Appendix A: Example Definition Files

June 6, 2023

1. Example logic circuit 1

```
/* Example circuit comment*/
INIT;
D1 is DTYPE;
CK1 is CLOCK with_simulation_cycles 6;
AND1 is AND with 2 inputs;
NOR1 is NOR with 2 inputs;
SW1 is SWITCH initially_at 0;
SW2 is SWITCH initially_at 0;
SW3 is SWITCH initially_at 0;
CONNECT;
CK1 connect_to D1.CLK;
CK1 connect_to AND1.I2;
SW1 connect_to D1.SET;
SW2 connect_to D1.DATA;
SW3 connect_to D1.CLEAR;
D1.Q connect_to NOR1.I1;
D1.QBAR connect_to AND1.I1;
AND1 connect_to NOR1.I2;
MONITOR;
Initial_monitor_at D1.Q D1.QBAR NOR1;
```

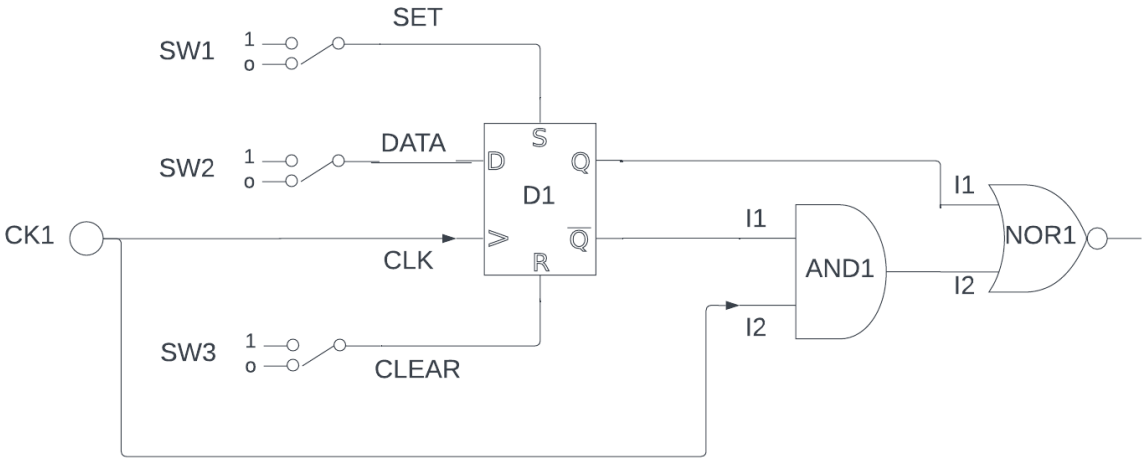


Figure 1: Example logic circuit 1

2. Example logic circuit 2

```
/* Example circuit comment*/
INIT;
NAND1 is NAND with 2 inputs;
OR1 is OR with 2 inputs;
XOR1 is XOR;
SW1 is SWITCH initially_at 0;
SW2 is SWITCH initially_at 0;
CONNECT;
SW1 connect_to NAND1.I1;
SW2 connect_to OR1.I2;
NAND1 connect_to OR1.I1;
OR1 connect_to NAND1.I2;
NAND1 connect_to XOR1.I1;
OR1 connect_to XOR1.I2;
MONITOR;
Initial_monitor_at NAND1 OR1 XOR1;
```

