Project GF2 Introduction

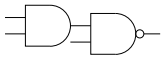
Andrew Gee

Department of Engineering, University of Cambridge

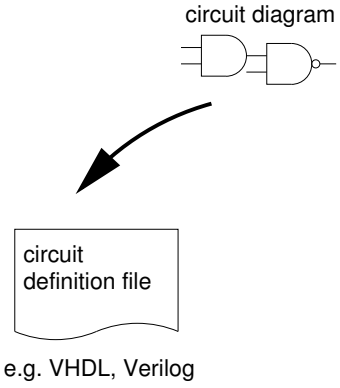
12 May 2022

Simulating digital circuits

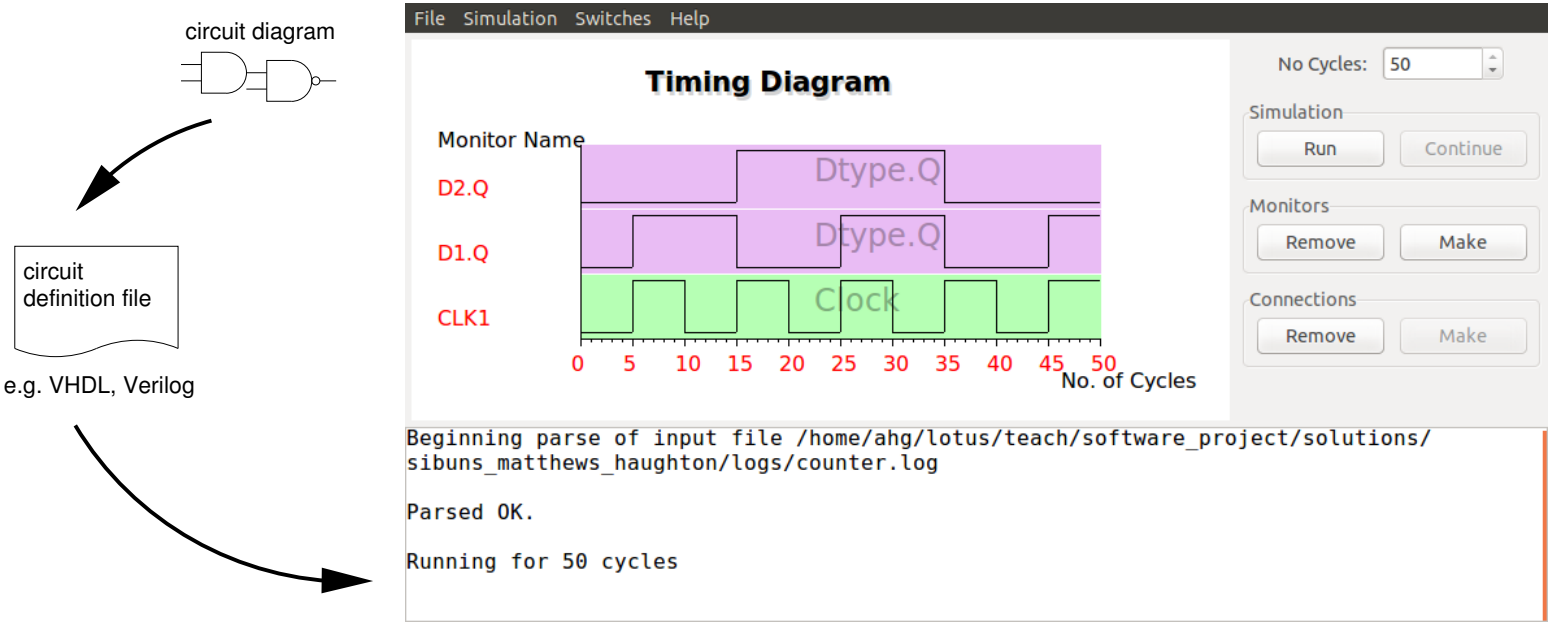
circuit diagram



Simulating digital circuits

