Standard Test Interface Language (STIL)

A New Language for Patterns and Waveforms

Tony Taylor

Credence Systems Corporation, 215 Fourier Ave. Fremont, CA 94539

Gregory A. Maston

Motorola Corporation, 1300 N Alma School Rd. Chandler, AZ 85224

1. Abstract

This paper presents the major features and capabilities of the new Standard Test Interface Language (STIL). A synopsis of the language, typical applications of the language, work that has been done to date, and degree of acceptance by the industry are discussed.

2. Purpose

The purpose of STIL is to provide a common language for the representation of test patterns and waveforms that are created from the simulation of a digital integrated circuit. The language represents the data in terms of intent (i.e., the waveform that is to be created at the device under test), rather than in terms of a specific piece of ATE hardware. The language is flexible enough to represent patterns from simple to the most complex devices, and has built in optimizing features to minimize the volume of data that is generated by today's designs.

3. Background

Up until now the transfer of data from simulation into the ATE environment has been through the proprietary language of the individual ATE system. There have been other attempts at developing standard languages for this purpose (EDIF, WAVES, ATLAS), but none has been broad enough or efficient enough to meet the needs. None of these has achieved any degree of acceptance in the commercial test world.

Value Change Dump (VCD) formats have been the typical way of capturing simulation output. This is where every transition on each pin of the simulated device is captured in a sequence of timed events. This is fine for capturing a picture of the waveform, however it has limitations when used for creating test programs. VCD does not allow for any representation of the relation between events such as is needed to any kind of analysis or characterization of the pattern on a real device. The VCD format requires an involved process to make the waveform/pattern realizable on most ATE systems which is usually done by means of expensive and time consuming conversion software. Also, since the pattern/waveform is processed into something loadable and runnable on a particular tester, it usually loses all reference to the original source, making it extremely difficult to correlate device failures back to the simulation process.

It was because of the above situation that in the beginning of 1995, a number of Semiconductor manufacturers united in their push for a common language. Most tool vendors, and ATE vendors were quick to support this movement, as it was an obvious need, but one that no individual company could succeed in creating alone.

In February, a working group comprised of between 8 and 14 members was formed. This group, through monthly meetings and extensive phone/email communication, has developed the language

specification where it is now complete to the intent of the first level of expectation. In addition to the working group meetings, the full oversight group (comprised of approximately 40 supporting companies) has met on about a 3 month interval to provide direction, guidance, and approval of the work.

The work is being done under the guidelines of the IEEE. PAR1450 (Project Authorization Request) was approved on June 20, 1996. A panel session was conducted at ITC 1995 whereby the relative merits of the various standards was discussed, with consensus that there was room for, and indeed need for one more standard to address this need.

4. Overview of the standard

In the course of developing the standard and in the review process, it became clear that the presentation of the standard was almost as important as the standard itself. People upon their introduction to the standard tended to look for things that were important to them or their application, which may not have been part of the intent of STIL (at least for the first revision). For example, many things like flow control, parametric testing, specific tester resource assignment were specifically not included in the charter.

To make the standard clear to the initial reader, the document is organized as follows:

1. Definition of the purpose, goals, glossary.
2. A tutorial section starting with a “hello tester” program illustrating the basics of the language, and advancing to complete examples of test patterns.
3. Reference sections containing the detailed definition of each statement in the language.
4. Appendices that define the syntax in BNF form, and example patterns to illustrate specific capabilities of the language that might not be implicit in the language definition itself.

What is not provided as a part of the standard is any actual software - i.e., writers, readers, API’s, tools, etc. The STIL standard establishes the language whereby individual companies can develop this software either for their own internal use or for commercial distribution.

5. Major Capabilities

The following sections comprise a brief run through each of the major areas of the language and explains what this section of the standard is used for.

Patterns

In STIL patterns are made up of a sequence of vectors, where a vector defines what occurs on each signal of the device and references a WaveformTable which defines the exact waveform for the activity on each signal. The pattern can therefore be thought of as periods of time (vectors) that are to be executed one after the other.

The format for a simple vector could be like: “W t1; V {CLK=K; DATA=FF02;}”, where t1 is the name of the WaveformTable, “K” is a reference to the format for the CLK signal, and “FF02” is hexadecimal data to be applied to the “DATA” bus. There may be many other signals on the device, whose states are defined by the state of the prior vector (i.e., only changing states need be specified)

Additional facilities exist for specifying timed events directly within a vector. The signal groupings may be varied from vector to vector.

Scan patterns

There are special constructs in STIL to define scan data. Consideration was given to partial scan chains, unequal scan chains, special load/unload requirements, and compact formats that can handle multiple states (i.e., output scan chains of L/H/X).