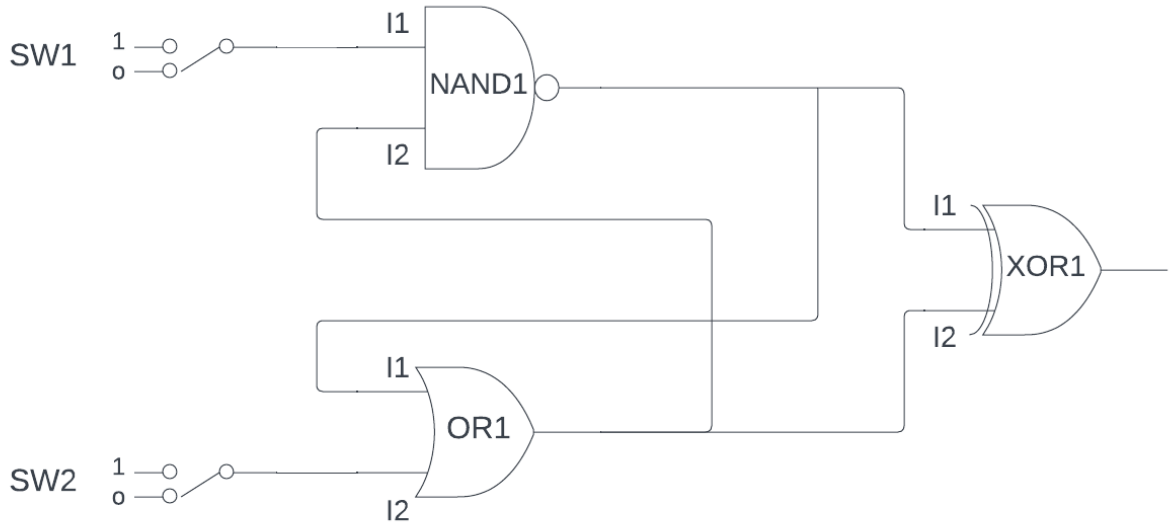


Figure 2: Example logic circuit 2

3. Example logic circuit 3

```
/* Example circuit comment*/
INIT;
D1 is DTYPE;
CK1 is CLOCK with_simulation_cycles 6;
SG1 is SIGGEN with "01010";
RC1 is RC with_simulation_cycles 5;
AND1 is AND with 2 inputs;
NOR1 is NOR with 2 inputs;
SW1 is SWITCH initially_at 0;
```

```
SW2 is SWITCH initially_at 0;
CONNECT;
CK1 connect_to D1.CLK;
SG1 connect_to AND1.I2;
RC1 connect_to D1.SET;
SW1 connect_to D1.DATA;
SW2 connect_to D1.CLEAR;
D1.Q connect_to NOR1.I1;
D1.QBAR connect_to AND1.I1;
AND1 connect_to NOR1.I2;
MONITOR;
Initial_monitor_at D1.Q RC1 AND1 NOR1;
```

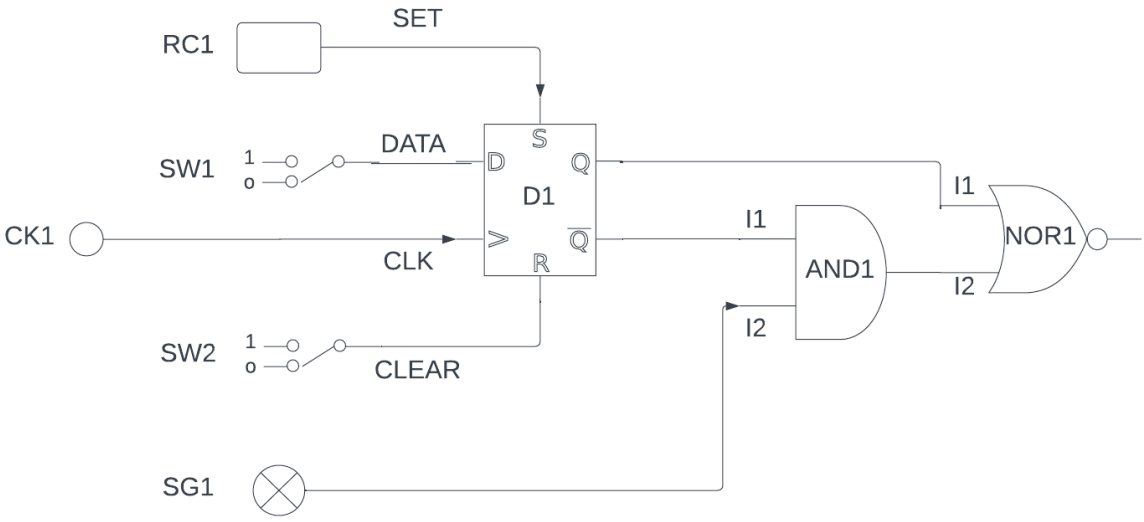


Figure 3: Example logic circuit 3

EBNF Syntax Handbook

Group 12
Amanda Ge, Chulabutra Chuenchoksan, Koko Ishida

June 2023

1 EBNF Syntax Rule

1. Comments

```
Comments = "/*", any comment, "*/";
```

2. Description file format

```
description = "INIT;", initialisation,{initialisation},
"CONNECT;", connection,{connection}, ["MONITOR;",initial monitor]
```

