Project GF2: Software
Second Interim Report
Software Design Team 1

George Ayris	James Glanville	**Andrew Holt**
gdwa2	jg597	**ah635**
Emmanuel	Emmanuel	**Emmanuel**
Parser & Scanner	Scanner & Parser	**GUI**

31 May 2013

1 Introduction

This report gives a brief summary of the graphical user interface (gui) development for the GF2 Software Project. Version 1.0 of the logic simulation package has been produced and tested. In addition to the program file listings, test files for the software and the corresponding circuit diagrams are shown. Finally, a single page user guide describing the operation of the software package is given.

2 File Listings and Descriptions

For the gui development, the vast majority of my code has been in the `gui.h` and `gui.cc` files, and primarily concerned with the `MyFrame` class, with some fairly minor modifications made to the `MyGLCanvas` class. These files are listed in full in appendix

The header file, `gui.h`, contains the class and function templates for `MyFrame` and `MyGLCanvas`. It also contains an `enum` type, defining the widgets used in the gui.

The main file, `gui.cc`, contains the definitions for `MyFrame` and `MyGLCanvas`. Considerable modifications have been made to the `MyGLCanvas::Render` method. For the gui implemented, multiple canvases were required, each showing just a single trace. For this purpose, an `int` was passed to the function to determine which monitor to use to get the trace. The contents of the `MyGLCanvas::OnMouse` was deleted as no mouse interaction is desired. The code will be completely removed given more time to ensure no adverse side effects are brought about by the removal of the function.

The event table is defined, starting from line 126. This ties the events generated by user interaction to the functions and procedures defined in the callback functions.

The constructor underwent heavy modification to produce a more flexible, more usable user interface. The canvases are stored in a vector, and two more vectors contain the sizers and labels for the canvases. This design offers great flexibility and easy control over multiple traces, and they can be easily hidden and shown to avoid unnecessary traces distracting from the important information.

Additionally, a terminal style text output was implemented in the gui to display error messages. A particular challenge was to ensure that a monospaced font was used in the text display so that the error messages are displayed correctly and showing the error in the right place.

The callback functions are largely self explanatory. One neat feature in these is the implementation of the `OnAddMonitor` and `OnRemMonitor`, which ensure that the multiple canvases and sizers display correctly and the lists of monitors that may be added or removed are properly maintained.

3 Test Files and Circuit Diagrams [1]

Two main test files were used to test the whole system. In addition to these, many test files with carefully devised errors were used for testing the parser and scanner. A limited amount of user testing was run to ensure the gui operated as expected.

The first test file defines a four bit shift register. This type of circuit uses bistables to pass signals through the array on each clock pulse (by convention, they tend to be rising edge triggered). Shift registers find heavy application in driving serial buses from parallel data sources, and are widespread in may other uses too. The circuit diagram is shown in figure . The definition file is listed in appendix B.

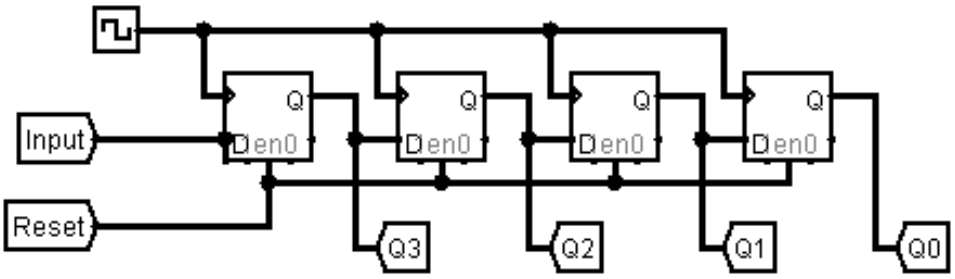

Figure 1: 4 bit shift register.

The other main testing file defined a simple gate array. The gate array was selected as it utilises all of the required gate types. Between the gate array and the shift register all the devices required are tested. While this particular gate array has no specific purpose, logic gate arrays are very common in digital electronic design, and the design of such circuits would be a primary usage of a software package such as this. The circuit diagram is shown in figure 2. The definition file is shown in appendix B.

4 User Guide

4.1 Starting the Program

The logic simulation package may be run by running the command `./logsim` in the directory of the compiled file. The definition file to be tested should be passed as an argument.[2]

4.2 Definition Errors

Any errors in the definition file will be highlighted in the message display at the bottom of the program window. This will give detailed information about the cause and location of the error, making debugging fast and efficient. Once all errors have been corrected the program may be run fully.

