Application Note

Real-Time Pattern Generation Answers Functional Validation Challenges

Introduction

Digital modules and the systems they comprise are growing ever more powerful – a benefit that brings with it increased complexity. As this complexity grows, so too does the need for more rigorous debug, verification, and stress testing. These activities are components of an engineering process known as "functional validation."

In contrast to characterization measurements in which a device or System Under Test (SUT) is driven to the extremes of its performance envelope (usually to determine the unit's absolute operating limits), functional validation examines the SUT's behavior under relatively "normal" conditions. The purpose of functional validation is to test the unit's ability to perform its logic operations correctly, typically over a range of voltage and timing tolerances.

The applied test data is called the "pattern," and a functional test simply sends the pattern to a SUT and compares the device's actual reaction against predicted data. In reality, the functional test is a Boolean truth table of the SUT's logic states. The task of producing the stimulus falls to a tool known as a pattern generator, which is separate from and complementary to the test instrument's acquisition capability. Even though a functional test is usually limited to nominal operating values (frequency, timing, and voltage), the pattern generator is responsible for producing constantly varying stimulus information at high clock rates. To do so, it recognizes and reacts, in real time, to instructions from the SUT and the other elements of the measurement system.

This application note discusses the TLA7PG2 pattern generator, a module in the TLA 700 Logic Analyzer Series, and its use in the functional validation environment.

Overview of Functional Validation Requirements and Solutions

Digital system functional validation challenges are growing in several dimensions:

- Higher clock rates
- Lower and mixed logic voltages
- More I/O connections
- More functionality at every level (chip, module, bus, system)

Responding to the incessant call for digital systems that run bigger applications faster, designers are implementing higher clock speeds at both the chip and system level. This places more exacting demands on the functional validation test platform. Stimulus signals must be stable at frequencies into the hundreds of megahertz. In addition the signals must have faster edge rates and narrower data pulses than ever before. The TLA7PG2's programmable clock delivers frequencies up to 268 MHz. It provides accurate internally derived clock signals, and also accepts an external clock input and faithfully reproduces the duty cycle of the external signal. This feature is particularly important when testing systems that use both edges of the clock. Even a small amount of duty cycle distortion can cause SUT instability at today's higher operating frequencies

In pursuit of increased data bandwidth throughout a digital system, inter-module and intra-module buses are getting wider. Moreover, the number of buses is increasing as devices add new functions. Consequently the number of stimulus channels required for functional validation has grown dramatically. TLA7PG2 modules can be concatenated to achieve a wide range of channel counts – as few as 64 channels, or as many as 1024. This modular architecture makes it possible to configure for your initial requirements and expand cost-effectively as your needs change.

As SUT functionality increases, longer and more complex test patterns are required. That means an ever-increasing appetite for pattern memory. The TLA7PG2 offers up to 2M lines of pattern memory, ample for many applications. In addition, sequencing techniques expand the pattern memory, in effect, to infinite capacity.

Sequencing Multiplies Pattern Capacity Without Limit

At first glance, a functional test pattern might look rather imposing. If stretched "end to end," it would consist of many thousand, or even millions, of cycles of binary data. If it was necessary to store each cycle discretely in a conventional memory, the pattern generator would have to be huge – and costly. Fortunately, there is an elegant solution. What appears to be a long and convoluted string of binary information can be broken into a series of smaller, more manageable segments that repeat periodically.

For example, consider the problem of emulating a series of front-panel push-button actions. A button push may prompt a large number of operational cycles. But the resulting data might be very similar among all of the buttons, with only a few bits changing to distinguish a particular button. Even so, the SUT needs to see the whole series of cycles every time.

The brute-force method would be to store a copy of every button command with all its associated code. But this simplistic approach is wasteful. Obviously it would quickly consume the stimulus source's memory, especially if the routine is long. The button command is a perfect candidate to be defined once and used many times – if the fixed portion of the pattern can be separated from the bits that change with each new button (see Figure 1). This is where the TLA7PG2 pattern generator steps in.

The TLA7PG2 includes a sophisticated sequencer designed to take advantage of repetitive situations like this. The pattern generator's pattern memory can be partitioned into "blocks" of varying lengths. A block is a self-contained code module, potentially made up of many individual lines of pattern data. In our push-button example, the fixed portion of the button routine is defined as a block. The block is used time after time – but is stored only once. Smaller blocks containing the changing bits are interleaved with the fixed blocks, in effect producing a sequence of commands that moves from button to button.