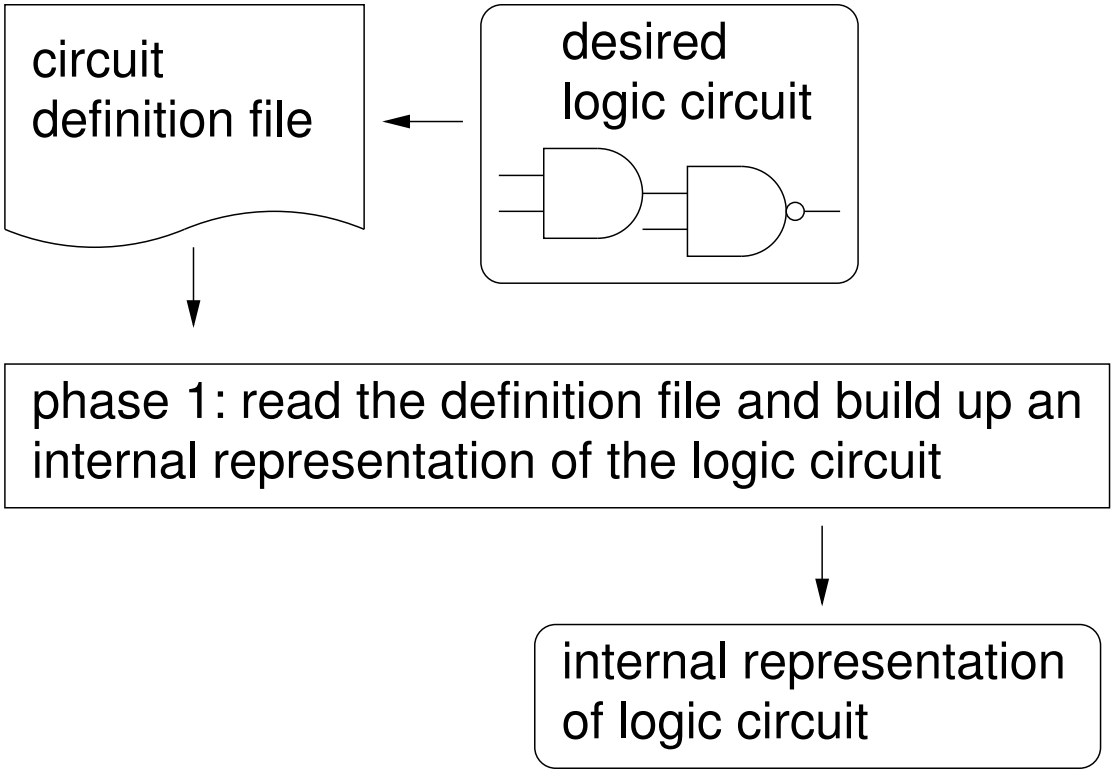


Simulating digital circuits



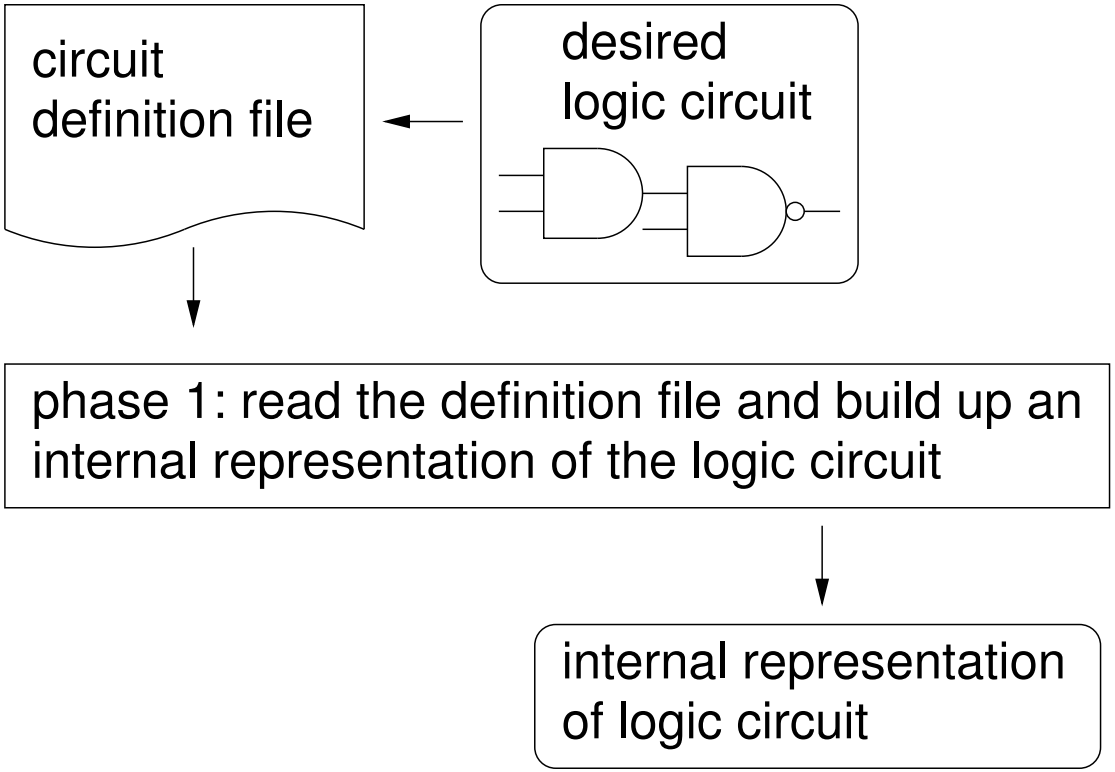
The GF2 logic simulator

- The simulator is already partially complete.



The GF2 logic simulator

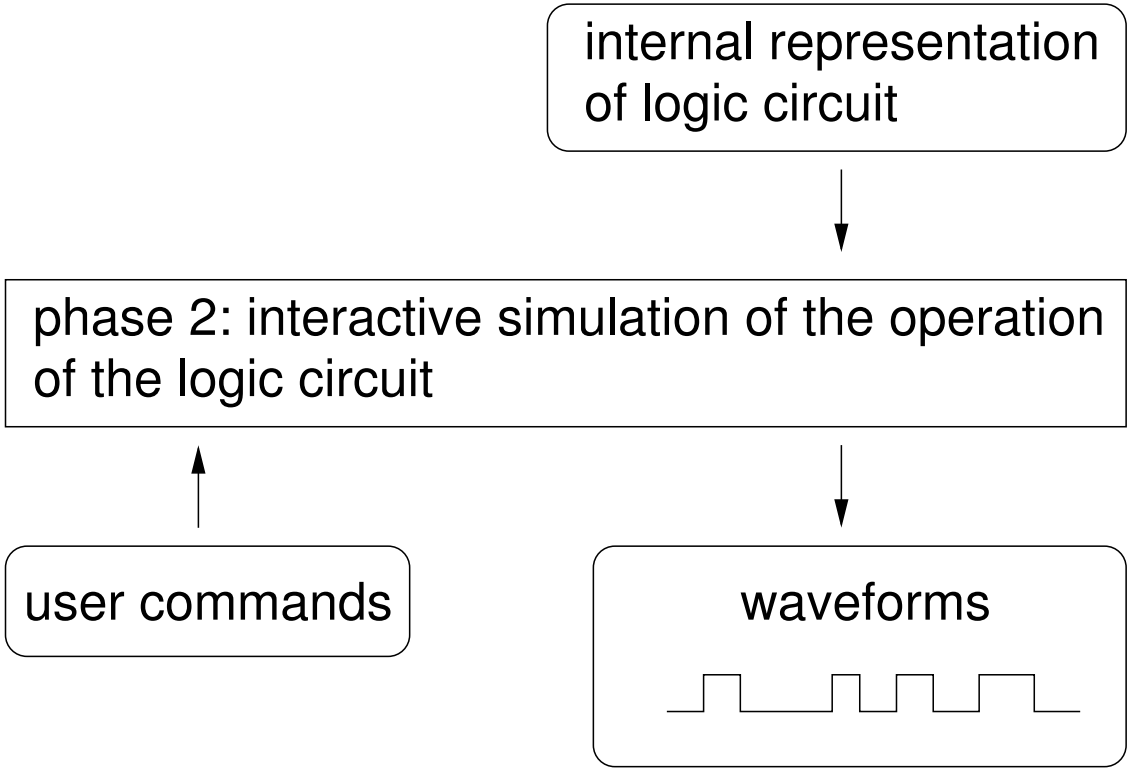
- The simulator is already partially complete.