4.3 Setting up Monitors and Switches

Once the definition file has been loaded, the **load** button should be pressed. This loads the monitors and switches into the dropdown menus for selection. To add a monitor point, simply select the desired monitor from the list in the **add monitor** drop-down menu. They may be removed by selecting from the **remove monitor** menu. To ensure all required information is visible together, without many distracting and irrelevant output traces, up to 10 monitors may be observed at once. The value of a switch may be set at any time during the simulation by selecting the desired switch from the **switches** drop-down menu, followed by the level to be set from the **switch value** menu.

[1] Shared between all of team.

[2] Development of a file selection dialogue from within the main graphical interface has begun, but was not polished or tested enough to make it into this release.

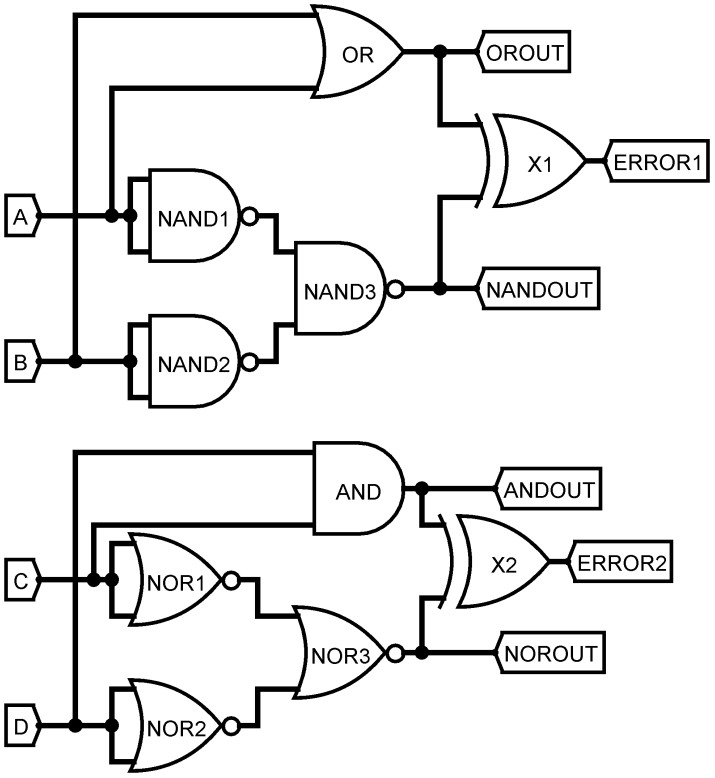

Figure 2: Logic Gate Array

4.4 Running and Continuing the Simulation

To run the simulation, simply select the number of clock cycles to simulate in a single run (between 1 and 50), then press the **run** button. The traces for all defined monitors will be shown. The simulation may be **continued**, to observe the long term behaviour or to investigate the effect of a changed switch level on the circuit. The simulation may be run for up to 350 clock cycles. If the simulation is run for a large number of cycles, the traces may no longer be visible. In this case, simply drag the window to become wider and the traces will increase in size so that more clock cycles are visible.

