A Testing Methodology to Assure that Requirements Met by an eTPU Solution are Met by a GTM Solution

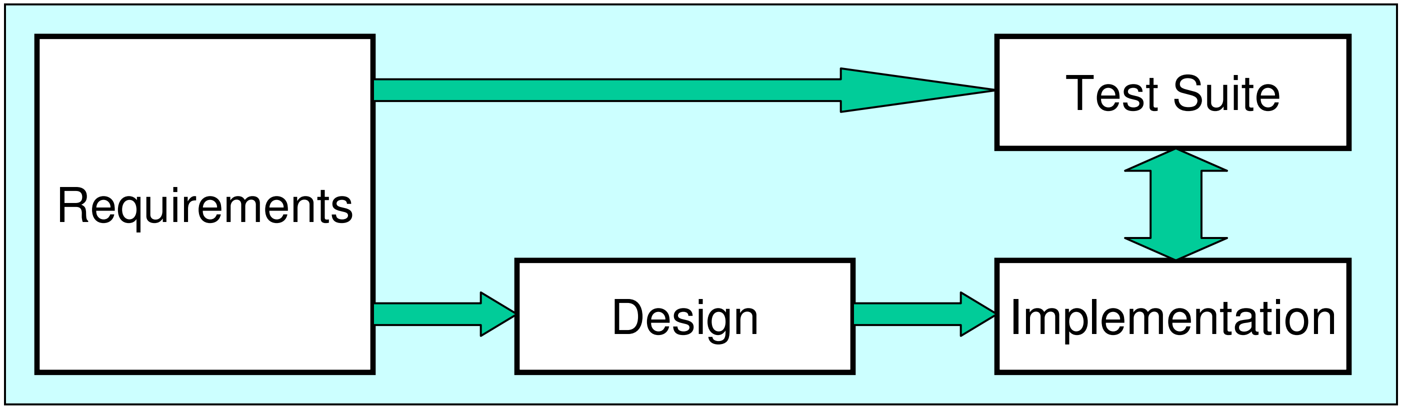
Andy Klumpp, ASH WARE
Mark Dlugoszewski, Renesas
John Diener, ASH WARE

Overview

- Both 'Black Box' and 'White Box' testing will be employed
  - White Box testing will validate that for a given input tooth signal, a 'ToothIndex' variable properly identifies and represents the incoming tooth pattern
  - Black Box testing will validate that for a given input signal a resulting output signal will result
- The ASH WARE DevTool Simulator toolset will be used to test both eTPU and GTM implementations
- Both implementations will be tested using the same methodology. In fact, identical 'Script' and 'Behavior Verification' files will be used to test both.
- Finally, an automated regression test that proves that the same requirements are met by both will be shown

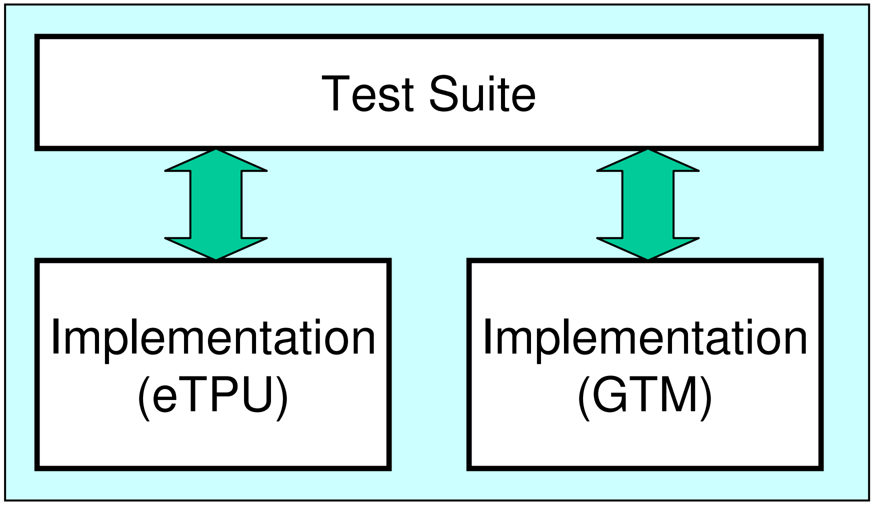
A Typical Development Process

- The following is a typical requirements-based testing methodology.
  - A set of requirements is developed.
  - The design is based on the requirements and the implementation is based on the design.
  - A test suite is developed from the requirements which is applied to the implementation.
  - When the test suite passes, then the implementation meets all the requirements.



