The ISCAS '85 benchmark circuits and netlist format

David Bryan, MCNC
919-248-1432
email: bryan@mcnc.org
9-30-88

The ISCAS '85 benchmark circuits are ten combinational networks provided to authors at the 1985 International Symposium on Circuits And Systems.[1] They subsequently have been used by many researchers as a basis for comparing results in the area of test generation. Each circuit is characterized in the table below:

Circuit Name	Circuit Function	Total Gates	Input Lines	Output Lines	Faults[1]
C432	Priority Decoder	160 (18 EXOR)	36	7	524
C499[2]	ECAT	202 (104 EXOR)	41	32	758
C880	ALU and Control	383	60	26	942
C1355[2]	ECAT	546	41	32	1574
C1908	ECAT	880	33	25	1879
C2670	ALU and Control	1193	233	140	2747
C3540	ALU and Control	1669	50	22	3428
C5315	ALU and Selector	2307	178	123	5350
C6288	16-bit Multiplier	2406	32	32	7744
C7552	ALU and Control	3512	207	108	7550

[1]Reduced equivalent fault set based on equivalence fault collapsing.
[2]Circuits C499 and C1355 are functionally equivalent. All EXOR gates of C499 have been expanded into their 4-NAND gate equivalents in C1355.

Table 1: ISCAS '85 Benchmark Circuit Characteristics

The ISCAS '85 netlist format was never formally documented; rather, it was distributed on magnetic tape along with a FORTRAN translator that would generate netlists in a few other

[1]F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", distributed on a tape to participants of the Special Session on ATPG and Fault Simulation, Int. Symposium on Circuits and Systems, June 1985; partially characterized in F. Brglez, P. Pownall, R. Hum, "Accelerated ATPG and Fault Grading via Testability Analysis", Proc. IEEE Int. Symposium on Circuits and Systems, pp. 695-698, June 1985, reprint of the article is available with the tape from MCNC.

formats. Now, a new translator written in C is being distributed with the ISCAS '85 benchmarks that outputs a single, generic-looking format which is more useful for translation to other formats (e.g., hilo, cadat). The ISCAS '85 format has, however, become viable despite its shortcomings.

One reason for this is that it contains information not present in most other netlist formats. For instance the ISCAS '85 format lists each network node in levelized order[2], and this information may be lost when translating to other formats. Also, the ISCAS '85 format lists fanout branches separately as distinct nodes (with distinct names) and specifies the connectivity for each fanout branch. This is valuable for test generation purposes and must be extracted from other, more generic, netlists.

The remainder of this document explains the ISCAS '85 netlist format. As an example, a small, six-NAND-gate circuit, known as "c17", will be used. Listed below is the netlist for c17:

```
* These first five lines are comments.
* The comment character is the "*" (asterisk).
* The comment character may appear anywhere on a
   * line and remains in effect until the end of
                          * the line is reached.
1     1gat inpt    1   0      >sa1
2     2gat inpt    1   0      >sa1
3     3gat inpt    2   0 >sa0 >sa1
8     8fan from     3gat      >sa1
9     9fan from     3gat      >sa1
6     6gat inpt    1   0      >sa1
7     7gat inpt    1   0      >sa1
10   10gat nand    1   2      >sa1
  1     8
11   11gat nand    2   2 >sa0 >sa1
  9     6
14   14fan from    11gat      >sa1
15   15fan from    11gat      >sa1
16   16gat nand    2   2 >sa0 >sa1
  2    14
20   20fan from    16gat      >sa1
21   21fan from    16gat      >sa1
19   19gat nand    1   2      >sa1
 15     7
22   22gat nand    0   2 >sa0 >sa1
 10    20
23   23gat nand    0   2 >sa0 >sa1
 21    19
```

Although the netlist translator provided with the original distribution was written in FORTRAN, the input is format-free (i.e., the data is not located by its columnar position in a line).

[2]Usually, primary inputs are assigned a level of zero. Each time a gate is traversed, en route to a primary output, the level is incremented by one. Nodes are in no particular order within a given level. For example, all the primary inputs are guaranteed to appear before any other nodes, but the order of appearance of the primary inputs is unspecified.

The valid delimiters between fields of data are: spaces, horizontal tabs and newlines.

The first data line of this netlist tells the following information about a single node in the circuit:

- the node *address* is *1* (the first entry on the line)
- the node *name* is *1gat* (the second entry on the line)
- the node *type* is *inpt* (the third entry on the line)
- the node *fanout* is *1* (the fourth entry on the line)
- the node *fanin* is *0* (the fifth entry on the line)
- the node *fault(s)* is >*sa1* (the last entry on the line)

The definitions for each field are,

- *address* – a unique number that differentiates this node from all others in the circuit
- *name* – a string of characters used to provide more meaningful information about the node usage (hopefully)
- *type* – the function performed by the gate driving this node. Legal node types are:
  1. *inpt* (a primary input; not driven)
  2. *and*
  3. *nand*
  4. *or*
  5. *nor*
  6. *xor*
  7. *xnor*
  8. *buff* (a non-inverting buffer)
  9. *not* (an inverter)
  10. *from* (a fanout branch; driven by a fanout stem)
- *fanout* – the number of gates driven by this node
- *fanin* – the number of nodes driving the gate that drives this node
- *fault(s)* – the stuck-at fault(s) on this node that are included in the fault set. Possible values are >*sa0* for stuck-at-zero and >*sa1* for stuck-at-one.

This type of line, called the *node line*, provides the basic information for every node in the circuit. However, there are two other types of lines that may be associated with it. The first of these is the *fanin line*. The fanin line provides a list of the addresses of the nodes that fan in to the gate driving this node. It always appears immediately after its associated node line. Note that primary inputs do not have a fanin line because, by definition, primary inputs have a fanin of zero. Also, the number of addresses appearing on the fanin line will equal the fanin reported on the node line. For example, looking at the node with address 10 in the c17 example, we see it has a fanin of 2 and the two fanin addresses are 1 and 8. The node with address 1 is a primary input and the node with address 8 is a fanout branch (type *from*). This brings us to the third and final type of line in the ISCAS '85 format, the *fanout branch line*. Fanout branch lines are similar to node lines in that they have an address, a name and a type (always *from*), but the information is associated with a single fanout branch of a *fanout stem*. Fanout branch lines must appear immediately after the node line (and its fanin line) with which they are associated. Since fanout branches always have a fanout and a fanin of one, this information is not listed as with a node line. Instead, the name of the fanout stem is given after the type. This is redundant information, but, nevertheless, it is part of the format. As a consequence of having a fanout of one, fanout branches may never represent a primary output since primary outputs always have a fanout of zero. Also, note that nodes with a fanout less than or equal to one do not have fanout branches. If the fanout is greater than one, the number of fanout branch lines will equal the fanout reported on the node line.

Finally, let's look at a node of c17 that ties the three types of lines together. The node with address 16, named 16gat, is type NAND, has a fanout of 2 and a fanin of 2. Immediately following the node line is the fanin line which lists node addresses 2 and 14 as the inputs: a primary input and a fanout branch of a NAND gate. Following the fanin line are the two fanout branch lines; one for each fanout. The netlist then continues with the next node.

There is one last caveat. No formal language description such as BNF exists for the ISCAS '85 format. It is up to the user to provide a parser based on the description given above. Also, there is no maintenance of the ISCAS language. It is considered frozen as is. Anyone who wishes to make improvements to the language is on their own.