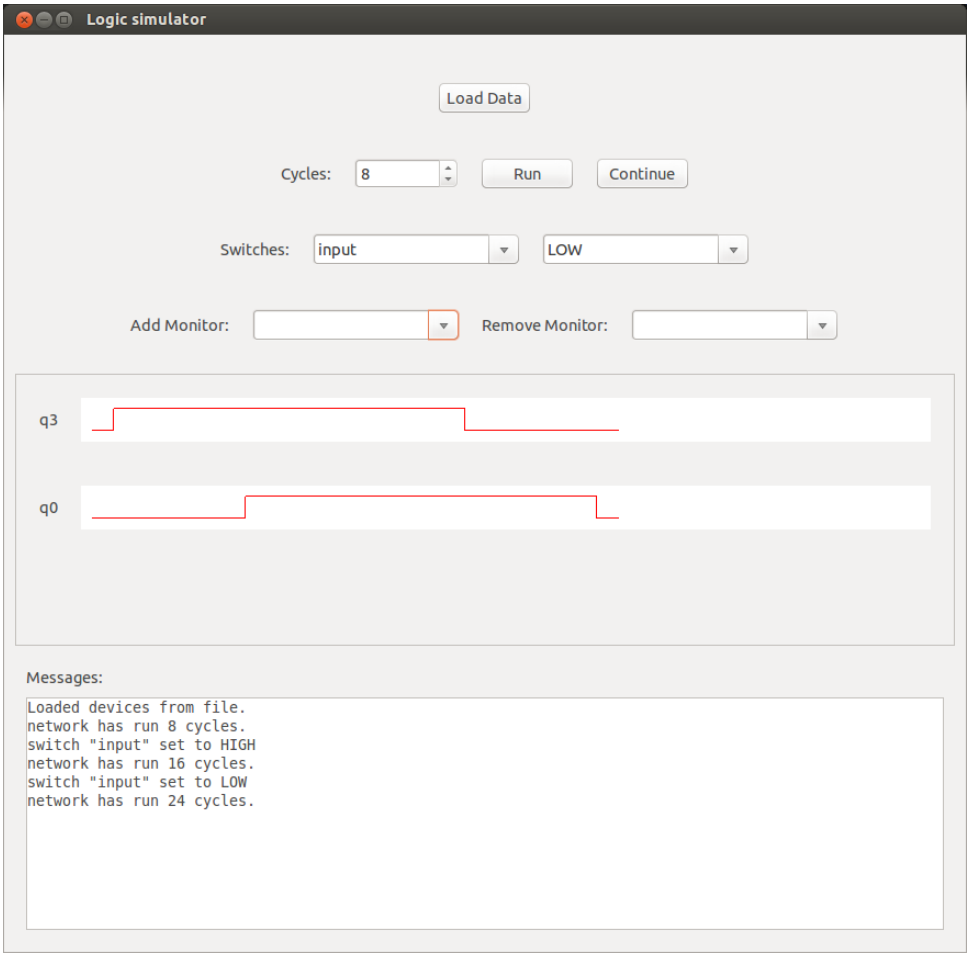

Figure 3: Screenshot of Logic Simulator Running.

A Source Code

A.1 gui.h

```
#ifndef gui_h
#define gui_h

#include <wx/wx.h>
#include <wx/glcanvas.h>
#include <wx/spinctrl.h>
#include <wx/textctrl.h>
#include "names.h"
#include "devices.h"
#include "monitor.h"

enum {
  CYCLES_SPIN = wxID_HIGHEST + 1,
  //  FILE_BUTTON,
  LOAD_BUTTON,
  RUN_BUTTON_ID,
  CONT_BUTTON_ID,
  DISP_SCROLL,
  SWITCH_LIST,
  SWITCH_OPTION,
  MONITOR_ADD,
  MONITOR_REM,
```

```
}; // widget identifiers

class MyGLCanvas;

class MyFrame: public wxFrame
{
 public:
  MyFrame(wxWindow *parent,
          const wxString& title,
          const wxPoint& pos,
          const wxSize& size,
          names *names_mod = NULL,
          devices *devices_mod = NULL,
          monitor *monitor_mod = NULL,
          long style = wxDEFAULT_FRAME_STYLE); // constructor

  //  string filetoopen = "";
 private:
  vector<wxBoxSizer*> vtracesizers;       /* Vector to hold sizers for traces */
  vector<MyGLCanvas*> canvases;           /* vector to hold canvases */
  vector<wxStaticText*> tracelabels;

  wxString tracesizer;
  wxString tracename;
  string devname;

  wxBoxSizer *topsizer;
  wxBoxSizer *toptracesizer;

  wxSpinCtrl *spin_cycles;
  wxComboBox *switch_list;
  wxComboBox *switch_option;
  wxComboBox *add_monitor;
  wxComboBox *rem_monitor;
  wxWindow *disp_scroll;

  names *nmz;                             // pointer to names class
  devices *dmz;                           // pointer to devices class
  monitor *mmz;                           // pointer to monitor class
  int cyclescompleted;                    // how many simulation cycles have been completed
  void runnetwork(int ncycles);           // function to run the logic network

  void OnExit(wxCommandEvent& event);     // callback for exit menu item
  void OnAbout(wxCommandEvent& event);    // callback for about menu item
  //  void OnFileButton(wxCommandEvent &event);
  void OnLoadButton(wxCommandEvent &event);
  void OnRunButton(wxCommandEvent &event);
  void OnContButton(wxCommandEvent &event);
  void OnSwitchSelect(wxCommandEvent& event);
  void OnSwitchOption(wxCommandEvent& event);
  void OnAddMonitor(wxCommandEvent& event);
  void OnRemMonitor(wxCommandEvent& event);
  DECLARE_EVENT_TABLE()
};

class MyGLCanvas: public wxGLCanvas
{
 public:
  MyGLCanvas(wxWindow *parent,
             wxWindowID id = wxID_ANY,
             monitor* monitor_mod = NULL,
             names* names_mod = NULL,
             const wxPoint& pos = wxDefaultPosition,
             const wxSize& size = wxDefaultSize,
             long style = 0,
             const wxString& name = wxT("MyGLCanvas")); // constructor
  void Render(int monren = 0, int cycles = -1); // function to draw canvas contents

 private:
  bool init;                          // has the GL context been initialised?
```

```
    int cyclesdisplayed;                // how many simulation cycles have been displayed
    monitor *mmz;                       // pointer to monitor class, used to extract signal traces
    names *nmz;                         // pointer to names class, used to extract signal names
    void InitGL();                      // function to initialise GL context
    void OnSize(wxSizeEvent& event);    // callback for when canvas is resized
    void OnPaint(wxPaintEvent& event); // callback for when canvas is exposed
    void OnMouse(wxMouseEvent& event); // callback for mouse events inside canvas
    DECLARE_EVENT_TABLE()
};

#endif /* gui_h */
```

