.DATA;
D4.QBAR->D4.DATA;
}

MONITORS{
CLK1;
D1.Q;
D2.Q;
D3.Q;
D4.Q;
}
```

Figure 4: 4-bit Adder




```
// This defines the logic of a 4-bit adder.
DEVICES{
SWITCH A1:0;
SWITCH A2:0;
SWITCH A3:0;
SWITCH A4:0;
SWITCH B1:0;
SWITCH B2:0;
SWITCH B3:0;
SWITCH B4:0;
SWITCH CIN:0;
XOR L1;
XOR L2;
XOR L3;
XOR L4;
XOR L5;
XOR L6;
XOR L7;
XOR L8;
AND N1:2;
```

```
AND N2:2;
AND N3:2;
AND N4:2;
AND N5:2;
AND N6:2;
AND N7:2;
AND N8:2;
OR O1:2;
OR O2:2;
OR O3:2;
OR O4:2;
}

CONNECTIONS{
A1->L1.I1,N2.I1;
B1->L1.I2,N2.I2;
CIN->L2.I2,N1.I1;
L1->L2.I1,N1.I2;
N1->O1.I1;
N2->O1.I2;
O1->L4.I2,N3.I1;
A2->L3.I1,N4.I1;
B2->L3.I2,N4.I2;
L3->L4.I1,N3.I2;
N3->O2.I1;
N4->O2.I2;
O2->L6.I2,N5.I1;
A3->L5.I1,N6.I1;
B3->L5.I2,N6.I2;
L5->L6.I1,N5.I2;
N5->O3.I1;
N6->O3.I2;
O3->L8.I2,N7.I1;
A4->L7.I1,N8.I1;
B4->L7.I2,N8.I2;
L7->L8.I1,N7.I2;
N7->O4.I1;
N8->O4.I2;
}

MONITORS{
L2;
L4;
L6;
L8;
O4;
}
```