- You are to produce the scanner and parser for phase 1 . . .

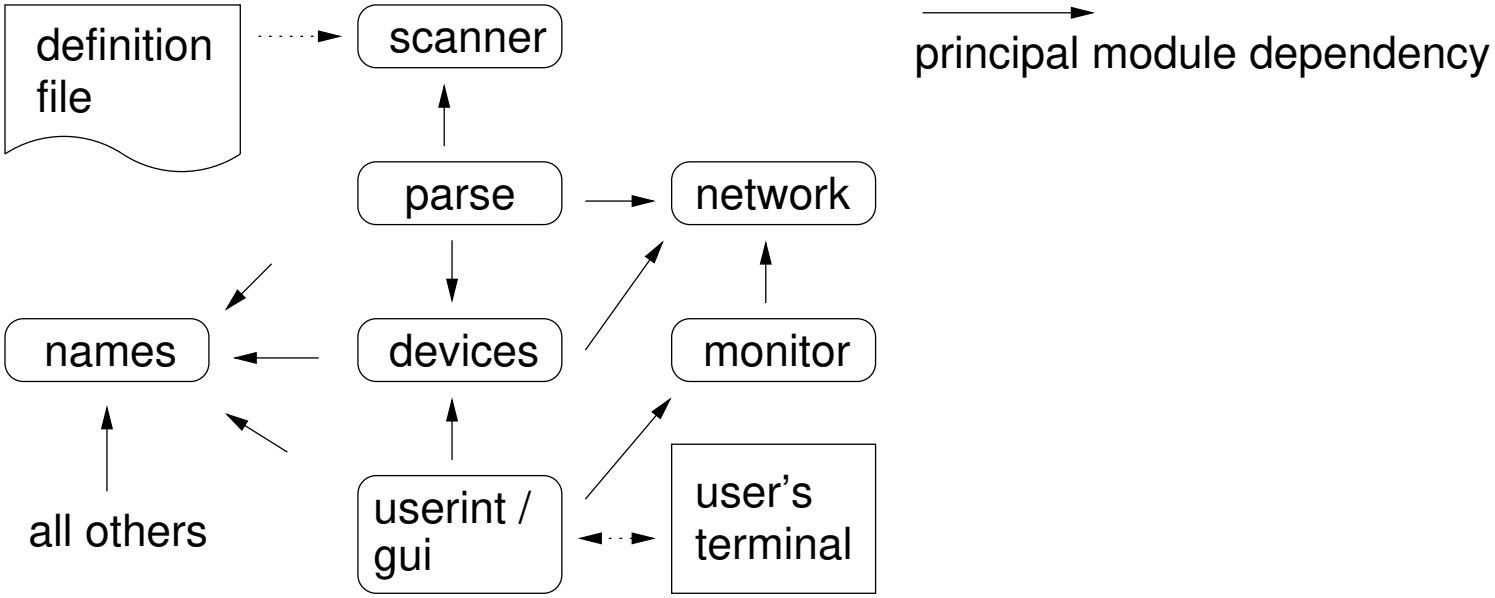
The GF2 logic simulator

- The simulator is already partially complete.

