FILE FORMATS FOR SIS PACKAGE

The various file formats that will be used in this course are:

1. PLA
2. BLIF
3. EQN
4. KISS

1. PLA format - Programmable Logic Array

Given a circuit, how do we describe it in the PLA format?

Consider the following circuit.

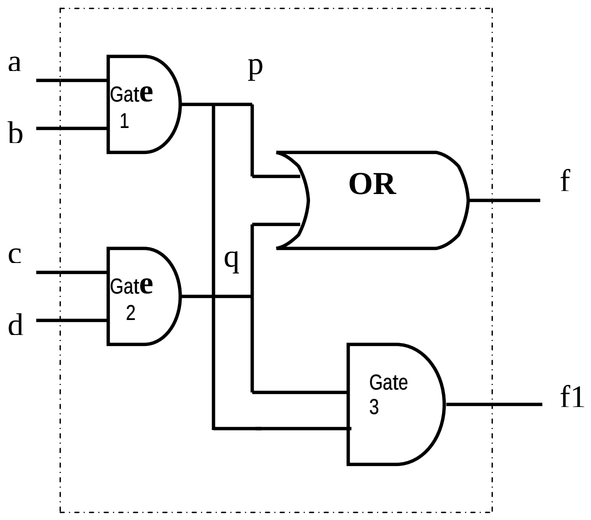


Fig. 1

In the above circuit,

1. Note the number of inputs, it is 4. (a,b,c,d) **{Specified by ".i"}**

2. Note the number of outputs, it is 2. (f, f1) **{Specified by ".o"}**

So, in the pla file we write

```
.i 4
.o 2
```

←PLA file

Naming the inputs and outputs

3. We define the names of the wires in the input.

   For the above circuit it is a, b, c, d

4. We define the names of wires in the output.

   For the above circuit it is f, f1

```
.i  4
.o 2
.ilb a b c d
.ob f f1
```

←PLA file

Giving the Truth Table

5. After specifying the inputs and outputs, we specify the truth table of the circuit. The number of terms in the truth table is represented by ".p" in the pla file.

A	B	C	D	F	F1
1	1	-	-	1	0
-	-	1	1	1	0
1	1	1	1	1	1

For the above truth table, the last 4 lines in the following snapshot have been added:

```
.i  4
.o 2
.ilb a b c d
.ob f f1
.p 3
 11-- 10
 --11 10
1111 11
```

←PLA file

Ending the file

6. The end of the file is specified using the notation “.e”.

```
.i  4
.o 2
.ilb a b c d
.ob f f1

 11--  10
 --11  10
 1111 11
.e
```

Final PLA file

2. Blif Format –Berkeley Interface Logic Format

Specifying the Model

Every BLIF file begins with the model declaration section. This model description is used to refer to the current circuit in other BLIF files.

A model is a flattened hierarchical circuit. A blif file can contain many models and references to models described in other blif files. A model is declared as follows:

```
.model <decl-model-name>
.inputs <decl-input-list>
.outputs <decl-output-list>
.clock <decl-clock-list>
<command>
.
.
<command>
.end
```

For the same circuit, as described in Fig. 1 for the PLA format,

1. Note the number of inputs, it is 4. (a,b,c,d) **{Specified by ".inputs"}**

2. Note the number of outputs, it is 2. (f, f1) **{Specified by ".outputs"}**

So, in the Blif file, we write:

```
.model example
.inputs  a,b,c,d
.outputs f,f1
```

← BLIF File

Naming the inputs and outputs

3. We define the names of the wires in the input.

   For the above circuit it is a, b, c, d

4. We define the names of wires in the output.

   For the above circuit it is f, f1

Giving the Circuit description

5. In the blif format we can specify each gate in the network by using the command "**.names**".

For example, the gate 1 in the above circuit is written as

```
.names a b p
11 1
```

What does this mean? The output p is high when the inputs a,b are both high. In fact, this represents an AND gate in the circuit. Similarly for Gate 2 in the above circuit, we write

```
.names c d q
11 1
```

