# Extended BLIF Specification with Black and White Boxes

**Alan Mishchenko**

Department of EECS, University of California, Berkeley

alanmi@eecs.berkeley.edu

October 20, 2009

## Abstract

*Industrial designs are composed of several types of logic blocks: those to be synthesized, those already synthesized (such as adders), and those with unknown/irrelevant logic representation (such as memories). The latter two types of blocks should not be (or cannot be) synthesized and may be abstracted as boxes. This technical report describes a format to represent such boxes. The boxes can be white (logic is known) and black (logic is unknown). An implementation of a design flow with boxes is available in the synthesis and verification system ABC. The logic and timing information of the boxes is used in several optimization steps: AIG rewriting, technology mapping, and sequential synthesis. Future work will extend the use of boxes to the computation of structural choices and optimization with don't-cares.*

## 1 Overview

This report describes the extended BLIF format used for representing hierarchical sequential designs on the level of gates. Described below are the changes to the standard BLIF [1] made to specify white and black boxes, as implemented in ABC [2].

The design should be specified in one BLIF file. Many-file designs are currently not supported. The root model should be listed at the beginning of the file. The auxiliary models can follow in any order. Only two levels of hierarchy are currently supported.

## 2 Root model timing

The root model can have timing information for all or some of the inputs/outputs specified. The input timing information in the root model is represented using directive *.input_arrival <name> <float>*. By default, if the arrival time of an input is not specified, it is assumed to be zero. It is allowed to use expression *-inf* instead of *<float>* to represent an early arrival time of an input.

The output timing information in the root model is represented using directive *.output_required <name> <float>*. By default, if the required time of an output is not specified, it is assumed that the output is required as early as possible, and the synthesis system will try to minimize the arrival time of this output.

## 3 Box representation

Summarized in Table 1 are several types of boxes allowed by the extended BLIF format. The first three types should be specified using directive *.attrib*. For example, *.attrib white seq keep box* means that the sequential logic (*seq*) of the box is known (*white*) and the box cannot be removed (*keep*). There may be one or more *.attrib* directives per box. The forth box type (*merge/no_merge*) cannot be listed on the *.attrib* line because is applied individually for each box output. For example, if a box has four outputs but only the first two cannot be merged, it can be represented as follows: *.no_merge box_out1 box_out2*.

Table 1: Summary of box attributes in extended BLIF.

| Box attribute | Option 1 (default) | Option 2 |
|---|---|---|
| Object type | *box* (box should not be collapsed after reading) | *logic* (box will be collapsed after reading) |
| Presence type | *white* (logic is known) | *black* (logic is unknown) |
| Logic type | *comb* (logic is combinational) | *seq* (logic is sequential) |
| Persistence type | *sweep* (box is removable) | *keep* (box is not removable) |
| Merging type | *merge* (merging of an output is allowed) | *no_merge* (merging of an output is not allowed) |

## 4 Box timing

Both white and black boxes can have timing information specified. Representation of this information for combinational and sequential boxes is different.

For combinational boxes, directive *.delay* is used. There are three ways of using this directive. If only one number is listed, *.delay <float>*, it means that every input-to-output connection of the box has the same delay. If there is a signal name followed by the number, *.delay <name> <float>*, it means that any input-to-output connection involving the signal (either input or output) has the same delay. Finally, individual input-to-output delays are specified using directive *.delay <inname> <outname> <float>*. If more than one *.delay* directive is present, the later one overrides the former one. If there is no *.delay* directive, the combinational box is assumed to have unit delay, that is, the delay from any input to any output is equal to 1.

Delay of sequential boxes is specified using two directives: *.input_required <input > <float>* and *.output_arrival <output> <float>*. The first directives represents the required times for the box inputs while the second one represents the arrival times of the box outputs, counting from the last clock edges. In a way, the representation of timing for the inputs/outputs of the box is similar to the representation of timing for outputs/inputs of the root model, but the directive names are different.

## 5 Register classes

The *.latch* directive used to specify registers (flip-flops) in the traditional BLIF format looks as follows: *.latch <input> <output> <type> <control> <init_val>,* where *<type>* is one of {*ffe, re, ah, al, asg*}, which correspond to "falling edge", "rising edge", "active high", "active low" and "asynchronous".

To represent sequential designs with multiple clock domains and to distinguish between different types of registers belonging to the same clock domains (for example, rising-edge and falling-

edge registers), the extended BLIF format allows for representing a register class.

The *.latch* directive in the extended BLIF can list <*type*> as an integer number. This number represents the class of a given register. The register class can be used to limit transformations during sequential synthesis. For example, two registers that are proved equivalent cannot be merged if they belong to different class. When computing equivalent registers, computation is limited to one clock-domain at a time while the registers of other domains are treated as primary inputs. In the example below, the second registers is listed as having class 15.