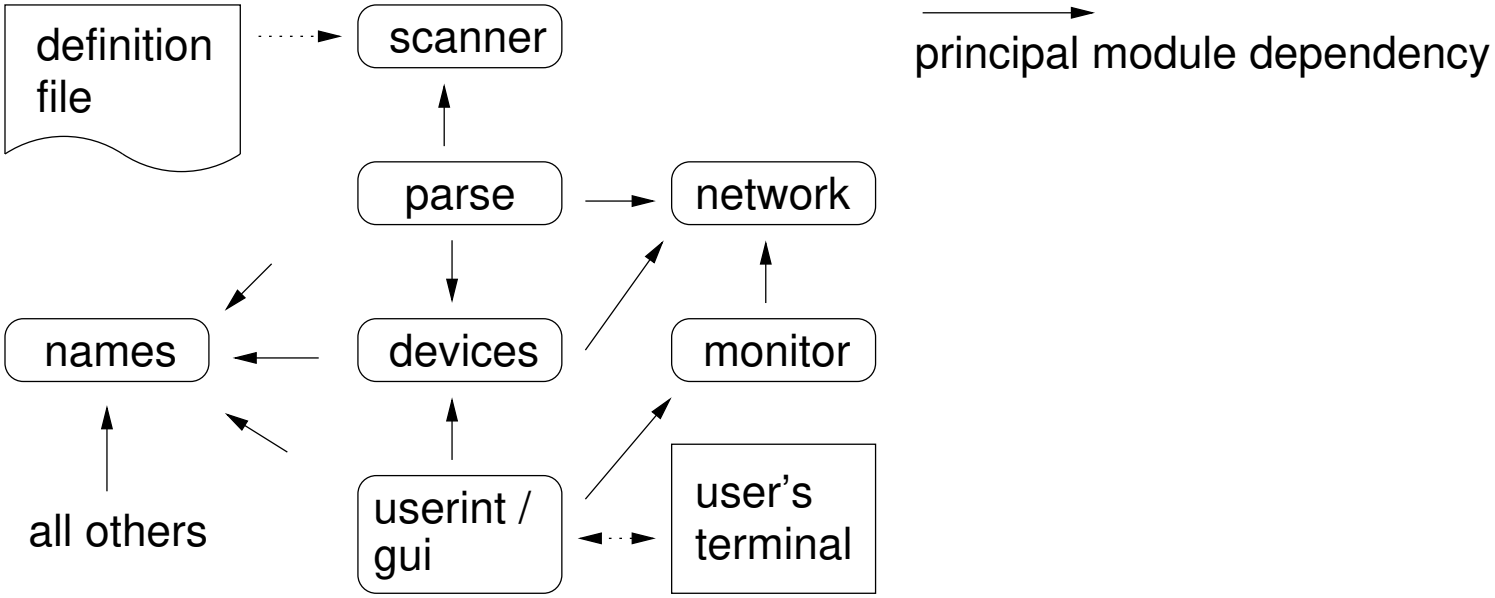


- . . . and the graphical user interface for phase 2.

Software structure

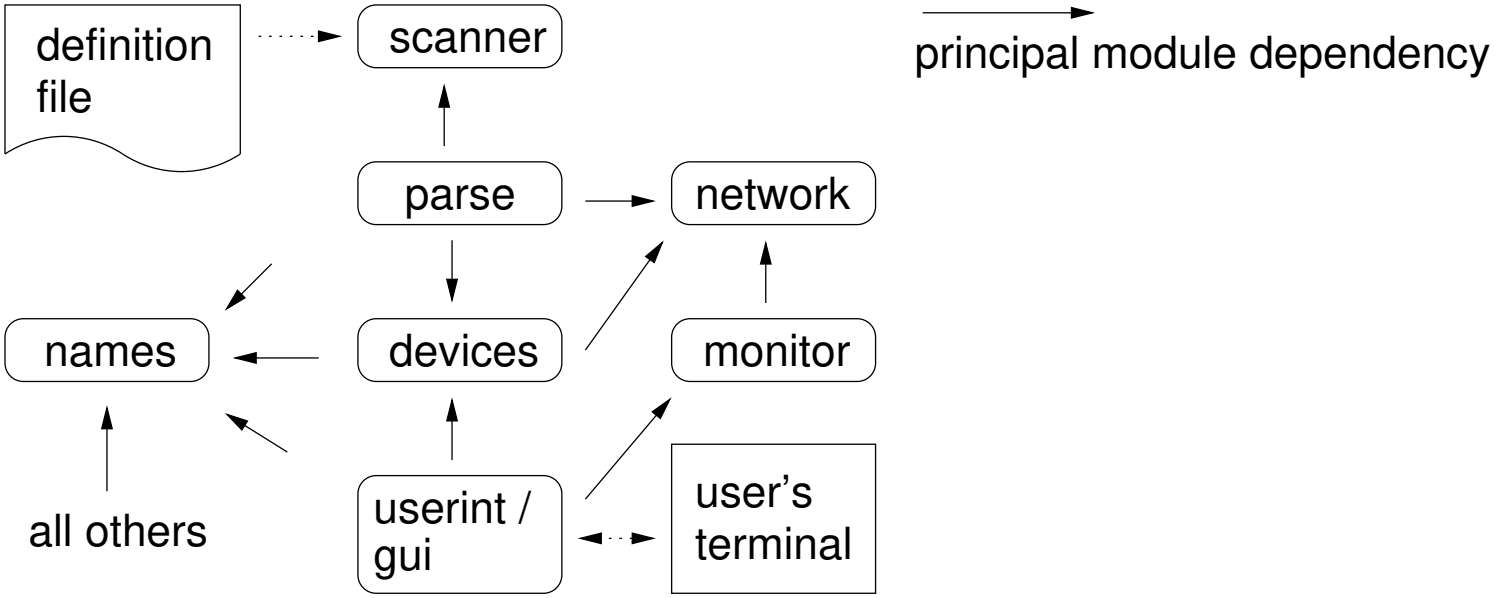


Software structure



- You will need to write most of the `names`, `scanner`, `parse` and `gui` modules.

Software structure



- You will need to write most of the `names`, `scanner`, `parse` and `gui` modules.
- Other modules are supplied and functional, but maybe not to your taste. This is normal when working in teams.

Aims and objectives

You will learn about:

Aims and objectives

You will learn about:

- The importance of specifying interfaces *first*

Aims and objectives

You will learn about:

- The importance of specifying interfaces *first*
- Modular decomposition, objects and classes

Aims and objectives

You will learn about:

- The importance of specifying interfaces *first*
- Modular decomposition, objects and classes
- Unit testing

Aims and objectives

You will learn about:

- The importance of specifying interfaces *first*
- Modular decomposition, objects and classes
- Unit testing
- Formal language theory, scanning and parsing

Aims and objectives

You will learn about:

- The importance of specifying interfaces *first*
- Modular decomposition, objects and classes
- Unit testing
- Formal language theory, scanning and parsing
- Graphical user interface programming with events and event handlers

Aims and objectives

You will learn about:

- The importance of specifying interfaces *first*
- Modular decomposition, objects and classes
- Unit testing
- Formal language theory, scanning and parsing
- Graphical user interface programming with events and event handlers
- Working in teams

Aims and objectives

You will learn about:

- The importance of specifying interfaces *first*
- Modular decomposition, objects and classes
- Unit testing
- Formal language theory, scanning and parsing
- Graphical user interface programming with events and event handlers
- Working in teams
- Collaborative coding and version control

Aims and objectives

You will learn about:

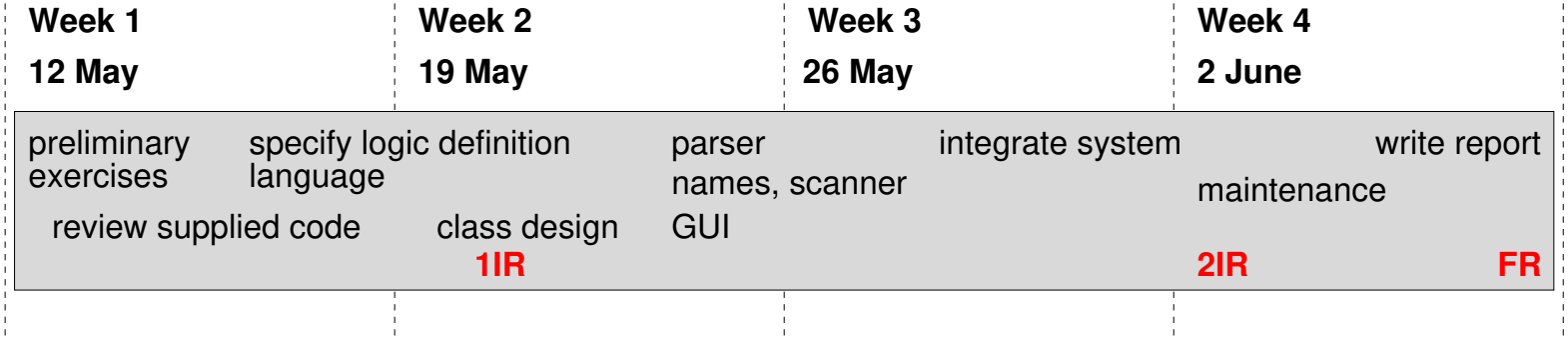
- The importance of specifying interfaces *first*
- Modular decomposition, objects and classes
- Unit testing
- Formal language theory, scanning and parsing
- Graphical user interface programming with events and event handlers
- Working in teams
- Collaborative coding and version control
- Documentation and user guides

Aims and objectives

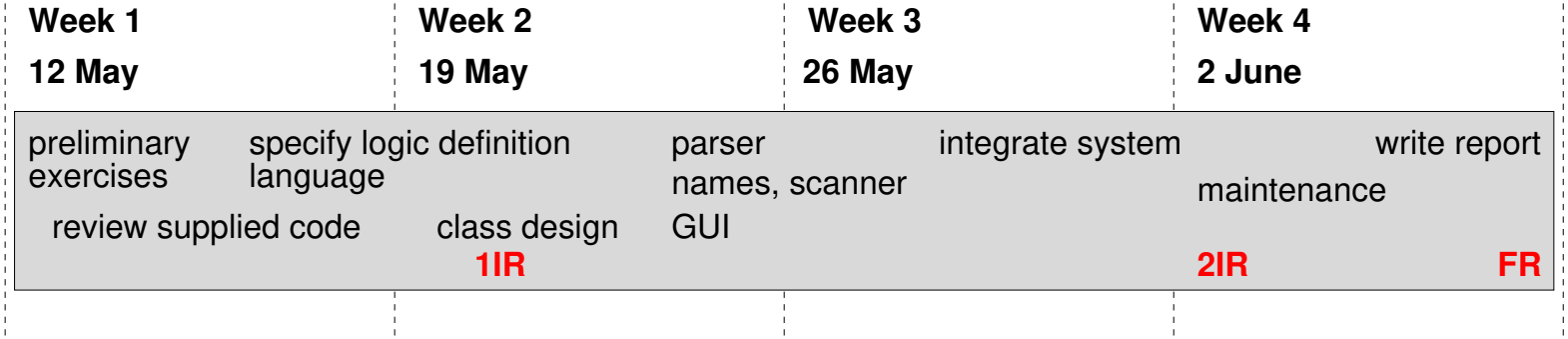
You will learn about:

- The importance of specifying interfaces *first*
- Modular decomposition, objects and classes
- Unit testing
- Formal language theory, scanning and parsing
- Graphical user interface programming with events and event handlers
- Working in teams
- Collaborative coding and version control
- Documentation and user guides
- Working to deadlines

Project schedule and deadlines

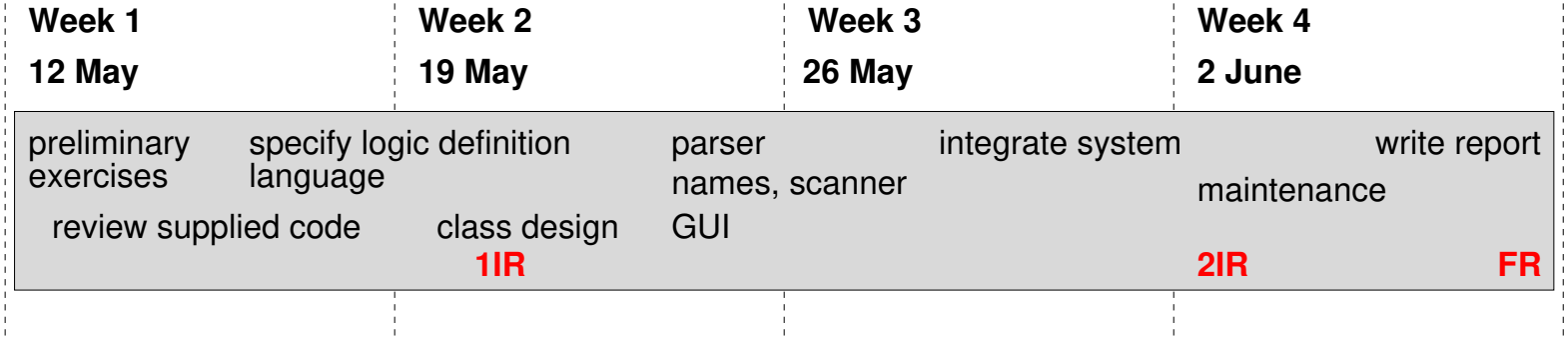


Project schedule and deadlines



- Compulsory project sessions are Thursdays 11am–1pm and Mondays 9am–11am. Monday afternoons, 2pm–4pm, are encouraged but optional.