3. Initialisation of devices

```
initialise_gate = device_name,"is",gate_type,"with", {digit}, "inputs"|"input";
initialise_switch = device_name, "is", switch_type, "initially_at", "0"|"1";
initialise_xor = device_name, "is", xor_type;
initialise_dtype = device_name, "is", dtype_type;
initialise_clock = device_name, "is", clock_type,
                "with_simulation_cycles", digit,{digit};
initialise_rc = device_name, "is", rc_type,
                "with_simulation_cycles", digit,{digit};
initialise_siggen = device_name, "is", siggen_type, "with",
                siggen_waveform;
initialisation = initialise_gate|initialise_switch|
                 initialise_dtype|initialise_clock|
                 initialise_rc|initialise_siggen,ter;
```

4. Connections

```
connection = device_output, "connect_to", device_input,ter;
```

5. Monitoring

```
Monitor_point = device_output,{device_output};
initial_monitor = "Initial_monitor_at", "0"|Monitor_point,ter;
```

6. User-defined device names

```
        device_name = alphabet, {alphabet}, digit, {digit};
```

7. Definition of devices

```
        gate_type = "AND"|"NAND"|"OR"|"NOR";
        xor_type = "XOR";
        switch_type = "SWITCH";
        dtype_type = "DTYPE";
        clock_type = "CLOCK";
        rc_type = "RC";
        siggen_type = "SIGGEN";
```

8. SIGGEN waveform

```
        binary_wave = "0"|"1"
        siggen_waveform = "\", binary_wave, {binary_wave}, \""
```

9. Input/Output of devices

```
        gate_input = device_name, ".", "I", {digit};
        xor_input = device_name, ".", "I", "1"|"2";
        dtype_input = device_name, ".", "DATA"|"CLK"|"SET"|"CLEAR";
        device_input = gate_input | dtype_input | xor_input;

        gate_output = device_name;
        switch_output = device_name;
        clock_output = device_name;
        rc_output = device_name;
        siggen_output = device_name;
        xor_output = device_name;
        dtype_output = device_name, ".", "Q"|"QBAR";
        device_output = gate_output|switch_output|clock_output|
                        rc_output|siggen_output|dtype_output|xor_output;
```

10. Alphabets and digits

```
        alphabet = "A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"
                  |"J"|"K"|"L"|"M"|"N"|"O"|"P"|"Q"|"R"
                  |"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z";
        digit = "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|"0";
```

11. Termination

```
        ter = ";";
```

2 EBNF Errors

NOTE: The software will only report the first error on each line and once it had found an error it will move to the next semicolon to resume parsing.

1. Invalid Initialisation Sentence

   (a) Missing keywords - eg. missing “is”, ”with”, ”initially_at”.

```
G1 is AND 2 inputs;
         ^
SYNTAX[Invalid Initialisation]: Missing keywords
```

   (b) Invalid device name - eg. “123”,”%$”.

```
123 is AND with 2 inputs;
  ^
SYNTAX[Invalid Initialisation]: Invalid device name
```

   (c) Missing/invalid settings of properties

      i. Unspecified number of inputs;Unspecified/invalid initial state for switch; N=”abc” for simulation cycle.

```
G1 is AND with x inputs;
                ^
SYNTAX[Keyword Not Found]: Invalid keyword
```

      ii. Specifying settings of properties with some random keywords.

```
G1 is AND with with inputs;
                    ^
SYNTAX[Invalid Initialisation]: Invalid setting
```

2. Invalid Connection Sentence

   (a) Missing keywords - eg. “connect_to”.

```
SW1 G1.I1;
   ^
SYNTAX[Invalid Connection]: Missing keywords
```

   (b) Invalid device output/input

      i. Invalid output/input point eg. ”123”,”%$”

```
SW1 connect_to G1.O1;
                   ^
SYNTAX[Keyword Not Found]: Invalid keyword
```

      ii. Specifying device output/input with some random keywords

```
SW1 connect_to with;
                   ^
SYNTAX[Invalid Connection]: Invalid device I/O
```

3. Invalid Monitoring Sentence

   (a) Missing keywords -eg. ”Initial_monitor_at”.

```
MONITOR; G1;
        ^
SYNTAX[Invalid Monitor]: Missing keywords
```

(b) Invalid monitor point - eg. Invalid device output "123" etc.

```
Initial_monitor_at 123;
                   ^
SYNTAX[Invalid Monitor]: Invalid monitor point
```