## 6 Flip-flops

The *.latch* directive of the traditional BLIF allows us to represent a clocked flip-flop but it does not allow us to represent set, reset, enable, etc. One way to circumvent this limitation is to use white-boxes for each flip-flop type. An easier way, however, is to use a new directive *.flop* added to the extended BLIF.

The syntax of the *.flop* directive is the following: the keyword *.flop* is followed, in any order, by flop input (*D*=<signal_name>), flop output (*Q*=<signal_name>), optional clock (*C*=<signal_name>), optional set (*S*=<signal_name>), optional reset (*R*=<signal_name>), and optional enable (*E*=<signal_name>). The following two optional attributes (*async* and *negedge*) can specify asynchronous reset and negative-edge triggered clock, along with initial state attribute (*init*=<integer_value>). The default is the don't-care (*init*=2).

For example, the following extended BLIF line

*.flop async init*=1 *D*=*n*11 *C*=*n*22 *R*=*n*14 *Q*=*n*16

represents a flop with asynchronous reset, initialized to value 1, with input *n*11, clock *n*22, reset *n*14, and output *n*16.

## 7 Example

The following example is included to illustrate the use of the extended BLIF format. It can be cut-and-pasted into a file and read into one of the latest versions of ABC using command **r*.

```
# This is an example of a design in extended BLIF
# illustrating the use of black/white boxes and
# timing info for the boxes and the root model.
.model example
.inputs a0 b0 a1 b1 C CE R S
.outputs s0 cout0 s1 cout1 Out
.input_arrival a0 1.0
.input_arrival b0 -inf
.output_required s0 2.0
.output_required s1 inf
# representation of subcircuits
.subckt FA_black a=a0 b=b0 cin=Zero \
        s=d0 cout=cout0
.subckt FA_white a=a1 b=b1 cin=Zero \
        s=d1 cout=cout1
.subckt FDRSE_black C=C CE=CE D=d0 R=R S=S Q=s0
.subckt FDRSE_white C=C CE=CE D=d1 R=R S=S Q=s1
# representation of combinational nodes
.names Zero
.names a0 b0 a1 b1 Out
11-- 1
--11 1
.end

# full adder as a black box
.model FA_black
.inputs a b cin
.outputs s cout
.attrib black box comb
.delay a s 0.01
.delay b s 0.01
.delay cin s 0.01
.delay a cout 0.02
.delay b cout 0.02
.delay cin cout 0.02
.end

# full adder as a white box
.model FA_white
.inputs a b cin
.outputs s cout
.no_merge s
.attrib white box comb
.delay a s 0.01
.delay b s 0.01
.delay cin s 0.01
.delay a cout 0.02
.delay b cout 0.02
.delay cin cout 0.02
.names a b cin s
100 1
010 1
001 1
111 1
.names a b cin cout
-11 1
1-1 1
11- 1
.end

# complex flop as a black box
.model FDRSE_black
.attrib black box seq
.inputs C CE D R S
.outputs Q
.input_required C 0.0
.input_required CE 0.1
.input_required D 0.1
.input_required R 0.3
.input_required S 0.3
.output_arrival Q 0.4
.end

# complex flop as a white box
.model FDRSE_white
.attrib white box seq
.inputs C CE D R S
.outputs Q
.input_required C 0.0
.input_required CE 0.1
.input_required D 0.1
.input_required R 0.3
.input_required S 0.3
.output_arrival Q 0.4
.latch int_r Q 15 C 1
.names D CE Q int_ce
-01 1
11- 1
.names int_ce S int_s
-1 1
1- 1
.names int_s R int_r
10 1
.end
```

## 8 Conclusions and future work

The hierarchical BLIF representation is extended by adding several new directive *.attrib* as well as several directives to represent the timing information.

The following features in this report have not been fully implemented, or are implemented but not sufficiently tested:

- Timing information for sequential boxes.
- Required times for the outputs of the root model.

## Acknowledgements

This work is supported in part by SRC contracts 1361.001 and 1444.001, NSF grant CCF-0702668 entitled "Sequentially Transparent Synthesis", and the California MICRO Program with industrial sponsors Actel, Altera, Calypto, IBM, Intel, Intrinsity, Magma, Synopsys, Synplicity, Tabula, and Xilinx.

## References

[1] Berkeley Logic Interchange Format (BLIF), **http://vlsi.colorado.edu/~vis/blif.ps**

[2] Berkeley Logic Synthesis and Verification Group. *ABC: A system for sequential synthesis and verification*. **http://www-cad.eecs.berkeley.edu/~alanmi/abc** (recent versions available upon request)