1 Introduction

The report focuses on the planning of the whole project in terms of task assignments and time planning. The project is divided into a number of tasks including design and implementation, for which the time lines are set in order to allow time for integration. Additionally the report addresses the design of syntax grammar in EBNF language, which will be adhered to throughout the project. It also describes the identification and handling for both syntax and semantic errors. Two examples of the definition files are appended.

2 Teamwork planning

2.1 Interface design for scanner and GUI

Designed by the whole team, before Tuesday 23 May.

2.2 *Names* class implementation

Implementation finished by Zhengyang before Saturday 27 May.
Testing finished by Shaowu before Tuesday 30 May.

2.3 *Scanner* class implementation

Implementation finished by Shaowu before Saturday 27 May.
Testing finished by Zhiwei before Tuesday 30 May.

2.4 *Parser* class implementation

Implementation finished by Zhiwei and Zhengyang before Saturday 27 May.
Testing finished by Shaowu before Tuesday 30 May.

2.5 *GUI* class implementation

Implementation finished by Zhiwei and Shaowu before Saturday 27 May.
Testing finished by Zhengyang before Tuesday 30 May.

2.6 System integration

Integration and testing by the whole team, before the deadline of the second interim report (Friday 2 June).

2.7 System maintenance

Modification and testing by the whole team, before the deadline of the final report (Thursday 8 June).

3 Syntax in EBNF language

```
file =   'DEVICES', {DEV, ','}, DEV, ';', 'CONNECT', {CON, ','}, CON, ';',
         'MONITOR', {MON, ','}, MON, ';';
DEV  =   'CLOCK', DEV_NAME, digit, {digit}                              |
         'SWITCH', DEV_NAME, ( 1 | 0 )                                  |
         'AND' | 'NAND' | 'OR' | 'NOR', DEV_NAME, [1], digit            |
         'D_TYPE', DEV_NAME                                             |
         'XOR', DEV_NAME;
DEV_NAME* =      digit | letter, {digit | letter | '_'};
CON      =       O_PIN, '=>', I_PIN;
O_PIN    =       DEV_NAME                          |
                 DEV_NAME, '.', 'Q' | 'QBAR';
I_PIN    =       DEV_NAME, '.', 'I', [1], digit              |
                 DEV_NAME, '.', 'DATA'|'CLK'|'SET'|'CLEAR';
MON      =       O_PIN | I_PIN;
```

*DEV_NAME = [0−9a−zA−Z_]+, DEV_NAME can be any combination of letter and number and '_', other than "DEVICES", "CONNECT", "MONITOR", "CLOCK", "SWITCH", "AND", "NAND', "OR", "NOR", "D_TYPE", "XOR"

4 Syntax error identification and handling

Error Name	Identification	Handling
Input Connected to Input	I_PIN =>I_PIN under CONNECT	Report, syntax error count + 1, continue parsing, cancel simulation
Input Connected to Output	I_PIN =>O_PIN under CONNECT	Report, syntax error count + 1, continue parsing, cancel simulation
Output Connected to Output	O_PIN =>O_PIN under CONNECT	Report, syntax error count + 1, continue parsing, cancel simulation
No Device Found	Empty DEVICE section	Report, syntax error count + 1, continue parsing, cancel simulation
No Connection Found	Empty CONNECT section	Report, syntax error count + 1, continue parsing, cancel simulation
No Monitor Found	Empty MONITOR section	Report, syntax error count + 1, continue parsing, cancel simulation
Undefined Object	Device used in CONNECT or MONITOR not found in the symbol table	Report, syntax error count + 1, continue parsing, cancel simulation
Unexpected Token	When the type of token does not match the expected type, e.g. CONNECT used as device name	Report, Report Token type expected, syntax error count + 1, continue parsing from next valid statement, cancel simulation

5 Semantics error identification and handling

Error Name	Identification	Handling
Floating Input	Input of a device is not connected to any output. Check after CONNECT section is parsed	Report, semantic error count +1,continue parsing, cancel simulation
Unused Output	Output of a device is not connected to any input or monitored. Check after MONITOR section is parsed.	Report, semantic warning count +1, continue parsing
Multiple Output connected to Single Input Pin	Check if an input pin is referred to twice under CONNECT section after CONNECT section is parsed	Report, semantic error count +1,continue parsing, cancel simulation
Invalid Clock Period (T ¡= 0)	Check if the period of a clock is positive	Report, semantic error count +1,continue parsing, cancel simulation
Invalid gate option	Check if the number of input to a gate is an integer between 1 and 16	Report, semantic error count +1,continue parsing, cancel simulation
Invalid Switch option	Check if a switch is set to any state other than 0 and 1	Report, semantic error count +1,continue parsing, cancel simulation
Name Conflict	Check if multiple device have the same name	Report, semantic warning count +1, continue parsing
Undefined Pin	Check if any pin in CONNECT and MONITOR is not defined for the specific device (e.g. I17 of a NAND gate)	Report, semantic error count +1,continue parsing, cancel simulation