Project schedule and deadlines



- Compulsory project sessions are Thursdays 11am–1pm and Mondays 9am–11am. Monday afternoons, 2pm–4pm, are encouraged but optional.
- First interim report due 4pm Saturday 21 May.
- Second interim report due 11am Thursday 2 June.
- Final report due 4pm Thursday 9 June.

Marking scheme

first interim report	15 group marks
second interim report	15 marks (7 group, 8 solo)
final report	50 solo marks
missing project sessions per hour, or part thereof	–1 mark
handing in either interim report late (per day, or part thereof)	–3 marks

Marking scheme

first interim report	15 group marks
second interim report	15 marks (7 group, 8 solo)
final report	50 solo marks
missing project sessions per hour, or part thereof	–1 mark
handing in either interim report late (per day, or part thereof)	–3 marks

- It is up to you to make sure you are marked in by a demonstrator within 5 minutes of the start of each session.
- You may not leave before the end of the session.

Marking scheme

first interim report	15 group marks
second interim report	15 marks (7 group, 8 solo)
final report	50 solo marks
missing project sessions per hour, or part thereof	–1 mark
handing in either interim report late (per day, or part thereof)	–3 marks

- It is up to you to make sure you are marked in by a demonstrator within 5 minutes of the start of each session.
- You may not leave before the end of the session.
- No deadline extension is possible for the final report. If you hand it in late you may get zero marks for it.

Moodle

Part IIA Project: GF2: Software

- Announcements
- Project survey
- 2019 teams

Documentation

- Introductory lecture (set P2)
- Main project handout (set P2)
- Source code for the preliminary exercises
- Client's maintenance requirements (set P2)

Restricted Available from **30 May 2019, 11:00 AM**

Generic web links

Use these as starting points for further reading.

- Unix from the command line
- A simple guide to git
- Tim's notes on git
- Extended Backus-Naur Form
- Recursive descent parsing of LL(1) grammars
- Tim's notes on debugging
- wxWidgets tutorials
- Tim's notes on wxWidgets
- OpenGL programming guide

Python web links

- File I/O (preliminary exercises 2, 3 and 4)
- Exception handling (preliminary exercise 2)
- Loops (preliminary exercise 3)
- The print statement (preliminary exercise 3)
- String processing (preliminary exercises 4, 5 and 6)
- Lists (preliminary exercises 5, 6 and 7)
- Classes (preliminary exercise 7)
- Pytest tutorial (preliminary exercise 8)
- PEP 8 style guide (preliminary exercise 9)
- PEP 257 docstring conventions (preliminary exercise 9)
- wxPython reference manual
- Full pytest documentation

Getting help

Demonstrators are available during the timetabled sessions, at other times please post a query to the forum. In the spirit of constructive collaboration, please feel free to answer queries from other students if you are able to.

- GF2 discussion forum

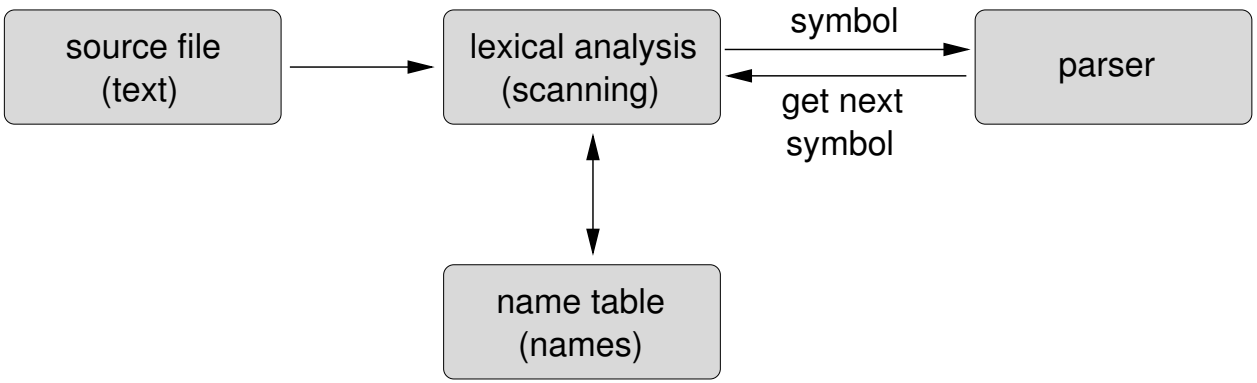
Submit your reports here

- First interim report (set P2)
- Second interim report (set P2)
- Final report (set P2)

- Documentation, resources, calendar, forum, report submission, report feedback.

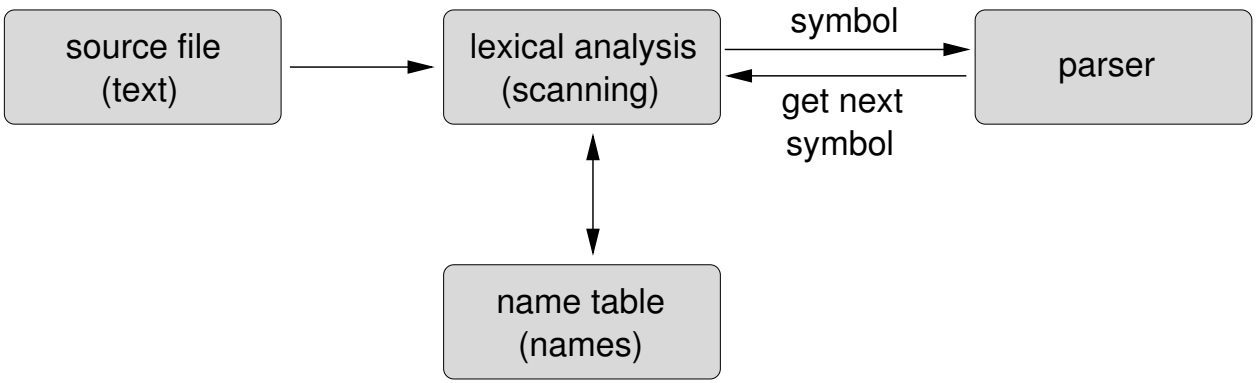
Scanning and parsing

- There are two stages to “reading in” the logic definition file: scanning (lexical analysis) and parsing.