A.2 gui.cc

```
#include "gui.h"
#include <GL/glut.h>
#include "wx_icon.xpm"
#include <iostream>
#include <string>
#include <sstream>
#include <wx/font.h>

using namespace std;

wxTextCtrl* textout;

// MyGLCanvas ////////////////////////////////////////////////////////////////

BEGIN_EVENT_TABLE(MyGLCanvas, wxGLCanvas)
  EVT_SIZE(MyGLCanvas::OnSize)
  EVT_PAINT(MyGLCanvas::OnPaint)
  EVT_MOUSE_EVENTS(MyGLCanvas::OnMouse)
END_EVENT_TABLE()

int wxglcanvas_attrib_list[5] = {WX_GL_RGBA,
                                 WX_GL_DOUBLEBUFFER,
                                 WX_GL_DEPTH_SIZE,
                                 16, 0};

MyGLCanvas::MyGLCanvas(wxWindow *parent,
                       wxWindowID id,
                       monitor* monitor_mod,
                       names* names_mod,
                       const wxPoint& pos,
                       const wxSize& size,
                       long style,
                       const wxString& name):
    wxGLCanvas(parent, id, pos, size, style, name, wxglcanvas_attrib_list)
  // Constructor - initialises private variables
{
  mmz = monitor_mod;
  nmz = names_mod;
  init = false;
  cyclesdisplayed = -1;
}

void MyGLCanvas::Render(int monren, int cycles)
{
  float y;
  unsigned int i;
  asignal s;

  if (cycles >= 0) cyclesdisplayed = cycles;

  SetCurrent();
  if (!init) {
    InitGL();
    init = true;
  }
  glClear(GL_COLOR_BUFFER_BIT);
```

```
57
    if ((cyclesdisplayed >= 0) && (mmz->moncount() > 0)) {
      // draw the first monitor signal, get trace from monitor class
      glColor3f(1.0, 0.0, 0.0);
      glBegin(GL_LINE_STRIP);
62    for (i=0; i<cyclesdisplayed; i++) {
        if (mmz->getsignaltrace(monren, i, s)) {
          if (s==low) y = 10.0;
          if (s==high) y = 30.0;
          glVertex2f(20*i+10.0, y);
67        glVertex2f(20*i+30.0, y);
        }
      }
      glEnd();

72  }

    // We've been drawing to the back buffer, flush the graphics
    // pipeline and swap the back buffer to the front
    glFlush();
77  SwapBuffers();
}

void MyGLCanvas::InitGL()
  // Function to initialise the GL context
82{
  int w, h;

  GetClientSize(&w, &h);
  SetCurrent();
87  glDrawBuffer(GL_BACK);
  glClearColor(1.0, 1.0, 1.0, 0.0);
  glViewport(0, 0, (GLint) w, (GLint) h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
92  glOrtho(0, w, 0, h, -1, 1);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}

97 void MyGLCanvas::OnPaint(wxPaintEvent& event)
  // Callback function for when the canvas is exposed
{
  int w, h;
  wxString text;
102
  wxPaintDC dc(this); // required for correct refreshing under MS windows
  GetClientSize(&w, &h);
  //text.Printf(wxT("Canvas redrawn by OnPaint callback, canvas size
  //is %d by %d"), w, h);
107 Render();
}

void MyGLCanvas::OnSize(wxSizeEvent& event)
  // Callback function for when the canvas is resized
112{
  wxGLCanvas::OnSize(event); // required on some platforms
  init = false;
  Refresh(); // required by some buggy nvidia graphics drivers,
  Update();  // harmless on other platforms!
117}

void MyGLCanvas::OnMouse(wxMouseEvent& event)
  // Callback function for mouse events inside the GL canvas
{}
122
// MyFrame ////////////////////////////////////////////////////////////////////


BEGIN_EVENT_TABLE(MyFrame, wxFrame)
```

```
  EVT_MENU(wxID_EXIT,        MyFrame::OnExit)
  EVT_MENU(wxID_ABOUT,       MyFrame::OnAbout)
  //EVT_BUTTON(FILE_BUTTON, MyFrame::OnFileButton)
  EVT_BUTTON(LOAD_BUTTON, MyFrame::OnLoadButton)
  EVT_BUTTON(RUN_BUTTON_ID, MyFrame::OnRunButton)
  EVT_BUTTON(CONT_BUTTON_ID, MyFrame::OnContButton)
  EVT_COMBOBOX(SWITCH_LIST, MyFrame::OnSwitchSelect)
  EVT_COMBOBOX(SWITCH_OPTION, MyFrame::OnSwitchOption)
  EVT_COMBOBOX(MONITOR_ADD, MyFrame::OnAddMonitor)
  EVT_COMBOBOX(MONITOR_REM, MyFrame::OnRemMonitor)
END_EVENT_TABLE()

MyFrame::MyFrame(wxWindow *parent,
                 const wxString& title,
                 const wxPoint& pos,
                 const wxSize& size,
                 names *names_mod,
                 devices *devices_mod,
                 monitor *monitor_mod,
                 long style):
  wxFrame(parent, wxID_ANY, title, pos, size, style)
  // Constructor - initialises pointers to names, devices and monitor
  // classes, lays out widgets using sizers
{
  SetIcon(wxIcon(wx_icon));

  nmz = names_mod;
  dmz = devices_mod;
  mmz = monitor_mod;
  if (nmz == NULL || dmz == NULL || mmz == NULL)
    {
      cout << "Cannot␣operate␣GUI␣without␣names,␣devices␣and␣monitor␣classes"
           << endl;
      exit(1);
    }