This methodology does more than just save memory space. It allows you to easily program variations that produce stresses such as key bounce and key noise. These can be interspersed with the "good" button-push routines, or can be cycled as part of a separate stress test. It is also possible to emulate more human-like button pushes by adding time-delay blocks to slow the key presses. In any event, the same blocks can be re-used with a variety of sequences to emulate different hierarchies of button-pushes. Up to 4096 blocks can be defined within the pattern memory.

In this simple example, we have seen how block-based sequencing can conserve pattern memory. Why is that so important when you can have up to 2 Mb of pattern memory serving each test pin?

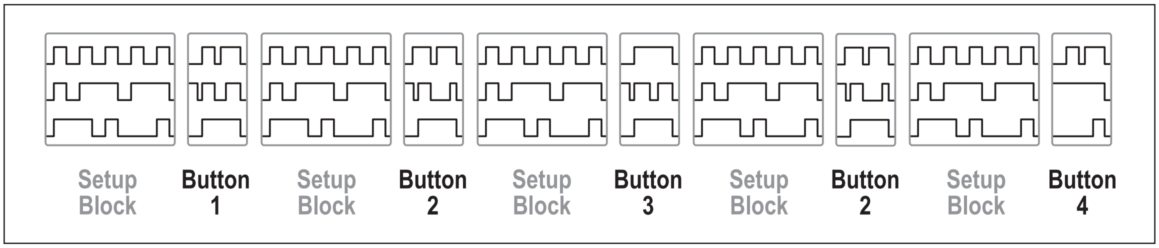


Figure 1: Interleaved Setup and Button blocks.

Because block sequencing provides effectively unlimited pattern depth. When it is necessary to run tests over a period of hours or even days (possibly while varying voltages or temperatures, for instance) the pattern generator sequencer is the only way to deliver the billions of operational cycles needed for the job.

The key to this capability resides in the TLA7PG2's sequencing instruction memory. This is a separate and independent memory that is used to store up to 4000 commands to control the pattern memory. Note again, the pattern data and the sequence instructions are stored in two different memories – not just partitions of the same contiguous RAM. This is important because it means you don't have to trade off sequence space as patterns grow longer or more numerous.

The sequencer is designed for ease of use and efficiency. Fundamentally, it allows you to choose which pattern blocks are used, in what order, and how many times. The sequencer implements a basic command set that makes it easy to multiply a few lines of pattern data into an exhaustive test with thousands or millions of operational cycles (see Figure 2).

Multiplying Nanoseconds to Create Longer Delays

Going back to our push-button example, it's easy to use commands such as Repeat to produce a realistic stimulus stream for the SUT. For example, the programmable Repeat count "pushes" any button multiple times (up to 65,536!), allowing you to capture responses to the button action throughout the SUT system. Equally important, the Repeat command is useful for setting up delay times between button pushes. By creating a "delay block" with no activity and repeating that block to accrue time, it's possible to generate delays that mimic real human reaction times. To illustrate the concept, let's assume we want a 200 ms delay between button pushes:

Required:
200 ms delay to simulate physical button actions

Given:
Minimum Block size = 40 vectors
Cycle speed = 10 MHz (100 ns cycle time)