Scanning and parsing

- There are two stages to “reading in” the logic definition file: scanning (lexical analysis) and parsing.



- In the scanning stage, the text file is translated into a set of symbols.

A lexical analysis song

A is one plus two times three

		(scanner output)	
		symbol	**type**
A:	a name I call my var	A	name
is:	assignment operator	=	operator
one:	a number, the number 1	1	number
plus:	addition operator	+	operator
two:	another number here	2	number
times:	a multiplying op	$\times$	operator
three:	guess what, the number 3	3	number

which will bring us back to A . . . A . . . A . . . A . . .

Parsing

- The parser checks the symbols to see whether they conform to the logic description language's grammar.

Parsing

- The parser checks the symbols to see whether they conform to the logic description language's grammar.
- If they do not, we have a *syntax error*.

Parsing

- The parser checks the symbols to see whether they conform to the logic description language's grammar.
- If they do not, we have a *syntax error*.
- If they do, but the symbols make no sense (e.g. monitoring a nonexistent device), we have a *semantic error*.

Parsing

- The parser checks the symbols to see whether they conform to the logic description language's grammar.
- If they do not, we have a *syntax error*.
- If they do, but the symbols make no sense (e.g. monitoring a nonexistent device), we have a *semantic error*.
- If there are no errors, the parser calls functions in other modules to construct the circuit.

Parsing

- The parser checks the symbols to see whether they conform to the logic description language's grammar.
- If they do not, we have a *syntax error*.
- If they do, but the symbols make no sense (e.g. monitoring a nonexistent device), we have a *semantic error*.
- If there are no errors, the parser calls functions in other modules to construct the circuit.
- You need to design the logic description language.

Parsing

- The parser checks the symbols to see whether they conform to the logic description language's grammar.
- If they do not, we have a *syntax error*.
- If they do, but the symbols make no sense (e.g. monitoring a nonexistent device), we have a *semantic error*.
- If there are no errors, the parser calls functions in other modules to construct the circuit.
- You need to design the logic description language.
- You will use EBNF syntax rules and informal semantics.

EBNF grammars

- Each EBNF syntactic rule has a LHS and a RHS.

$$\underbrace{\begin{array}{l}\text{specfile}\\ \text{block}\\ \text{connections}\\ \text{devices}\end{array}}_{\text{LHS}} \quad \begin{array}{c}=\\ =\\ =\\ =\end{array} \quad \underbrace{\begin{array}{l}\text{block , \{ block \} ;}\\ \text{connections | devices ;}\\ \text{"CONNECT" , con , \{ con \} ;}\\ \text{"DEVICE" , dev , \{ dev \} ;}\end{array}}_{\text{RHS}}$$

EBNF grammars

- Each EBNF syntactic rule has a LHS and a RHS.

$$\underbrace{\begin{array}{l}\text{specfile}\\ \text{block}\\ \text{connections}\\ \text{devices}\end{array}}_{\text{LHS}} \quad \begin{array}{c}=\\ =\\ =\\ =\end{array} \quad \underbrace{\begin{array}{l}\text{block , \{ block \} ;}\\ \text{connections | devices ;}\\ \text{"CONNECT" , con , \{ con \} ;}\\ \text{"DEVICE" , dev , \{ dev \} ;}\end{array}}_{\text{RHS}}$$

- Your grammar should be left to right with one lookahead symbol. LL(1) grammars are relatively easy to parse.

EBNF grammars

- Each EBNF syntactic rule has a LHS and a RHS.

```
specfile      =   block , { block } ;
block         =   connections | devices ;
connections   =   "CONNECT" , con , { con } ;
devices       =   "DEVICE" , dev , { dev } ;
```

LHS RHS

- Your grammar should be left to right with one lookahead symbol. LL(1) grammars are relatively easy to parse.
- With LL(1), whenever there is a choice on the RHS, it can be resolved by looking at the next symbol from the scanner.

Language design criteria

Ease of use: choice of keywords, not too concise or wordy.

Language design criteria

Ease of use: choice of keywords, not too concise or wordy.

Robustness to errors: for example, consider the grammar

```
connections  =  "CONNECT" , con , { con } ;
con          =  signal , "->" , signal ;
signal       =  devicename , [ "." , pinname ] ;
```

Language design criteria

Ease of use: choice of keywords, not too concise or wordy.

Robustness to errors: for example, consider the grammar

```
connections  =  "CONNECT" , con , { con } ;
con          =  signal , "->" , signal ;
signal       =  devicename , [ "." , pinname ] ;
```

and the alternative

```
con  =  signal , "->" , signal , ";" ;
```

Language design criteria

Ease of use: choice of keywords, not too concise or wordy.