  wxMenu *fileMenu = new wxMenu;
  fileMenu->Append(wxID_EXIT, wxT("&Quit\tCtrl-Q"));
  wxMenu *helpMenu = new wxMenu;
  helpMenu->Append(wxID_ABOUT, wxT("&About"));
  wxMenuBar *menuBar = new wxMenuBar;
  menuBar->Append(fileMenu, wxT("&File"));
  menuBar->Append(helpMenu, wxT("&Help"));
  SetMenuBar(menuBar);

  topsizer = new wxBoxSizer(wxVERTICAL);

  wxBoxSizer *filesizer = new wxBoxSizer(wxHORIZONTAL);
  //  filesizer->Add(new wxButton(this, FILE_BUTTON, wxT("Select File")),
  //             0,
  //             wxALL | wxALIGN_CENTER_VERTICAL,
  //             10);
  filesizer->Add(new wxButton(this, LOAD_BUTTON, wxT("Load␣Data")),
                 0,
                 wxALL | wxALIGN_CENTER_VERTICAL,
                 10);
  topsizer->Add(filesizer, 0, wxALIGN_CENTER_HORIZONTAL | wxALL, 10);

  wxBoxSizer *ctrlsizer = new wxBoxSizer(wxHORIZONTAL);
  ctrlsizer->Add(new wxStaticText(this, wxID_ANY, wxT("Cycles:")),
                 0,
                 wxALL | wxALIGN_CENTER_VERTICAL,
                 10);
  spin_cycles = new wxSpinCtrl(this, CYCLES_SPIN, wxString(wxT("10")),
                               wxDefaultPosition,
                               wxDefaultSize,
                               wxSP_ARROW_KEYS,
                               1, 50, 10);
  ctrlsizer->Add(spin_cycles, 0, wxALL | wxALIGN_CENTER_VERTICAL, 10);
  ctrlsizer->Add(new wxButton(this, RUN_BUTTON_ID, wxT("Run")),
```

```
                0,
                wxALL | wxEXPAND,
                10);
    ctrlsizer->Add(new wxButton(this, CONT_BUTTON_ID, wxT("Continue")),
                0,
                wxALL | wxEXPAND,
                10);
    topsizer->Add(ctrlsizer, 0, wxALIGN_CENTER_HORIZONTAL | wxALL, 10);

    wxBoxSizer *switchsizer = new wxBoxSizer(wxHORIZONTAL);
    switchsizer->Add(new wxStaticText(this, wxID_ANY, wxT("Switches:")),
                0,
                wxALL | wxALIGN_CENTER_VERTICAL,
                10);
    switch_list = new wxComboBox(this, SWITCH_LIST, wxEmptyString);
    switchsizer->Add(switch_list, 0, wxALL | wxALIGN_CENTER_VERTICAL, 10);
    switch_option = new wxComboBox(this, SWITCH_OPTION, wxEmptyString);
    switch_option->Append(wxT("HIGH"));
    switch_option->Append(wxT("LOW"));
    switchsizer->Add(switch_option, 0, wxALL | wxALIGN_CENTER_VERTICAL, 10);
    topsizer->Add(switchsizer, 0, wxALIGN_CENTER_HORIZONTAL | wxALL, 10);

    wxBoxSizer *monitorsizer = new wxBoxSizer(wxHORIZONTAL);
    monitorsizer->Add(new wxStaticText(this, wxID_ANY, wxT("Add␣Monitor:")),
                0,
                wxALL | wxALIGN_CENTER_VERTICAL,
                10);
    add_monitor = new wxComboBox(this, MONITOR_ADD, wxEmptyString);
    monitorsizer->Add(add_monitor, 0, wxALL | wxALIGN_CENTER_VERTICAL, 10);
    monitorsizer->Add(new wxStaticText(this, wxID_ANY, wxT("Remove␣Monitor:")),
                0,
                wxALL | wxALIGN_CENTER_VERTICAL,
                10);
    rem_monitor = new wxComboBox(this, MONITOR_REM, wxEmptyString);
    monitorsizer->Add(rem_monitor, 0, wxALL | wxALIGN_CENTER_VERTICAL, 10);
    topsizer->Add(monitorsizer, 0, wxALIGN_CENTER_HORIZONTAL | wxALL, 10);

    wxScrolledWindow* disp_scroll = new wxScrolledWindow(this,
                                                         -1,
                                                         wxDefaultPosition,
                                                         wxDefaultSize,
         wxSUNKEN_BORDER | wxHSCROLL | wxVSCROLL | wxFULL_REPAINT_ON_RESIZE);

    toptracesizer = new wxBoxSizer(wxVERTICAL);

    disp_scroll->SetSizer(toptracesizer);
    disp_scroll->SetScrollRate(10, 10);
    disp_scroll->SetAutoLayout(true);

    for(int i = 0; i<10; i++)
      {
        vtracesizers.push_back(new wxBoxSizer(wxHORIZONTAL));
        canvases.push_back(new MyGLCanvas(disp_scroll,
                                          wxID_ANY,
                                          monitor_mod,
                                          names_mod,
                                          wxPoint(-1,-1),
                                          wxSize(-1,40)));
        tracesizer = wxT("m");
        tracename = tracesizer << i;
        tracelabels.push_back(new wxStaticText(disp_scroll, wxID_ANY, tracename));
        vtracesizers[i]->Add(tracelabels[i],
                             0,
                             wxALL | wxALIGN_LEFT | wxALIGN_CENTER_VERTICAL,
                             10);
        vtracesizers[i]->Add(canvases[i], 1, wxALL | wxEXPAND, 10);
        toptracesizer->Add(vtracesizers[i], 0, wxEXPAND | wxALL, 10);