4. Incomplete Description

   (a) No(too less) initialisation/No connection.

```
INIT; G1 is AND with 2 inputs; SW1 is SWITCH initially_at 0;
SW2 is SWITCH initially_at 0; CONNECT;
MONITOR; Initial_monitor_at G1;

CONNECT;
      ^
SYNTAX[Incomplete File]: Missing sentences for CONNCET
```

   (b) Missing start mark - eg. Missing "CONNECT".

```
INIT; G1 is AND with 2 inputs; SW1 is SWITCH initially_at 0;
SW2 is SWITCH initially_at 0;
SW1 connect_to G1.I1; SW2 connect_to G1.I2;
^
SYNTAX[Incomplete File]: Missing start mark CONNECT
```

5. Missing Termination Mark

```
SW1 connect_to G1.I1  SW2 connect_to G1.I2;
                      ^
SYNTAX[No Termination]: Missing termination mark
```

6. Keyword Not Found

   (a) Any keywords not defined - eg. "gate","and".

```
G1 is NOT with 1 input;
      ^
SYNTAX[Keyword Not Found]: Invalid keyword
```

7. Invalid Comment

```
/* comment (EOF)
         ^
SYNTAX[Invalid Comment]: Missing end comment mark '*/'
```

3 Semantic Errors

NOTE: Any semantic errors in the initialisation will not lead to a semantic error in the connection. Only the first error in the monitoring will be shown (either syntax or semantic). All semantic errors raised from syntax errors will be reported.

1. Duplicate initialisation - This occurs when one device of the same name is initialised more than once.

```
G1 is OR with 2 inputs;
G1 is AND with 20 inputs;
 ^
SEMANTIC[INIT]: Device is being initialised twice
```

2. Invalid number of inputs - This occurs when a AND, NAND, OR, or NOR gate is initialised with a number of inputs, not in the range 1-16.

```
G1 is OR with 2 inputs;
G2 is AND with 20 inputs;
                ^
SEMANTIC[INIT]: Device is initialised with wrong qualifier
```

3. Multiple outputs connected to one input - When multiple outputs are connected to one input, an ambiguity is created and thus is an error.

```
G2 is_connected_to G1.I1;
G3 is_connected_to G1.I1;
                        ^
SEMANTIC[CONNECT]: Input is already connected
```

4. Input not connected - All input of the circuit should be connected to make the circuit make sense.

```
INIT; G1 is AND with 2 inputs; SW1 is SWITCH initially_at 0;
CONNECT; SW1 connect_to G1.I1;
MONITOR; Initial_monitor_at G1;

INIT; G1 is AND with 2 inputs; SW1 is SWITCH initially_at 0;
   ^
SEMANTIC[CONNECT]: There are unused inputs
```

5. Dtype CLK needs to connect to a clock output - This error occurs when the CLK of a Dtype is not connected to a clock output if the CLK is not connected to anything then this error is raised instead of the previous.

```
INIT; D1 is DTYPE; SW1 is SWITCH initially_at 0;
CONNECTION;
SW2 connect_to D1.CLK;

SW2 connect_to D1.CLK;
                   ^
SEMANTIC[CONNECT]: The input CLK of a dtype is not connected to CLOCK
```

6. RC device needs to connect to a Dtype SET or CLEAR port - This error occurs when the RC device is connected to a port that is not either Dtype SET or CLEAR port. This error will not raise if RC is initialised but not connected.

```
INIT; D1 is DTYPE; RC1 is RC with_simulation_cycles 6;
CONNECTION:
```

```
RC1 connect_to D1.DATA;

RC1 connect_to D1.DATA;
                    ^
SEMACTIC[CONNECT] An RC is expected to be connected to a DTYPE SET or CLEAR
```

7. Referencing non-existing input/output/device name

```
INIT; G1 is AND with 2 inputs; SW1 is SWITCH initially_at 0;
SW2 is SWITCH initially_at 0;
CONNECT; SW1 is_connected_to G1.I1; SW2 is_connected_to G1.I2;
MONITOR; Initial_monitor_at G2;

Initial_monitor_at G2;
                   ^
SEMANTIC[REFERENCE]: Referencing a nonexisting device
```

