IIA GF2 Software: 2nd Interim Report

Martin Jackson (mj380) - Team 8

1 User guide

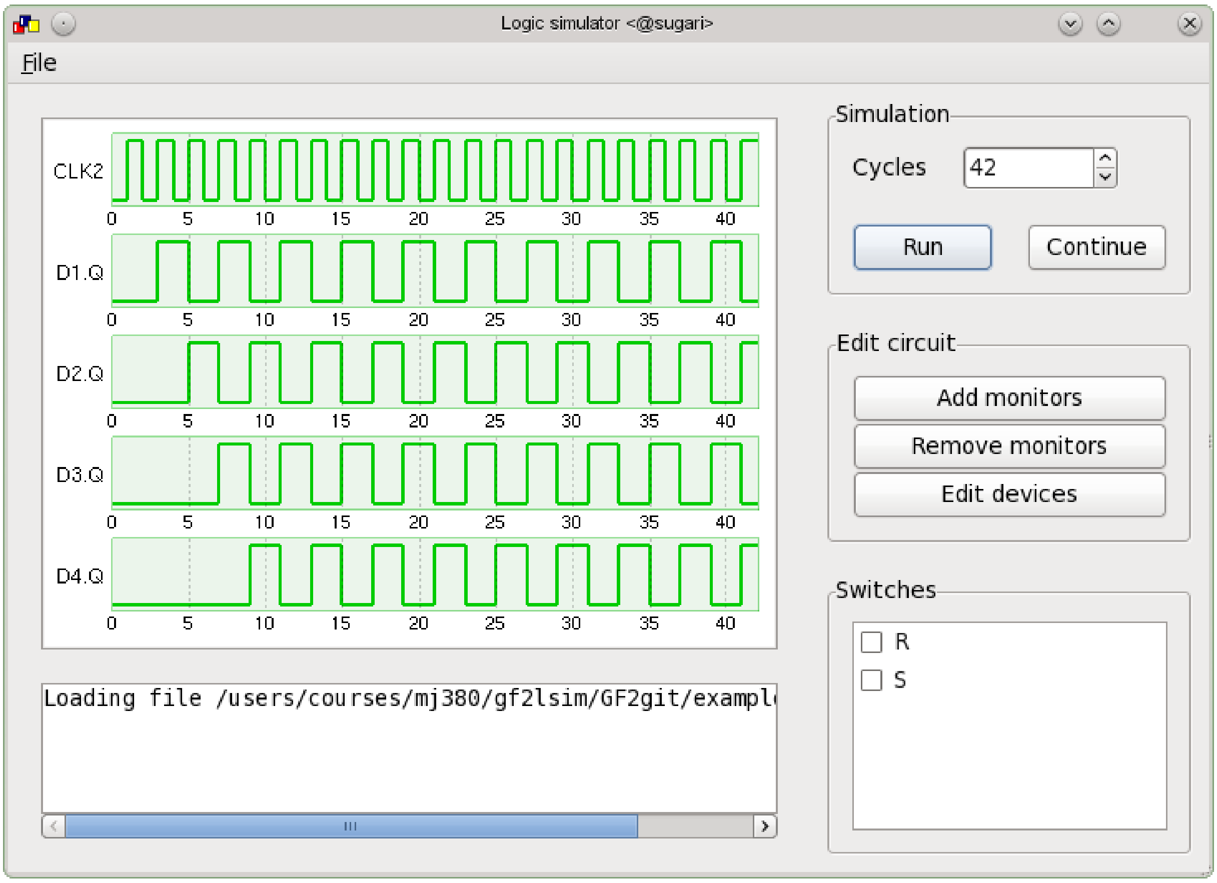
Opening files. There are two methods of opening a file. Either pass the filename as a command line argument when starting the logic simulator, or use the "File → Open" menu item and select the correct file. Logic circuit definition files are plain text files with extension `.gf2`. Examples are provided in section 2. Any errors which occur while loading the file will be shown in the textbox at the bottom of the window.

Running the simulation. After opening a circuit, select the number of cycles to simulate using the textbox in the top right corner. Then click the "Run" button just below the textbox. Results will be displayed in the large space to the left (see figure 1). The run button clears the displayed results and simulates the circuit for the specified number of cycles. To simulate for additional cycles without clearing the displayed results, use the "Continue" button instead. The only limit on the number of displayed signals and simulated cycles is the available memory on the computer. If necessary, scrollbars will appear around the result display area.

Adding or removing monitors. To change which signals are monitored and displayed, use the add or remove monitors buttons on the right. The add monitors button will show a list of unmonitored outputs that can be selected for monitoring, and the remove monitors button a list of current monitors that can be removed. In both cases, multiple items may be selected. After adding a monitor, you must use the run button before using the continue button, as no samples were recorded for the new monitor during the cycles displayed on screen, so the recorded signals cannot be continued, they must be reset first.