        toptracesizer->Hide(vtracesizers[i], true);
        toptracesizer->Layout();
      }
```

```
267
    topsizer->Add(disp_scroll, 1, wxEXPAND | wxALL, 10);

    wxBoxSizer *termwinsizer = new wxBoxSizer(wxVERTICAL);
    termwinsizer->Add(new wxStaticText(this, wxID_ANY, wxT("Messages:")),
272                   0,
                      wxLEFT | wxALIGN_LEFT | wxALIGN_CENTER_VERTICAL,
                      10);
    textout = new wxTextCtrl(this,
                         wxID_ANY,
277                      wxT(""),
                         wxDefaultPosition,
                         wxDefaultSize,
                         wxTE_MULTILINE | wxHSCROLL);
    //textout->SetDefaultStyle(wxTextAttr(wxFONTFAMILY_TELETYPE));
282 wxFont * monofont = new wxFont(10,
                                  wxFONTFAMILY_TELETYPE,
                                  wxFONTSTYLE_NORMAL,
                                  wxFONTWEIGHT_NORMAL,
                                  false,
287                               wxEmptyString,
                                  wxFONTENCODING_DEFAULT);
    textout->SetFont(*monofont);
    wxStreamToTextRedirector redirect(textout);
    termwinsizer->Add(textout, 1, wxEXPAND | wxALL, 10);
292 topsizer->Add(termwinsizer, 1, wxEXPAND | wxALL, 10);

    SetSizeHints(400, 400);
    SetSizer(topsizer);

297 }

  void MyFrame::OnExit(wxCommandEvent &event)
  {
    Close(true);
302 }

  void MyFrame::OnAbout(wxCommandEvent &event)
  {
    wxMessageDialog about(this, wxT("LogicSim\n\
307 ␣␣␣␣By␣James␣Glanville,␣George␣Ayris␣and␣Andy␣Holt"),
                          wxT("About␣LogicSim"),
                          wxICON_INFORMATION | wxOK);
    about.ShowModal();
  }
312
  /*
  void MyFrame::OnFileButton(wxCommandEvent &event)
  {

317 wxString filetoopenpath;
    wxFileDialog *loadNewFile = new wxFileDialog(this,
                                                 wxT("Choose file to open"),
                                                 wxEmptyString,
                                                 wxEmptyString,
322                     _("DEF files (*.def)|*.def|TXT file (*.txt)|*.txt"),
                                                 wxFD_OPEN,
                                                 wxDefaultPosition);
    if (loadNewFile->ShowModal() == wxID_OK) // if click OPEN rather
                                             // than CANCEL
327   {
        filetoopenpath = loadNewFile->GetPath();
      }

    loadNewFile->Destroy();
332
    filetoopen = (string)filetoopenpath.mb_str();
  }
  */
```

```
337 void MyFrame::OnLoadButton(wxCommandEvent &event)
    {
      wxStreamToTextRedirector redirect(textout);
      // get switches and put in the switches dialog box
      int i = 0;
342   while (dmz->getswitch(i).compare("") != 0)
        {
          //wxStreamToTextRedirector redirect(text);
          //cout << dmz->getswitch(i) << " is a switch." << endl;
          if(switch_list->FindString(wxString::FromAscii(dmz->getswitch(i).c_str()))
347          == -1)
            {
              switch_list->Append(wxString::FromAscii(dmz->getswitch(i).c_str()));
            }
          i++;
352     }


      // get monitors and put them in the add monitors dialog box
      for (int i=0; i < mmz->moncount(); i++)
357     {
          if(add_monitor->FindString(wxString::FromAscii(mmz->getmonprettyname(i).c_str()))
             == -1)
            {
              // get name as string, convert to char* then to wxstring
362           add_monitor->Append(wxString::FromAscii(mmz->getmonprettyname(i).c_str()));
            }
        }
      cout << "Loaded devices from file." << endl;

367
    }

    void MyFrame::OnRunButton(wxCommandEvent &event)
    {
372   // run network from scratch for selected number of cycles.
      int n, ncycles;
      cyclescompleted = 0;
      mmz->resetmonitor();
      //wxStreamToTextRedirector redirect(textout);
377   //cout << "run network for " << spin_cycles->GetValue() << " cycles." << endl;
      runnetwork(spin_cycles->GetValue());
    }

    void MyFrame::OnContButton(wxCommandEvent &event)
382 {
      // continue simulation - same as OnRunButton but w/o resetting
      // cyclescompleted
      int n, ncycles;
      //wxStreamToTextRedirector redirect(textout);
387   //cout << "network has run " << cyclescompleted << " cycles." << endl;
      //cout << "and will continue with " << spin_cycles->GetValue() << "
      //more cycles." << endl;
      runnetwork(spin_cycles->GetValue());
    }
392