Scan data in STIL is defined in device primary I/O pin format rather that in terms of the internal nodes being loaded/unloaded. As such, the scan data is in a state ready for direct transfer to a tester. A special construct in STIL defines the internal node structure so that the scan data can be referenced back to the nodes being defined.

Pattern Procedures and Macros

STIL patterns can be defined in terms of Macros or Procedures. The intent of a Macro is primarily a shorthand data representation and expected to be expanded for execution. The intent of a Procedure is to allow a tester that has procedure capability to load and execute optimally (some testers may still need to expand procedure calls).

Procedures and Macros are intended both for parallel vectors and scan vectors.

Timing and WaveformTables

The timing section is where the waveform for each signal of a vector is defined. The waveform is defined as an arbitrary long series of events in time. The events are fixed in definition and fall into three categories: 1) input or drive events, 2) output or compare events, and 3) events that are not yet fully resolved (i.e., need further processing before they are tester ready).

Each event has a time associated with it which may be an absolute number, an expression using variables from the "Specs" section, or expressions relative to other events in its own or other waveform definitions.

Waveforms may be constructed by inheriting data from other waveforms. Data inheritance may be done by inheriting all waveforms from another waveform table, and then redefining some of them. Data inheritance may also be used on an individual waveform, by inheriting another waveform and redefining some of the properties.

Specs and Variables

A spread sheet like structure is used for the definition of variables and expressions that are to be used in waveform and timing definitions. Variables may be arranged into categories, where each named variable may take on a new value or expression.

To conform to the way that a device spec sheet is often defined, each variable may define a "Min", "Typ", or "Max" value. Also for devices with relative timing, a "Meas" value allows the definition to be determined by a measurement that is made at pattern execution time.

A Selector section is used to specify for each pattern execution which of the various values of a variable are to be used.

Glue

The term "Glue" is used here to refer to the statements that connect the various data sections together into an executable sequence of patterns.

The "PatternExec" block defines the category selection for the Spec data, the timing selection for the vectors, and the patterns to be run.

The patterns may be organized into PatternBurst blocks which allow multiple patterns to be referenced as a single entity.

The Glue section also allows the definition or selection of one of many "Domains" to be selected. A "Domain" allows for a customized definition of the signal/group definitions, the Procedure/Macro definitions, and the WaveformTable definitions.

Modularity

Modularity refers to the ability to construct a set of patterns for a device from a common set of modules. This capability is becoming increasingly important as devices are built out of reusable parts (i.e., a processor module, a memory module, application modules). The modularity in STIL allows for these separate parts to have their own pattern files that can be included into a larger pattern set without conflict over name space.

Extensibility

It was recognized at the onset that STIL could not satisfy the needs of all users. Therefore, a basic requirement of the language is that it have a simple set of rules whereby any given user could extend the language and still have the syntax parsable by any given reader that follows a few simple rules.

A typical reason for extending STIL would be to insert tester specific information that is added by a tool set or translator designed as part of the importing process for that specific tester.

Another reason might be to incorporate functionality that has not yet been defined in the STIL standard.

Extending the language in this manner is like the definition of custom functions in a "C" program to add functionality beyond what standard library calls provide.

Binary

Much discussion and consideration was given to the creation of a standard that defined a binary format for pattern data.

It is clear that patterns lend themselves well to pattern compression techniques. However, there is a trade-off between the complexity of the pattern compression algorithm (and hence execution time

required by readers and writers of the compressed data, and difficulty of creation of readers and writers) and the effectivity of the compression. It is also recognized that any given compression scheme may not be satisfactory for a given ATE vendor where the compression process may be designed to go along with a hardware design that loads the data into the tester.

The conclusion, after much debate was to utilize in STIL existing compression standards that are available in the public domain - gzip and gunzip. These were found to be comparable in their effectivity to any custom compacting algorithms that could be defined for patterns.

6. Example STIL File

This section contains a short, but complete example of a STIL file. The pattern is for a 74LS245 device (octal bus transceiver). This example illustrates much of the capability of the language. In particular it shows how a test pattern can be represented that contains complex timing relationships and multiple categories for the variables that are used in the timing relationships. This and other examples, ranging from a simple “hello tester” to the more complex, are explained in detail in the STIL standard document.