Changing switch states. Use the checkboxes located in the bottom right corner of the window. Changes take effect immediately. Run or continue the simulation to see the effect on the circuit.

Editing devices. Click the edit devices button on the right hand side. This will open a window allowing you to add, edit, or delete devices (see figure 2). Select a device to edit using the list on the left. The top panel allows device properties to be modified. The bottom panels show inputs and outputs for the selected device, and which devices are connected to it. They also allow connections to be added and deleted. All changes to devices and connections take effect immediately, except for changes to device properties, which must be confirmed using the "Apply changes" button.

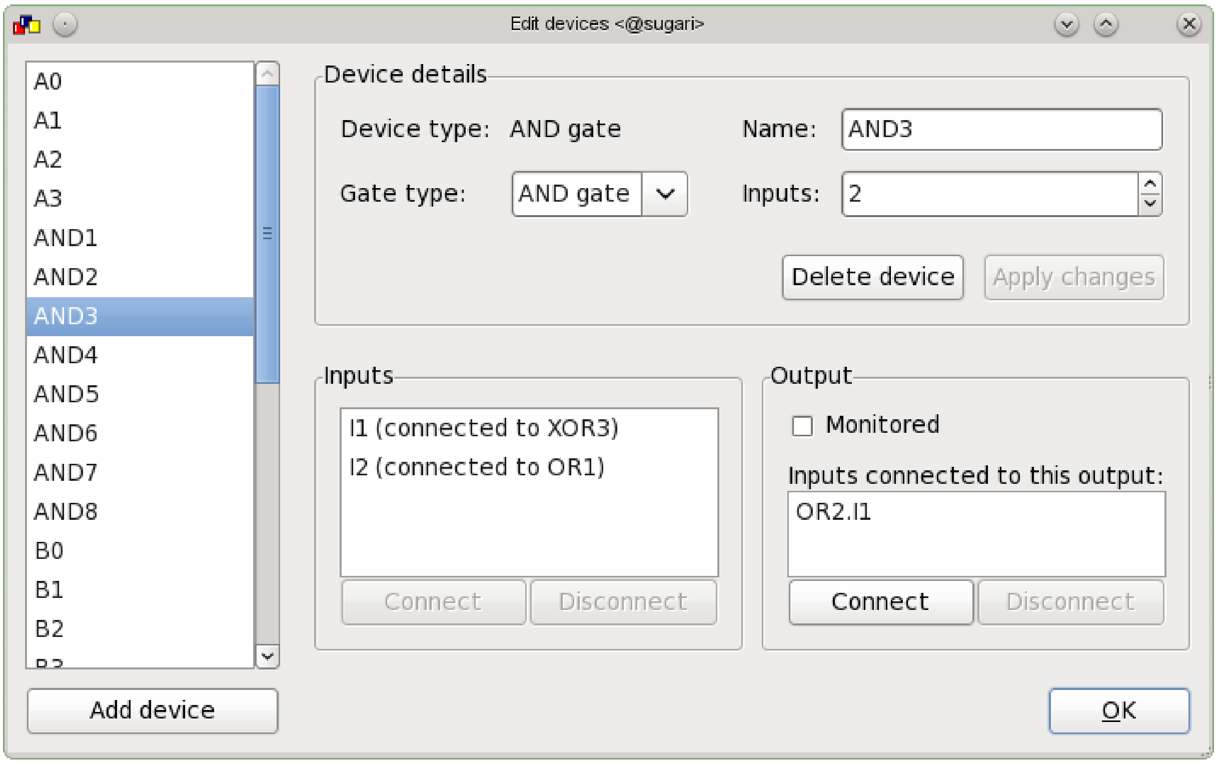


Cycles: 42

```
Loading file /users/courses/mj380/gf2lsim/GF2git/exampl
```

□ R
□ S

Figure 1: Main logic simulator window



Device type: AND gate

Name: AND3

Gate type: AND gate

Inputs: 2

I1 (connected to XOR3)
I2 (connected to OR1)

□ Monitored

Inputs connected to this output:
OR2.I1

Figure 2: GUI for editing devices

2 Example circuits

Jamie Magee made all definition files and circuit diagrams shown here.