    void MyFrame::runnetwork(int ncycles)
    {
      // Function to run the network, derived from corresponding function
      // in userint.cc
397   wxStreamToTextRedirector redirect(textout);
      bool ok = true;
      int n = ncycles;

      //cout << "in run network function." << endl;
402
      if (cyclescompleted + n >= 350)
        {
          cout << "Error: too many cycles." << endl;
          cout << "All good things must come to an end." << endl;
```

```
407       cout << "(Maximum␣is␣350.)" <<endl;
          return;
        }

      while ((n > 0) && ok) {
412     dmz->executedevices (ok);
        if (ok)
          {
            n--;
            mmz->recordsignals ();
417       }
        else
          cout << "Error:␣network␣is␣oscillating" << endl;
      }

422   //cout << "finished while" << endl;
      if (ok)
        {
          cyclescompleted = cyclescompleted + ncycles;
        }
427   else
        {
          cyclescompleted = 0;
        }
      cout << "network␣has␣run␣" << cyclescompleted << "␣cycles." << endl;
432
      int mon;

      // draw each trace
      for (int i=0; toptracesizer->IsShown(vtracesizers[i]); i++)
437     {
          mon = 0;
          while (mmz->getmonprettyname(mon)
                 != (string)tracelabels[i]->GetLabel().mb_str() && mon < 5)
            {
442           mon++;
            }
          canvases[i]->Render(mon, cyclescompleted);
        }
    }
447
    void MyFrame::OnSwitchSelect(wxCommandEvent& event)
    {

    }
452
    void MyFrame::OnSwitchOption(wxCommandEvent& event)
    {
      wxStreamToTextRedirector redirect(textout);

457   bool ok;
      if (switch_option->GetValue() == wxT("HIGH"))
        {
          dmz->setswitch(nmz->lookup((string)switch_list->GetValue().mb_str()),
                         high, ok);
462       if (ok)
            cout << "switch␣\"" << switch_list->GetValue().ToAscii()
                 << "\"␣set␣to␣HIGH" << endl;
          else
            cout << "Error:␣switch␣\"" << switch_list->GetValue().ToAscii()
467              << "\"␣not␣found." << endl;
        }
      else if (switch_option->GetValue() == wxT("LOW"))
        {
          dmz->setswitch(nmz->lookup((string)switch_list->GetValue().mb_str()),
472                      low, ok);
          if (ok)
            cout << "switch␣\"" << switch_list->GetValue().ToAscii()
                 << "\"␣set␣to␣LOW" << endl;
          else
```

```
477         cout << "Error: switch \"" << switch_list->GetValue().ToAscii()
                 << "\" not found." << endl;
      }
    }

482 void MyFrame::OnAddMonitor(wxCommandEvent& event)
    {
      rem_monitor->Append(add_monitor->GetValue());
      //cout << "Add trace at monitor point: " <<
      //add_monitor->GetValue().ToAscii() << endl;
487
      int i=0;
      while (toptracesizer->IsShown(vtracesizers[i]))
        {
          i++;
492     }
      tracelabels[i]->SetLabel(add_monitor->GetValue());
      vtracesizers[i]->Layout();
      toptracesizer->Show(vtracesizers[i], true, true);
      toptracesizer->Layout();
497   topsizer->Layout();

      add_monitor->Delete(add_monitor->FindString(add_monitor->GetValue()));
      add_monitor->SetValue(wxT(""));
    }
502
    void MyFrame::OnRemMonitor(wxCommandEvent& event)
    {
      add_monitor->Append(rem_monitor->GetValue());

507   int i = 0;
      while (tracelabels[i]->GetLabel() != rem_monitor->GetValue())
        {
          i++;
        }
512
      // bump down all traces above to fill empty slot
      // first, find how many are shown below:
      int j = i+1;
      while(toptracesizer->IsShown(vtracesizers[j]) && j<10)
517     {
          j++;
        }
      j = j-i-1;                      // there are now j below emptied trace
      for(int k=0; k<j; k++)
522     {
          tracelabels[i+k]->SetLabel(tracelabels[i+k+1]->GetLabel());
          vtracesizers[i+k]->Layout();
        }
      tracelabels[i+j]->SetLabel(wxT(""));
527   vtracesizers[i+j]->Layout();
      toptracesizer->Hide(vtracesizers[i+j], true);
      toptracesizer->Layout();
      topsizer->Layout();

532   /*
      tracelabels[i]->SetLabel(wxT(""));
      vtracesizers[i]->Layout();
      toptracesizer->Hide(vtracesizers[i], true);
      toptracesizer->Layout();
537   topsizer->Layout();
      */

      rem_monitor->Delete(rem_monitor->FindString(rem_monitor->GetValue()));
      rem_monitor->SetValue(wxT(""));
542   runnetwork(0);
      //cout << "Remove trace at monitor point: "
      //     << rem_monitor->GetValue().ToAscii()
      //     << endl;
    }
```