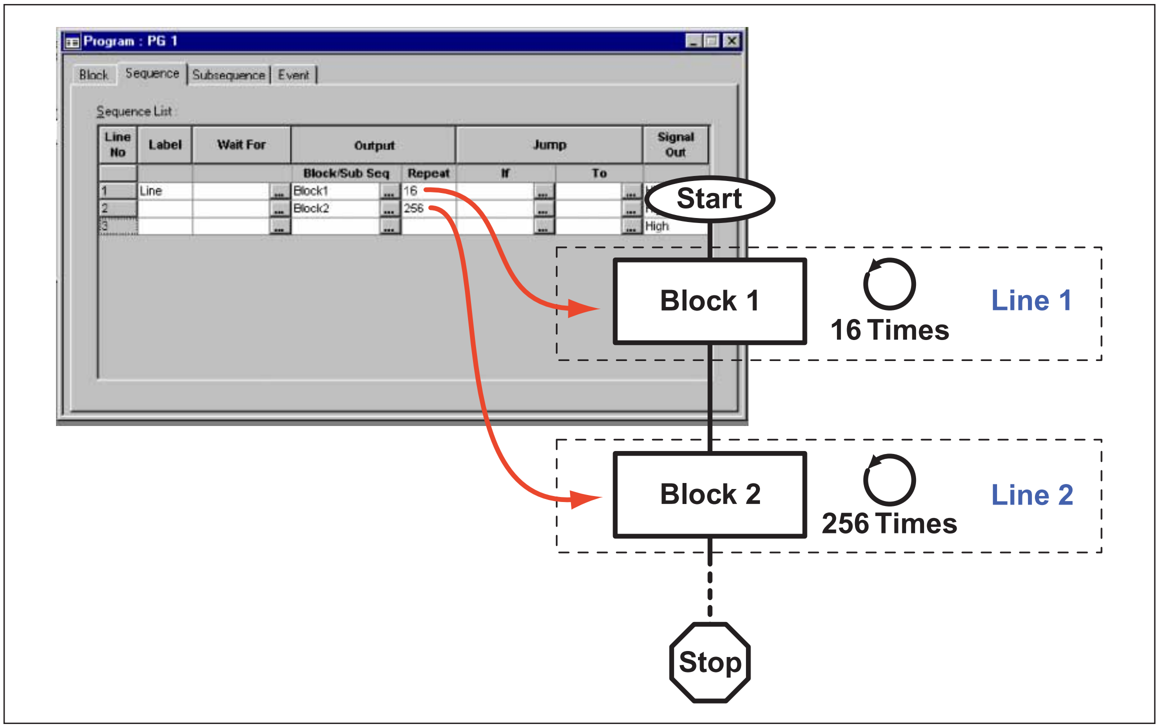
Delay Block:
40 vectors x 100 ns = 4000 ns = 4 µs

Repeat Delay Block 50,000 times:
50,000 x 4 µs = 200 ms

In addition to explicit commands such as Repeat, the TLA7PG2 can respond to external asynchronous interrupts from the SUT and the measurement system. These events make it possible to interact with SUT and TLA700 system activity in real time.

Sequencer Monitors Events to Respond to SUT and System Activities

The pattern generator offers a robust real-time event recognition capability that allows it to interact with the SUT to exercise buses and recognize state transitions. It can be programmed to respond to multiple events. When a particular event occurs, the sequence jumps to a predetermined sequence block.



Line No	Label	Wait For	Output		Jump		Signal Out
			Block/Sub Seq	Repeat	If	To	
1	Line		Block1	16			H
2			Block2	256			H
3							High

Figure 2: A few lines of sequence code produce many cycles worth of pattern vectors.

Many stimulus applications require some synchronization between the pattern generator and the SUT. For example:

- Diverting to an alternate pattern block when an exception state occurs in the SUT
- Holding the pattern output until the SUT reaches a specified state, then continuing the pattern sequence
- Synchronizing a drive-inhibit function to coordinate input and output operations on bidirectional or multi-source buses

The TLA7PG2 has eight signal inputs at the probe interface. Also, there's a system signal input that can be derived from the measurement modules in the TLA700 system. These nine signals can be sensed concurrently to detect a desired Boolean combination. Such combinations are known as "events," and up to 256 different events can be defined, as well as various AND and OR combinations of the signal inputs. Events can be named to associate their occurrence with an SUT function, state, or action. The advanced tracking capability of the TLA 700 measurement modules can detect complex error conditions and SUT state sequences which can be sent to the pattern generator as events.

To illustrate the use of events, imagine a test that verifies design margins for, say, setup times on another part of the system we tested earlier for button responses. One of the Q-outputs from the SUT is connected to a probe signal input. The procedure reduces the setup time in a series of coarse steps toward a violation. The violation, when it finally occurs, causes the Q-output to switch, which is detected by the signal line and prompts an event named "Fail." Responding to the Fail event, the Jump To command sends the sequence to a routine (again, made up of concatenated blocks) that gradually increases the setup time (in a series of finer steps) to pinpoint the margin.