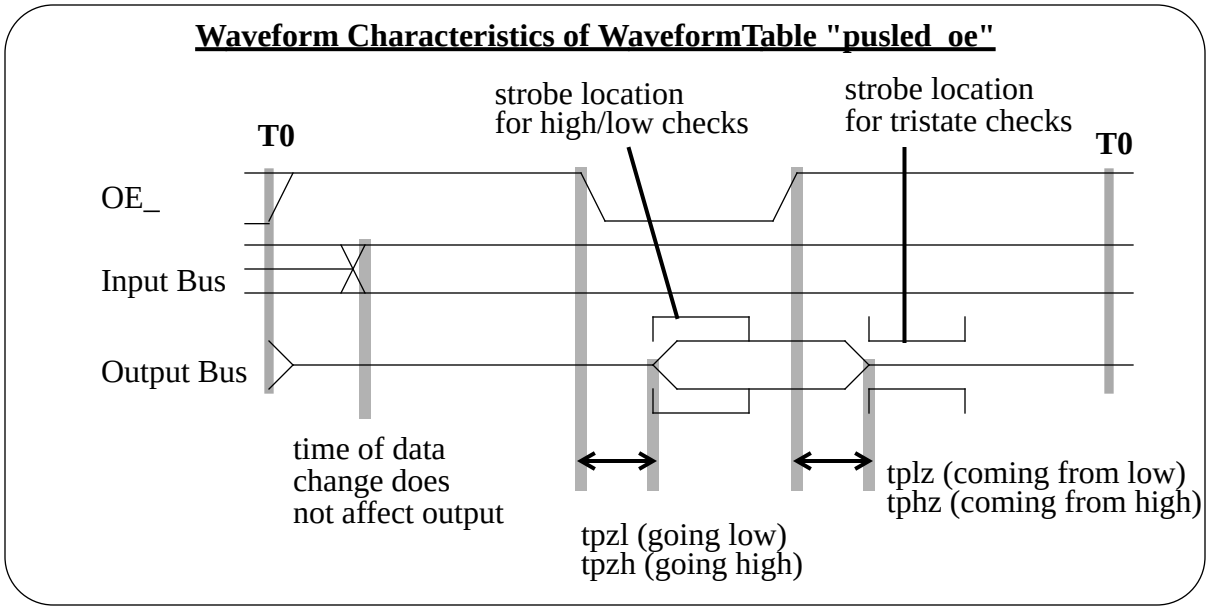
```
STIL 0.0;
Signals {
   DIR In;
   OE_ In;
   A0 InOut; A1 InOut; A2 InOut; A3 InOut;
   A4 InOut; A5 InOut; A6 InOut; A7 InOut;
   B0 InOut; B1 InOut; B2 InOut; B3 InOut;
   B4 InOut; B5 InOut; B6 InOut; B7 InOut;
}

SignalGroups {
      ABUS ’A7 + A6 + A5 + A4 + A3 + A2 + A1 + A0’;
      BBUS ’B7 + B6 + B5 + B4 + B3 + B2 + B1 + B0’;
      BUSES ’ABUS + BBUS’;
      ALL ’DIR + OE_ + BUSES’;
}

SignalGroups more {
   ABUS_I ’ABUS’ { Base Hex 01; }
   BBUS_I ’BBUS’ { Base Hex 01; }
}
```

```
Spec tmode_spec {
      Category tmode {
            tplh       { Typ ’8.00ns’; Max ’12.00ns’; }
            tphl       { Typ ’8.00ns’; Max ’12.00ns’; }
            tpzl       { Typ ’27.00ns’; Max ’40.00ns’; }
            tpzh       { Typ ’25.00ns’; Max ’40.00ns’; }
            tplz       { Typ ’15.00ns’; Max ’25.00ns’; }
            tphz       { Typ ’15.00ns’; Max ’25.00ns’; }
            strobe_width ’20ns’;
      }
}

Selector tmode_typ {
      tplh       Typ;
      tphl       Typ;
      tpzl       Typ;
      tpzh       Typ;
      tplz       Typ;
      tphz       Typ;
}
Timing to_specs {
   WaveformTable pulsed_oe {
       Period ’500ns’;
       Waveforms {
      DIR{ 01{ ’0ns’ D/U; }}
      OE_{ 01{ ’0ns’ U; OE_MARK: ’200ns’ D/U;
                  OE_CLOSE: ’OE_MARK+100ns’ U; }}
      BUSES{ 01{ ’10ns’ D/U; }
            L  { ’0ns’ Z;’0ns’ X; ’OE_MARK+tpzl’ l;
                  ’@+strobe_width’ X;}
            H  { ’0ns’ Z;’0ns’ X; ’OE_MARK+tpzh’ h;
                  ’@+strobe_width’ X;}
            D  { ’0ns’ Z;’0ns’ X; ’OE_CLOSE+tplz’ t;
                  ’@+strobe_width’ X;}
            U  { ’0ns’ Z;’0ns’ X; ’OE_CLOSE+tphz’ t;
                  ’@+strobe_width’ X;}
            X  { ’0ns’ Z;’0ns’ X; }}
       }          // end Waveforms
   }              // end WaveformTable pulsed_oe
   WaveformTable const_oe {
       Period ’500ns’;
       Waveforms {
      DIR{ 01{ ’0ns’ D/U; }}
      OE_{ 01{ ’0ns’ D; ’480ns’ U;}}
      BUSES{ 01{ IN_MARK: ’50ns’ D/U; }
            L  { ’0ns’ Z;’0ns’ X; ’IN_MARK+tphl’ l;
                  ’@+strobe_width’ X;}
            H  { ’0ns’ Z;’0ns’ X; ’IN_MARK+tplh’ h;
                  ’@+strobe_width’ X;}
            X  { ’0ns’ Z;’0ns’ X; }}
       }          // end Waveforms
   }              // end WaveformTable const_oe
}                 // end Timing to_specs
```

```
PatternBurst spec_check_burst {
    spec_check;
}           //end PatternBurst spec_check_burst

PatternExec {
    SignalGroups more;
    Timing to_specs;
    Selector tmode_typ;
    Category tmode;
    spec_check_burst;
}           //end PatternExec


Pattern spec_check {
    W pulsed_oe;
    // the first vector must specify states on all signals.
    V { ALL=00DDDDDDDDXXXXXXXX; }

    // first set of tests check delays from OE_ signal
    // check BBUS tpzl spec
    V { ABUS_I=00; BBUS=LLLLLLLL; }
    //check BBUS tpzh spec
    V { ABUS_I=FF; BBUS=HHHHHHHH; }
    // check BBUS tplz spec
    V { ABUS_I=00; BBUS=DDDDDDDD; }
    // check BBUS tphz spec
    V { ABUS_I=FF; BBUS=UUUUUUUU; }
    V { DIR=1;  ABUS=XXXXXXXX;
    BBUS=DDDDDDDD; }
    // check ABUS tpzl spec
    V { BBUS_I=00; ABUS=LLLLLLLL; }
    // check ABUS tpzh spec
    V { BBUS_I=FF; ABUS=HHHHHHHH; }
    // check ABUS tplz spec
    V { BBUS_I=00; ABUS=DDDDDDDD; }
    // check ABUS tphz spec
    V { BBUS_I=FF; ABUS=UUUUUUUU; }

    W const_oe;
    // second set of tests check data propagation delays
    // check ABUS tphl spec
    V { BBUS_I=00; ABUS=LLLLLLLL; }
    // check ABUS tplh spec
    V { BBUS_I=FF; ABUS=HHHHHHHH; }
    V { DIR=0; BBUS=XXXXXXXX;
    ABUS=DDDDDDDD; }
    V { ABUS_I=00; BBUS=LLLLLLLL; }
    // check BBUS tplh spec
    V { ABUS_I=FF; BBUS=HHHHHHHH; }
    // check BBUS tphl spec
    V { ABUS_I=00; BBUS=LLLLLLLL; }
}           //end Pattern spec_check
```