Robustness to errors: for example, consider the grammar

```
connections  =  "CONNECT" , con , { con } ;
con          =  signal , "->" , signal ;
signal       =  devicename , [ "." , pinname ] ;
```

and the alternative

```
con  =  signal , "->" , signal , ";" ;
```

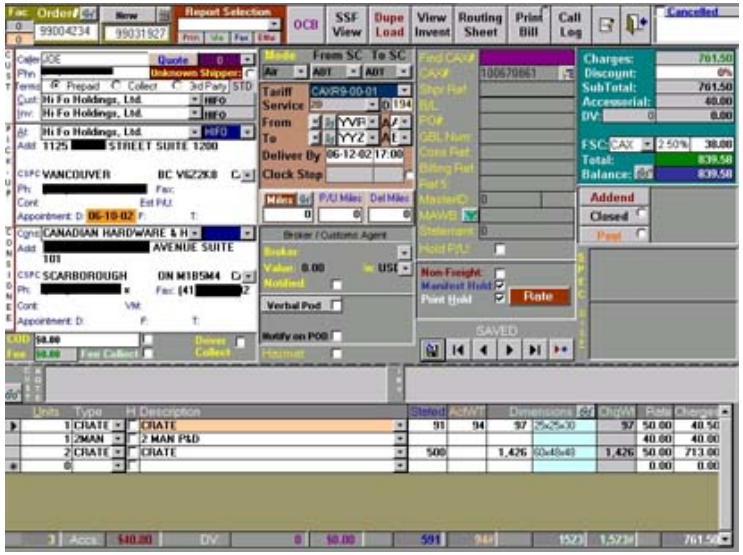
The redundant semicolon delimiter allows the parser to resume from a known state in the event of an error.

GUI design

- You will need to design, and *then* implement, a good GUI.

GUI design

- You will need to design, and *then* implement, a good GUI.



bad

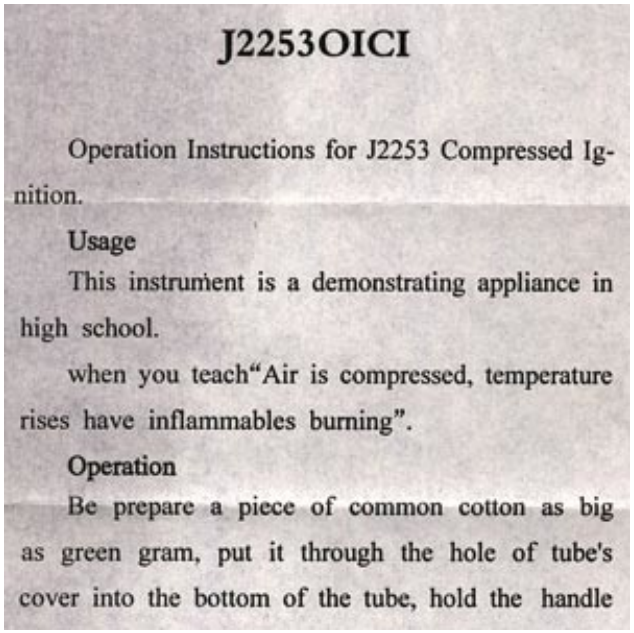
good

Documentation design

- You will also need to produce some good documentation.

Documentation design

- You will also need to produce some good documentation.



J2253OICI

Operation Instructions for J2253 Compressed Ignition.

Usage

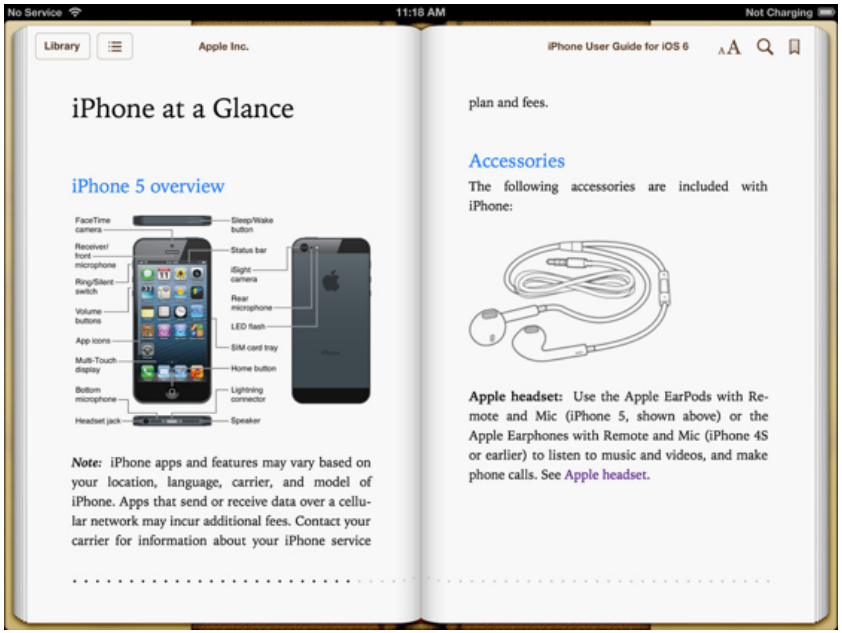
This instrument is a demonstrating appliance in high school.

when you teach"Air is compressed, temperature rises have inflammables burning".

Operation

Be prepare a piece of common cotton as big as green gram, put it through the hole of tube's cover into the bottom of the tube, hold the handle

bad



No Service 11:18 AM Not Charging

Library Apple Inc. iPhone User Guide for iOS 6

iPhone at a Glance

iPhone 5 overview

FaceTime camera, Receiver/front microphone, Ring/Silent switch, Volume buttons, App icons, Multi-Touch display, Bottom microphone, Headset jack, Sleep/Wake button, Status bar, iSight camera, Rear microphone, LED flash, SIM card tray, Home button, Lightning connector, Speaker

Note: iPhone apps and features may vary based on your location, language, carrier, and model of iPhone. Apps that send or receive data over a cellular network may incur additional fees. Contact your carrier for information about your iPhone service plan and fees.

Accessories

The following accessories are included with iPhone:

Apple headset: Use the Apple EarPods with Remote and Mic (iPhone 5, shown above) or the Apple Earphones with Remote and Mic (iPhone 4S or earlier) to listen to music and videos, and make phone calls. See Apple headset.

good