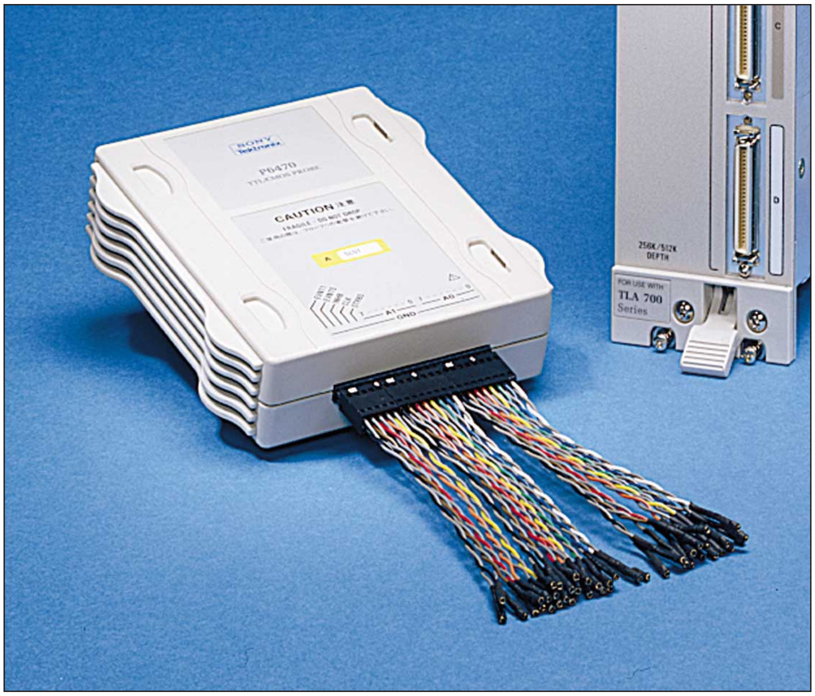


Figure 3: Photo of P6470 probe.

Alternatively, the TLA7PG2 could be programmed to jump to a troubleshooting routine. When an event occurs during block execution, it is latched until the sequencer reaches the end of the block. Then the sequencer jumps to a user-specified alternate pattern block, such as an error routine. An unconditional jump can be used at the end of the error routine to jump back to an entry point in the main pattern.

Note that jumps initiated by the pattern sequencer at block boundaries occur without any latency. This means that the block jumped to will drive valid data on the very next clock cycle, even at the maximum clock speed of 268 MHz!

The Pause/Wait For command also depends on event recognition. Often, the state machine sequencing of an SUT cannot be definitively predicted, and the time required to reach a particular state can vary. The WAIT FOR command solves this problem. At the end of any pattern block, the sequencer can be paused until it detects the status of the SUT indicating the desired state has been reached. As explained earlier, event definitions and signal inputs are used to detect the SUT status.

The signal inputs can help with Inhibit timing. In addition to probe-specific and programmable inhibit instructions in the pattern block, it's possible to set up an Inhibit function that occurs upon detection of a real-time event from the SUT. The Inhibit function (which drives the pattern generator probe outputs to a high-impedance state) is critical to preventing bus contention problems between the pattern generator and the SUT for bidirectional buses.

Programmable Drive Voltages Support Diverse Logic Families

Interfaces to many modules and systems today require multiple logic levels. It's common today to have systems under test that include CMOS logic, 5-volt TTL logic and 3.3-volt logic.

Imagine a digital module designed to be part of, say, a machine controller of some kind. The module's designer has many functions to implement: sensor signals to be buffered and monitored; control signals to be processed and bussed; instructions to be stored. In a perfect world, one logic family could handle all these chores.

In the world that most of us live in, though, there may be three or even four device families involved in the circuit's operation. The sensor buffers might be CMOS devices; the control logic and I/O circuitry might be a TTL-based family; the memory might be a low-power, low-voltage type for compatibility with backup batteries.

It all adds up to a nightmare for the designer trying to connect a pattern generator to such an SUT for functional validation purposes. Unless there is some alternative to hooking up a separate type of probe to each type of test point, the engineer will spend more time connecting than measuring! The process can become so cumbersome as to actually discourage thorough evaluation of the SUT.