2.1 XOR Gate

2.1.1 Definition File

```
DEVICES
SWITCH S1:0;
SWITCH S2:1;
NAND G1:2;
NAND G2:2;
NAND G3:2;
NAND G4:2;
END

CONNECTIONS
G1.I1 = S1;
G1.I2 = S2;
G2.I1 = S1;
G2.I2 = G1;
G3.I1 = G1;
G3.I2 = S2;
G4.I1 = G2;
G4.I2 = G3;
END

MONITORS
S1;
S2;
G4;
END
```

Listing 1: xor.gf2

2.1.2 Circuit Diagram

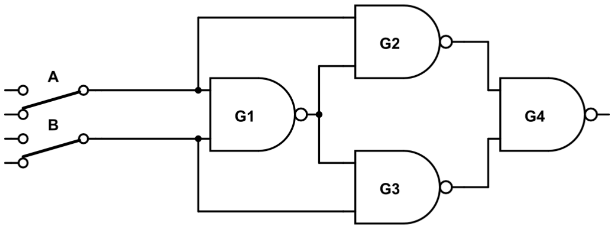


Figure 3: Circuit diagram of an XOR gate implemented using NAND gates

2.2 4-bit Adder

2.2.1 Definition File

```
DEVICES
/* 4 bit inputs */
SWITCH A0:1;
SWITCH A1:0;
SWITCH A2:0;
SWITCH A3:0;
SWITCH B0:1;
SWITCH B1:0;
SWITCH B2:0;
SWITCH B3:0;
SWITCH C0:1; /* Carry in */
AND AND1:2;
AND AND2:2;
AND AND3:2;
AND AND4:2;
AND AND5:2;
AND AND6:2;
AND AND7:2;
AND AND8:2;
XOR XOR1;
XOR XOR2;
XOR XOR3;
XOR XOR4;
XOR XOR5;
XOR XOR6;
XOR XOR7;
XOR XOR8;
OR OR1:2;
OR OR2:2;
OR OR3:2;
OR OR4:2;
END

CONNECTIONS
/* LSB adder */
XOR1.I1 = A0;
XOR1.I2 = B0;
AND1.I1 = XOR1;
AND1.I2 = C0;
AND2.I1 = A0;
AND2.I2 = B0;
XOR2.I1 = XOR1;
XOR2.I2 = C0;
OR1.I1 = AND1;
OR1.I2 = AND2;

XOR3.I1 = A1;
XOR3.I2 = B1;
AND3.I1 = XOR3;
AND3.I2 = OR1;
AND4.I1 = A1;
AND4.I2 = B1;
XOR4.I1 = XOR3;
XOR4.I2 = OR1;
OR2.I1 = AND3;
OR2.I2 = AND4;

XOR5.I1 = A2;
XOR5.I2 = B2;
AND5.I1 = XOR5;
AND5.I2 = OR2;
AND6.I1 = A2;
AND6.I2 = B2;
XOR6.I1 = XOR5;
XOR6.I2 = OR2;
OR3.I1 = AND5;
OR3.I2 = AND6;

/* MSB Adder */
XOR7.I1 = A3;
XOR7.I2 = B3;
```

```
AND7.I1 = XOR7;
AND7.I2 = OR3;
AND8.I1 = A3;
AND8.I2 = B3;
XOR8.I1 = XOR7;
XOR8.I2 = OR3;
OR4.I1 = AND7;
OR4.I2 = AND8;
END

MONITORS
/* Outputs */
XOR2;
XOR4;
XOR6;
XOR8;
OR4; /* Carry out */
END
```