B Definition Files

B.1 4 Bit Shift Register

```
{
DEVICES
{
  D1 = DTYPE;
  D2 = DTYPE;
  D3 = DTYPE;
  D4 = DTYPE;
  INPUT = SW(1);
  RESET = SW(0);
  ZERO  = SW(0);
  CLK1  = CLK(1);
}

CONNECTIONS
{
  D3.DATA <= INPUT;
  D3.CLK  <= CLK1;
  D3.SET  <= ZERO; //This should always be 0 for shift register.
  D3.CLEAR<= RESET;

        D2.DATA <= D3.Q;
        D2.CLK  <= CLK1;
        D2.SET  <= ZERO; //This should always be 0 for shift register.
        D2.CLEAR<= RESET;

        D1.DATA <= D2.Q;
        D1.CLK  <= CLK1;
        D1.SET  <= ZERO; //This should always be 0 for shift register.
        D1.CLEAR<= RESET;

        D0.DATA <= D1.Q;
        D0.CLK  <= CLK1;
        D0.SET  <= ZERO; //This should always be 0 for shift register.
        D0.CLEAR<= RESET;
}

MONITORS
{
  Q3 <= D3.Q;
  Q2 <= D2.Q;
  Q1 <= D1.Q;
  Q0 <= D0.Q;
}
}
```

B.2 Logic Gate Array

```
{
DEVICES
{
  A = SW(0);
  B = SW(1);
  C = SW(0);
  D = SW(1);
  NAND1 = NAND(2);
  NAND2 = NAND(2);
  NAND3 = NAND(2);
  OR    = OR(2);
  X1    = XOR;
  NOR1  = NOR(2);
  NOR2  = NOR(2);
  NOR3  = NOR(2);
  AND   = AND(2);
  X2    = XOR;
}
```

```
CONNECTIONS
{
  OR.I1 <= A;
  OR.I2 <= B;
  NAND1.I1 <= A;
  NAND1.I2 <= A;
  NAND2.I1 <= B;
  NAND2.I2 <= B;
  NAND3.I1 <= NAND1;
  NAND3.I2 <= NAND2;
  X1.I1 <= OR;
  X1.I2 <= NAND3;

        AND.I1 <= C;
        AND.I2 <= D;
        NOR1.I1 <= C;
        NOR1.I2 <= C;
        NOR2.I1 <= D;
        NOR2.I2 <= D;
        NOR3.I1 <= NOR1;
        NOR3.I2 <= NOR2;
        X2.I1 <= AND;
        X2.I2 <= NOR3;
}

MONITORS
{
  ERROR1  <= X1;
  ERROR2  <= X2;
  ANDOUT  <= AND;
  NOROUT  <= NOR3;
  NANDOUT <= NAND3;
  OROUT <= OR;
}
}
```