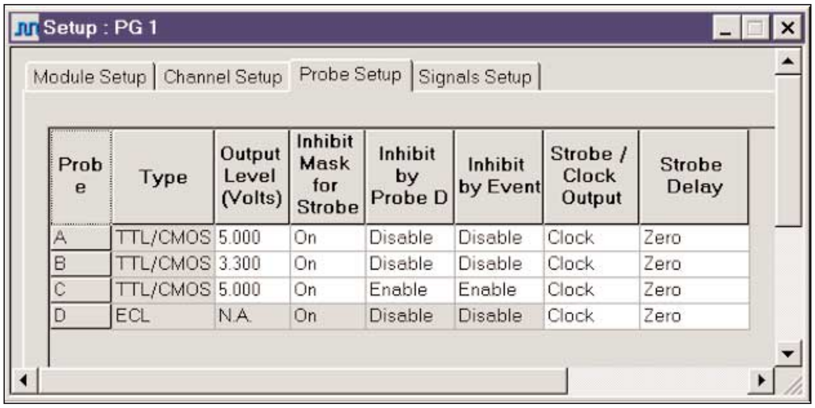
Fortunately, the TLA7PG2 has a solution. Its unique variable-voltage TTL/CMOS probe, the P6470 (see Figure 3), handles all three of these differing levels, with programmable “logic high” levels ranging from 2 volts to 5.5 volts. This simplifies the test setup and minimizes the need to reconfigure the test fixture when changing among different SUT types.

Moreover, the P6470’s variable voltage range also supports margin testing. By setting the drive levels close to the lower limit of the logic device and running test patterns with the TLA7PG2, it’s possible to verify the limits of system noise immunity. The voltage can be varied manually during a test to quickly determine the system margin.

An ECL probe, the P6471, is also available. It can be mixed freely on the TLA7PG2 with the TTL/CMOS probes. Between them, the P6470 and P6471 address all of the prevailing digital logic families (see Figure 4).

Putting the TLA7PG2 Pattern Generator to the Test

The best way to summarize the capability of the TLA7PG2 is to set it in the context of a realistic application. We will verify the real-time response of an embedded industrial controller. The test requirements for such an application are “typical” in that they pertain equally well to systems ranging from cockpit readouts to microwave ovens to telecommunications control cards for a wireless base station.



Prob e	Type	Output Level (Volts)	Inhibit Mask for Strobe	Inhibit by Probe D	Inhibit by Event	Strobe / Clock Output	Strobe Delay
A	TTL/CMOS	5.000	On	Disable	Disable	Clock	Zero
B	TTL/CMOS	3.300	On	Disable	Disable	Clock	Zero
C	TTL/CMOS	5.000	On	Enable	Enable	Clock	Zero
D	ECL	N.A.	On	Disable	Disable	Clock	Zero

Figure 4: Voltage setup screen showing various different probe types and logic levels.

The System Under Test

The system under test (SUT) is controlled by a single microprocessor and contains the following elements (see Figure 5):

- A set of static inputs set to one value for the entire test
- A bidirectional bus used to exchange data with other control points