Listing 2: 4bitadder.gf2

2.2.2 Circuit Diagram

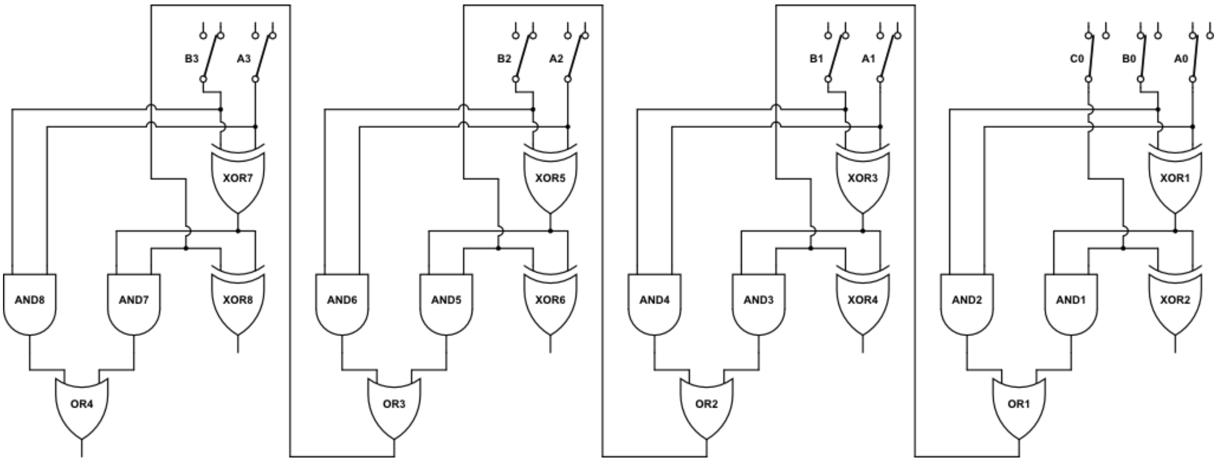


Figure 4: Circuit diagram of a 4-bit adder

2.3 Serial In Parallel Out Shift Register

2.3.1 Definition File

```
DEVICES
CLOCK CLK1:2;
CLOCK CLK2:1;
SWITCH S:0; /* Set switch */
SWITCH R:0; /* Reset switch */
DTYPE D1;
DTYPE D2;
DTYPE D3;
DTYPE D4;
END

CONNECTIONS
D1.DATA = CLK1;
D2.DATA = D1.Q;
D3.DATA = D2.Q;
D4.DATA = D3.Q;
```

```
D1.CLK = CLK2;
D2.CLK = CLK2;
D3.CLK = CLK2;
D4.CLK = CLK2;
D1.SET = S;
D2.SET = S;
D3.SET = S;
D4.SET = S;
D1.CLEAR = R;
D2.CLEAR = R;
D3.CLEAR = R;
D4.CLEAR = R;
END

MONITORS
CLK2;
D1.Q;
D2.Q;
D3.Q;
D4.Q;
END
```

Listing 3: sipo.gf2

2.3.2 Circuit Diagram

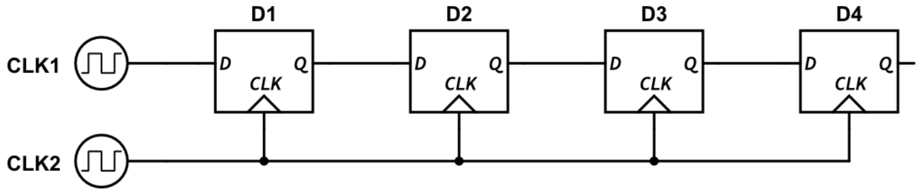


Figure 5: Circuit diagram of a serial in parallel out shift register

NB The software used to draw the circuit diagram does not support the same style of D flip-flop used in the definition file, and Fig. 5 was the closest achievable.

2.4 Gated D Latch

2.4.1 Definition File

```
DEVICES
CLOCK CLK1:1;
CLOCK CLK2:2;
NAND G1:1;
AND G2:2;
AND G3:2;
NOR G4:2;
NOR G5:2;
END

CONNECTIONS
G1.I1 = CLK1;
G2.I1 = G1;
G2.I2 = CLK2;
G3.I1 = CLK2;
G3.I2 = CLK1;
G4.I1 = G2;
G4.I2 = G5;
G5.I1 = G4;
```

```
G5.I2 = G3;
END

MONITORS
CLK1; /* D */
CLK2; /* E */
G4; /* Q */
G5; /* QBAR */
END
```

Listing 4: sipo.gf2

2.4.2 Circuit Diagram

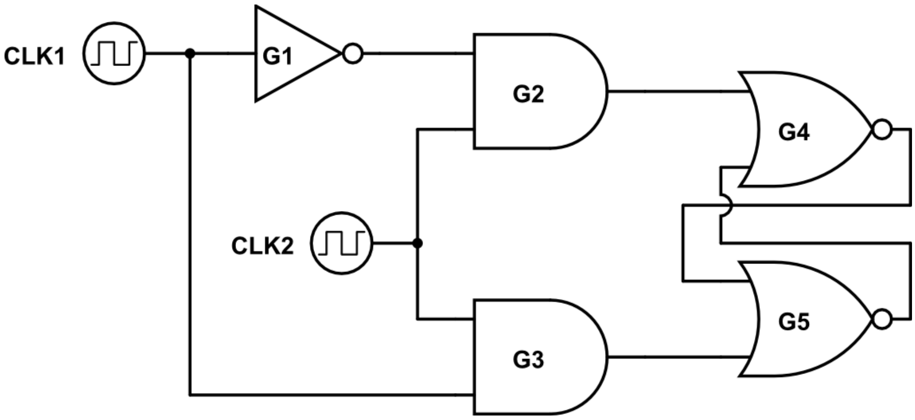


Figure 6: Circuit diagram of a Gated D Latch

NB The software used to draw the circuit diagram does not support NAND gates with one input. Therefore the NAND gate G1 was substituted for a NOT gate as can be seen in Fig. 6.