The output p is high when the inputs c,d are both high. This represents an AND gate in the circuit as well. The advantage is that for very large circuits specifying the truth table is very difficult. Imagine specifying the truth table for circuit of 128 inputs and 5 outputs. Under such cases using blif format would be easier. Blif formats also support sequential elements like latches.

```
.model example
.inputs  a,b,c,d
.outputs f,f1

.names a b p
11 1
.names c d q
11 1
.names p q  f
1-  1
- 1  1
.names p q f1
11 1
```

← BLIF File

Ending the file

6. The end of the file is specified using the notation “**.end**”.

```
.model example
.inputs  a,b,c,d
.outputs f,f1

.names a b p
11 1
.names c d q
11 1
.names p q  f
1–  1
- 1  1
.names p q f1
11 1
.end
```

← **FINAL BLIF File**

For more information on the blif formats (for specifying flip flops, clocks, sub-circuits), check out the documentation provided in the Tutorials section under the additional reading/reference material or download it here.

3. EQN format (Equation Format)

The Equation format is one of the easiest formats since, it is a straightforward logic expression representation of the various logic gates in the circuit.

In the above circuit

1. Note the number of inputs, it is 4. (a,b,c,d) **{Specified by "INORDER"}**

2. Note the number of outputs, it is 2. (f, f1) **{Specified by "OUTORDER"}**

So, in the EQN file, we write

```
INORDER = a b c d;
OUTORDER = f  f1;
```

← EQN File

Specifying Intermediate Signals

Intermediate signals are specified using the notation as shown below,

[signal name] = logic expression;

For example, in the circuit shown in Fig. 1, the intermediate signal **p** can be represented as

[p] = a * b;

```
INORDER = a b c d;
OUTORDER = f  f1;
[p] = a * b;
[q] = c* d;
```

← EQN File

Specifying the Outputs

The outputs in the circuit can be expressed as a logic function of both the intermediate and inputs signals in a circuit. For the circuit in Fig. 1, the Outputs f, f1 can be written as:

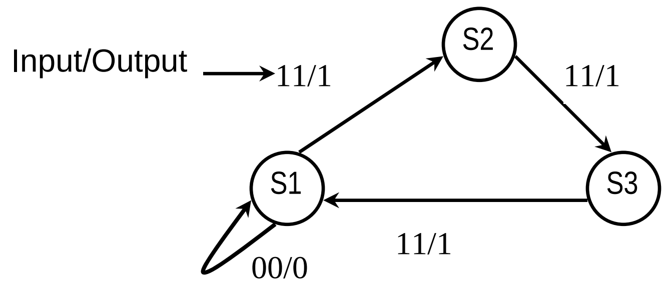
```
INORDER = a b c d;
OUTORDER = f  f1;
[p] = a * b;
[q] = c* d;
f= [p] + [q]
f1= [p]*[q];
```

← **Final EQN File**

4. KISS Format - **(Keep It Simple Stupid)**

Kiss format is used for state diagram minimization.

Given a state diagram as shown below,



Writing the Kiss format for the above state diagram:

In the above state diagram

1. Note the number of inputs, it is 2. (a, b) **{Specified by ".i"}**
2. Note the number of outputs, it is 1. (f) **{Specified by ".o"}**
3. Number of states is 3. **{Specified by ".s"}**
4. Number of terms in the state table is 4. **{Specified by ".p"}**

So, in the kiss file, we write

```
.i  2
.o 1
.s 3
.p 4
```

← Kiss File

Giving the State Table

After specifying the inputs and outputs, we specify the state table of the circuit.

Input	**PS**	**NS**	**Output**
00	S1	S1	0
11	S1	S2	1
11	S2	S3	1
11	S3	S1	1

PS-Present State
NS-Next State

```
.i  2
.o 1
.s 3
.p 4
00 s1 s1 0
11 s1 s2 1
11 s2 s3 1
11 s3 s1 1
```

← Kiss File

Ending the file

5. The end of the file is specified using the notation ".e"

```
.i  2
.o 1
.s 3
.p 4

00 s1 s1 0
11 s1 s2 1
11 s2 s3 1
11 s3 s1 1
.e
```

← FINAL Kiss File