6 Example definition files

Circuit 1 definition file.

```
// This is a single line comment.
DEVICES AND A 1, // an AND gate called 'A' with one input is declared.
        OR  B 1,
        XOR C,
        NAND D 3,
        SWITCH S1 1, // a switch S1 is declared and initialised to 1.
        SWITCH S2 1,
        SWITCH S3 1,
        SWITCH S4 1;
CONNECT S1 => A.I1, // switch S1 is connected to the first input of A.
        S2 => B.I1,
        S3 => C.I1,
        S4 => C.I2,
        A  => D.I1,
        B  => D.I2,
        C  => D.I3;
MONITOR D;
```

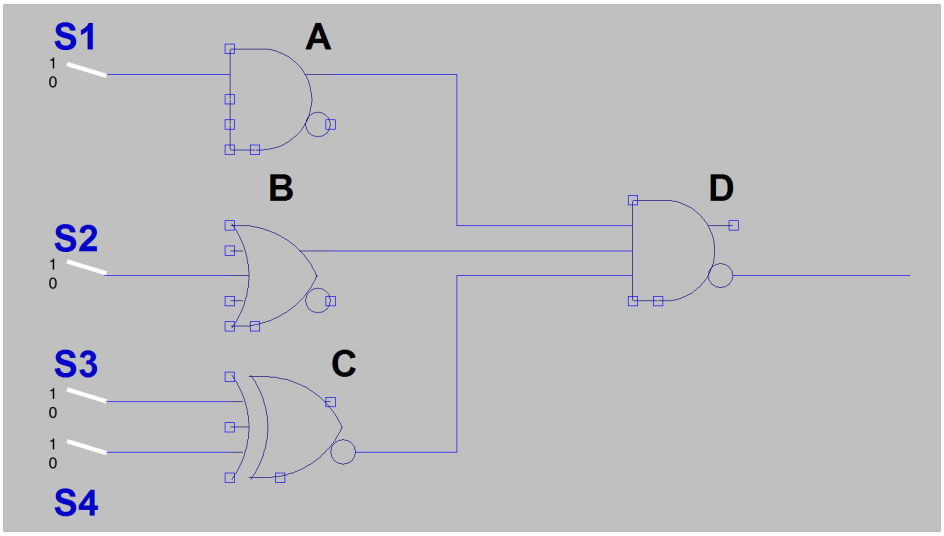

Figure 1: Circuit1

Circuit 2 definition file.

```
DEVICES CLOCK L 100, // clock output changes every 100 simulation cycles.
        SWITCH S1 1,
        SWITCH S2 0,
        SWITCH S3 0,
        DTYPE M,
        NOR A 2;
CONNECT S1 => M.SET,
        S2 => M.DATA,
        S3 => M.CLEAR,
        L  => M.CLK,
        M.Q => A.I1,
        M.QBAR => A.I2;
MONITOR A,
        QBAR;
/* A and QBAR are being monitored,
this is a multiline comment.*/
```

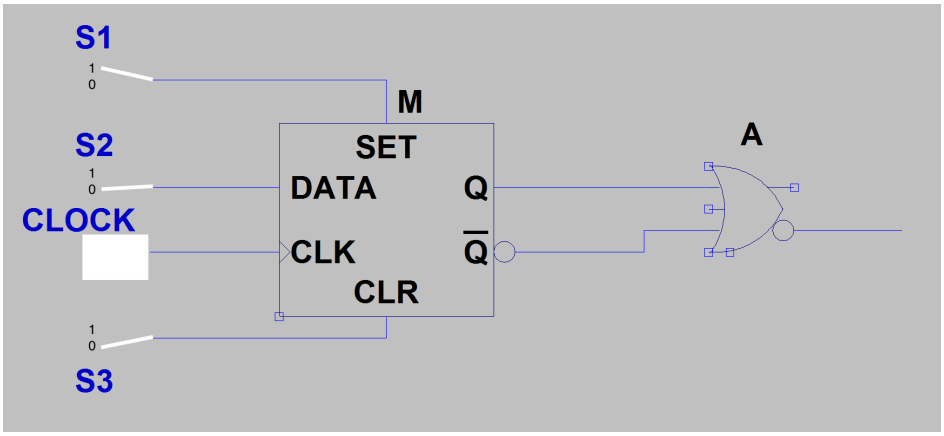

Figure 2: Circuit1