Logic Simulator – User Guide

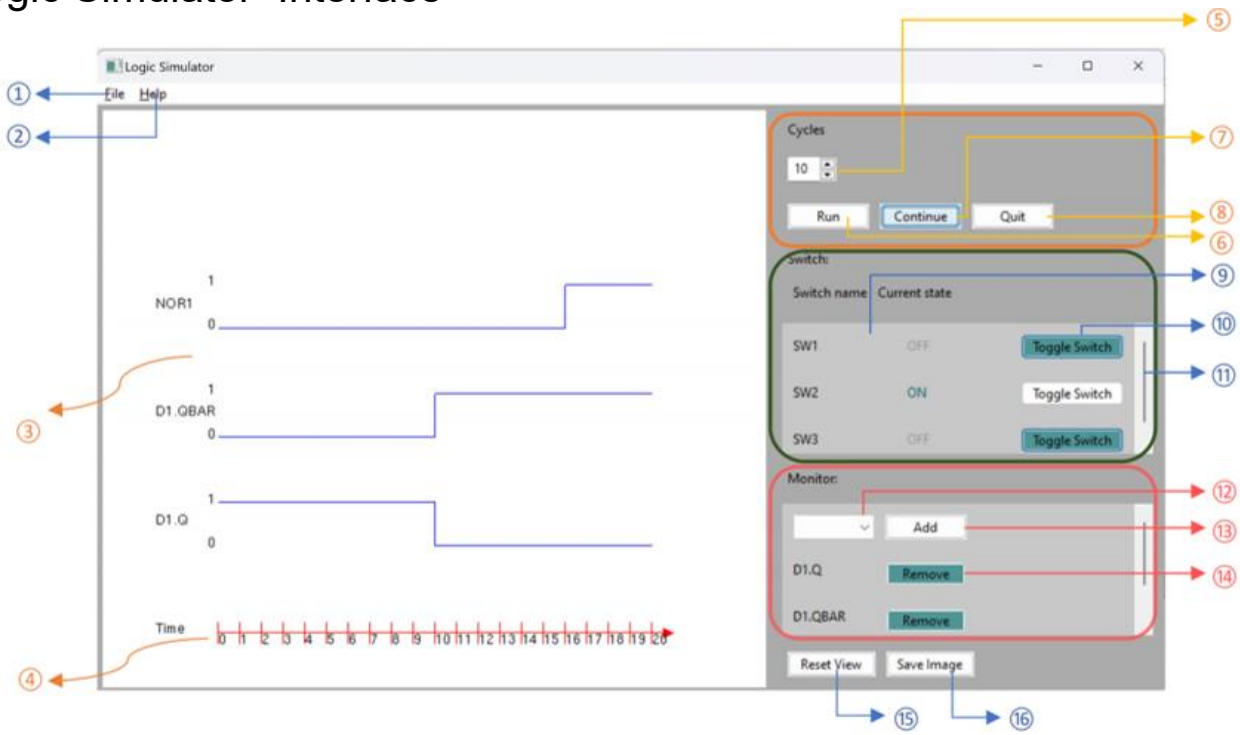
Start to use

Step 1: Write a txt description file. (See more instructions in EBNF rules.)

Step 2: Decide which version of simulator want to run and type the command line.
Normal version: python logsim.py <your file path>
Japanese version: python logsim.py -j <your file path>
Command-line version: python logsim.py -c <your file path>

Step 3: Now the Logic Simulator is launched, play with anything you like!

Logic Simulator Interface



1. **File**: Click to load new file / See information about developers / Quit program.
2. **Help**: Click to show help, all the instructions are here.
3. **Simulated signal graphs**: Display the simulated results. You can drag/zoom in the graphs.
4. **Time axis**: Time axis align with the Simulated signal graphs.
5. **Cycles**: Change the number of simulation cycles.
6. **Run**: Click to run from the start.
7. **Continue**: Click to continue running.
8. **Quit**: Click to quit the program.
9. **Switches**: Show all the information about switches.
10. **Toggle switch**: Click to toggle the switch.
11. **Scroll bar**: Scroll down to see more switches you defined.
12. **Dropdown list of output points**: Select the device output you want to monitor here.
13. **Add monitor points**: Click to add the monitor.
14. **Remove monitor points**: Click if you want to remove that monitor.
15. **Reset View**: Reset the zoomed/dragged graph to its original size and position.
16. **Save Image**: Click to save the simulated graph to your local computer.