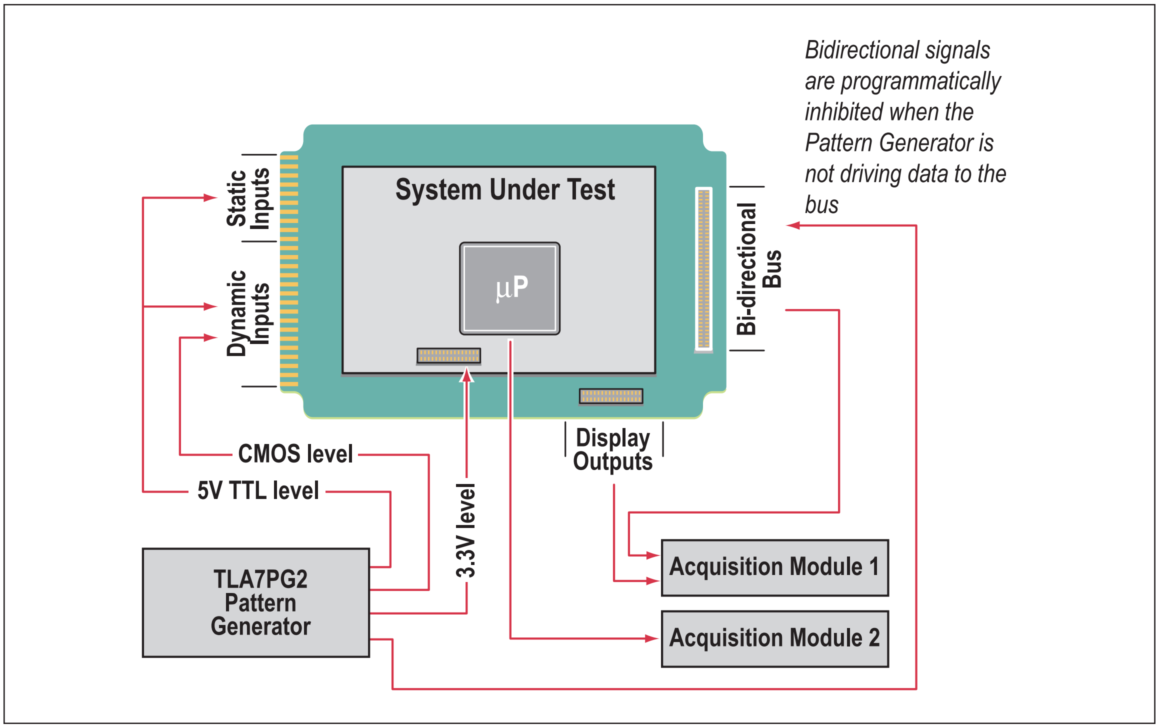


Figure 5: Connection layout for the system-under-test.

- A set of dynamic inputs changed throughout the test to exercise the software. Some of these inputs are interrupt-driven, and in our example, the response to interrupts is time-critical
- A set of dynamic outputs that sends the drive display signals and status information to the rest of the system

Because the external system design is not yet complete, its functions will have to be simulated. In addition, signals internal to the SUT will require some 3.3 V drive levels as well as a 5 V TTL and CMOS levels for the external board inputs.

Overview of the Test Plan

The test has two overall goals:

1. To validate the program logic with a variety of system inputs whose precise order and timing would be difficult to generate in the actual system.
2. To verify the real-time response to unique bursts of interrupt data designed to stress the system.

Since the SUT has its own control element (the microprocessor), it must first be initialized. The static signals and initial dynamic signals are applied and the clock and other "heartbeat" signals are repeated until the system signals a "ready" state. A series of tests are then executed to confirm the system's state transitions (as confirmed by bus output data, display outputs, and a software execution trace sampled by logic analyzer modules) are correct for the set of dynamic inputs.

The SUT's state at the end of each portion of the state transition test is not absolutely predictable. It may be in a critical section of code, where it might be sensitive to a series of real-time interrupts. Therefore a "burst" of interrupt data is sent to test whether the SUT can respond in time without corrupting the internal variables of the system. At the end of this burst test, and after the interrupt processing has been completed, the state transition testing is resumed. The test continues until a specific number of cycles are completed or a logic analyzer module detects an error.

Connecting the SUT and Starting the Test

The test calls for both pattern generation and acquisition capability, therefore both the TLA7PG2 and logic analyzer modules such as the TLA7N4 will be required. Any bidirectional signals and all relevant output signals (both internal and external to the SUT) are probed by a logic analyzer module. A second logic analyzer module probes the SUT's processor. A processor support package is loaded on the logic analyzer module to allow software execution tracking.

The pattern generator is set to drive external signals at 5 V TTL and CMOS levels, as appropriate, and internal signals at 3.3 V TTL. The bidirectional bus is programmed to drive only during chosen output patterns and will be inhibited the remainder of the time.

Initialization Blocks

Initialization consists of two pattern blocks (see Figure 6). The first block sets the static inputs and the initial dynamic inputs, as well as any control signals necessary to set those values.