7. Usage of STIL to Date

At this time, several ATE companies have developed parsers and readers that process STIL data. At ITC 1996, some ATE vendors are demonstrating STIL by using a reader to convert STIL data files into their proprietary language and using graphics pattern and waveform tools to display the data.



Several semiconductor design companies have plans to incorporate STIL into their test generation process, and are actively working towards the completion of STIL to this end.

8. Plan for Future Work

As of this writing (June 1996), the STIL document is essentially complete from a technical point of view. Some of the details of the language are being revised and adjusted as a result of the on-going work that is being done using the language. The document is being revised to conform to the IEEE style guide for standards. Plans are going forward to begin the ballot process required to make STIL a standard.

Although no actual planning has been done for extensions to the current STIL language, there is much interest in adding syntax to address other areas of test beyond the digital test pattern itself. Areas that are prime are prime candidates are the addition of levels definition, addition of parametric test definition, addition of characterization test definition, and the addition of flow control.

9. Conclusion

The STIL standard has come a long way towards completion in the year and a half since this standards development group was formed. At this time it looks like it is well positioned to satisfying its original goal of providing a transportation mechanism from Design to Test for the bulk data of patterns.

10. References

W. Sebesta, B. Verhelst, M. Wahl, “Development of a New Standard for
Test”, Proc. of the International Test Conference, pp 988-993, 1990.

L. Moran, R. Hillman, P. Burlison, T. Gurda, “The Waveform and Vector
Exchange Specification (WAVES)”, Proc. of the International Test
Conference, pp. 978-987, 1990.