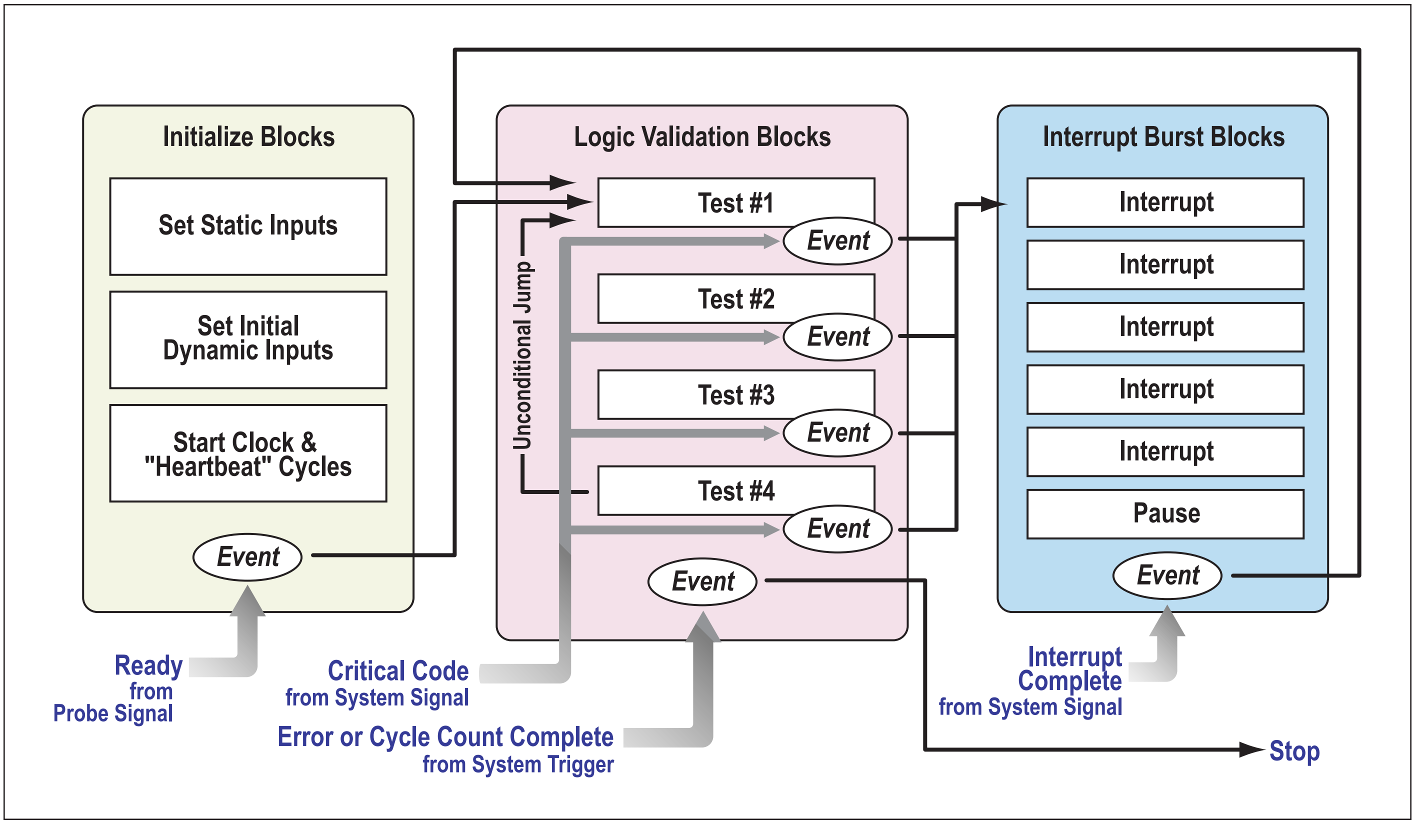


Figure 6: Test flow.

Pattern Generator Must Keep Pace With SUT Status

To simulate real-world stresses on SUT hardware and software, the pattern generator is required to carry out sophisticated interactions with the SUT. This is a challenge that can only be met if the test platform includes two key features:

Context-sensitive event recognition

Context-sensitive event recognition enables the pattern generator to tailor its response to signals from the SUT or the logic analyzer system. For example, the pattern generator's program may contain several alternative Jump destinations; the correct Jump during a given cycle depends on the current SUT state or condition.

Our earlier examples illustrate the concept. When the SUT reaches a section of critical code, we want the pattern generator to initiate interrupt stress testing; when the SUT has finished servicing interrupts, we want to resume normal test conditions. In both instances, the pattern generator monitors the same system signal to prompt the action. Detecting the difference between these circumstances and reacting accordingly is a key capability of the TLA700 stimulus and acquisition system.

Real-time communication

Context-sensitive event recognition is meaningless without real-time, "on the fly" response to the event in question. And this response isn't possible unless the test system allows real-time communication between its components. The unique signal communications bus between modules in the TLA 700 systems allows logic analyzers to control pattern generators and vice-versa. Test systems that use simplistic start-and-stop linkages cannot provide the real-time communications required for stress testing of today's complex systems.

The second block maintains these input states and cycles the "heartbeat" signals. This repeats infinitely with a test for a valid "Ready" event at the end of each cycle. The Ready event is sensed using a probe signal input. When the sequencer detects the Ready event, it advances to the logic validation tests.

Logic Validation Blocks

The logic validation consists of blocks defined for specific button pushes or signal pulses (see Figure 6). The bidirectional bus is driven with a set of blocks that apply valid data, exercise the appropriate control lines, and then inhibit the pattern generator's drive. The validation test is segmented into a series of independent tests that exercise specific functions of the SUT. The logic analyzer trigger machine is set up to watch for, and act on, any of four types of events during the functional validation test:

1) Detect when the software is in a critical section of code and notify the pattern generator through a system signal line.
2) Detect when the interrupt processing has been completed and notify the pattern generator through the same system signal line.
3) Detect when a system corruption occurs through a stack overrun or an improper exception vector, and capture the data surrounding the error.
4) Count the number of completed cycles and stop the system when it reaches the desired count.

At the end of each validation test segment, the pattern generator sequencer checks for the system signal that indicates a critical code section. If it detects the signal, it jumps to a set of functional pattern blocks that send a burst of interrupt data to the SUT. When the interrupt burst is complete, the pattern generator again pauses until it senses the system signal indicating that the interrupt service routines are complete and the SUT is ready to resume normal operation. Then the pattern generator returns to the logic validation test flow.

A second probe event signal monitors the system trigger line of the TLA 700. At the end of each repetition of the logic validation tests, this event is examined to determine whether the logic analyzer has signaled an end to the test, a result of reaching a specified cycles count, or has encountered a fatal error. If not, an unconditional jump sends the sequence back to the first logic validation test block.

For subsequent tests that require a different order of inputs, the sequencer can be reprogrammed without having to modify the detailed vectors within the blocks.

Summary

The TLA7PG2 Pattern Generator extends the functionality of the TLA 700 Series Logic Analyzers by producing essentially endless pattern variations at-speed, matching the needs of today's fast, complex microprocessor-based systems. The TLA7PG2 is an integrated, easy-to-use solution for functional validation chores. It is equally able to exercise a complex bus or substitute for an as-yet-unavailable circuit element.

For further information, contact Tektronix:

Worldwide Web: for the most up-to-date product information visit our web site at: www.tektronix.com/LA

ASEAN Countries (65) 356-3900; Australia & New Zealand 61 (2) 9888-0100; Austria, Central Eastern Europe, Greece, Turkey, Malta,& Cyprus +43 2236 8092 0;
Belgium +32 (2) 715 89 70; Brazil and South America 55 (11) 3741-8360; Canada 1 (800) 661-5625; Denmark +45 (44) 850 700; Finland +358 (9) 4783 400;
France & North Africa +33 1 69 86 81 81; Germany + 49 (221) 94 77 400; Hong Kong (852) 2585-6688; India (91) 80-2275577; Italy +39 (2) 25086 501;
Japan (Sony/Tektronix Corporation) 81 (3) 3448-3111; Mexico, Central America, & Caribbean 52 (5) 666-6333; The Netherlands +31 23 56 95555; Norway +44 22 07 07 00;
People's Republic of China 86 (10) 6235 1230; Republic of Korea 82 (2) 528-5299; South Africa (27 11)651-5222; Spain & Portugal +34 91 372 6000;
Sweden +46 8 477 65 00; Switzerland +41 (41) 729 36 40; Taiwan 886 (2) 2722-9622; United Kingdom & Eire +44 (0)1344 392000; USA 1 (800) 426-2200.

ISO 9001

From other areas, contact: Tektronix, Inc. Export Sales, P.O. Box 500, M/S 50-255, Beaverton, Oregon 97077-0001, USA 1 (503) 627-1916.

Copyright © 2000, Tektronix, Inc. All rights reserved. Tektronix products are covered by U.S. and foreign patents, issued and pending.
Information in this publication supersedes that in all previously published material. Specification and price change privileges reserved.
TEKTRONIX and TEK are registered trademarks of Tektronix, Inc. All other trade names referenced are the service marks,
trademarks or registered trademarks of their respective companies.

Tektronix

0600 TD/XBS 52W-13770-0