The Test Suite

- In many cases a test suite already exists
- The key is to apply the existing test suite (or develop a ‘dual-purpose’ test suite) that proves that the same requirements are also met by the GTM implementation

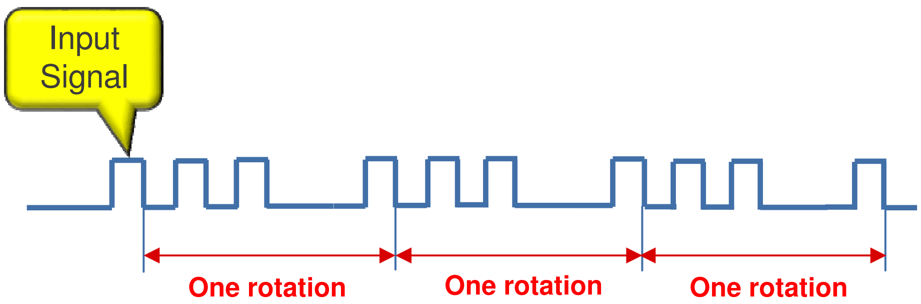


EXAMPLE: 4X1 ROTATION MONITORING

REQUIREMENTS

Input signal processing

- Calculated variable ‘Period’ will contain the micro-seconds per period and term will be micro-seconds (usec)
- The period will vary between **3000us** (milliseconds) and **60us**
- The maximum accelerate / decelerate rate +/- 50% per revolution
- Simple gap detection (synch while not accelerating or decelerating) to make it easy for 30 minute presentation
- These requirements will be validated using ‘**Data Flow Verification**’ (‘White Box’)



Rotation Monitoring Requirements

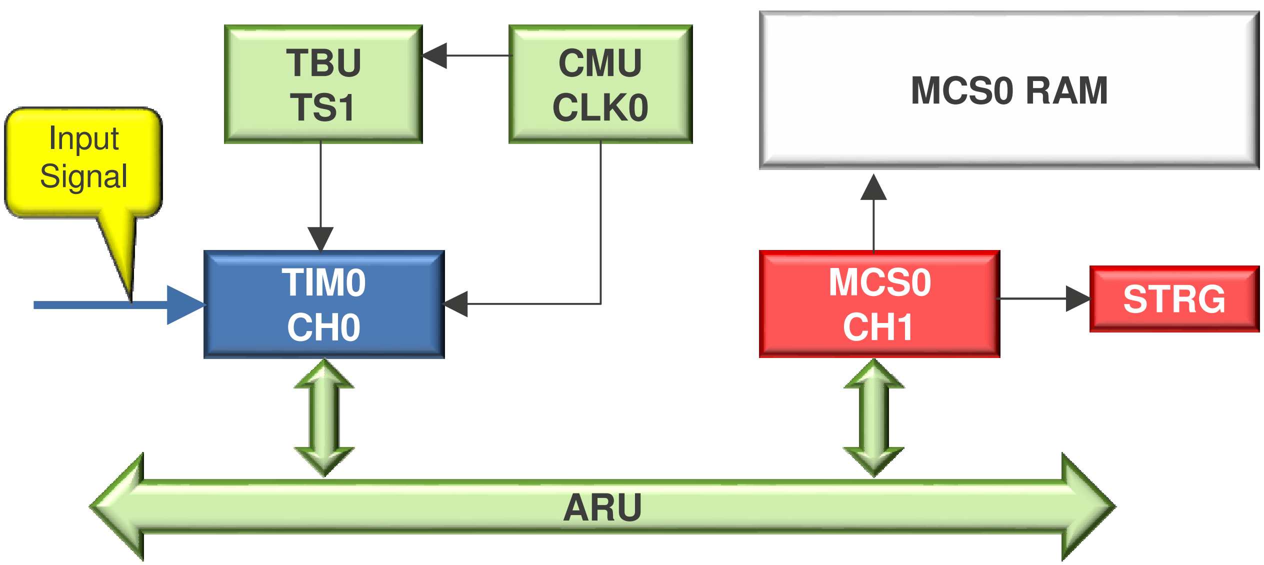
Tooth Index

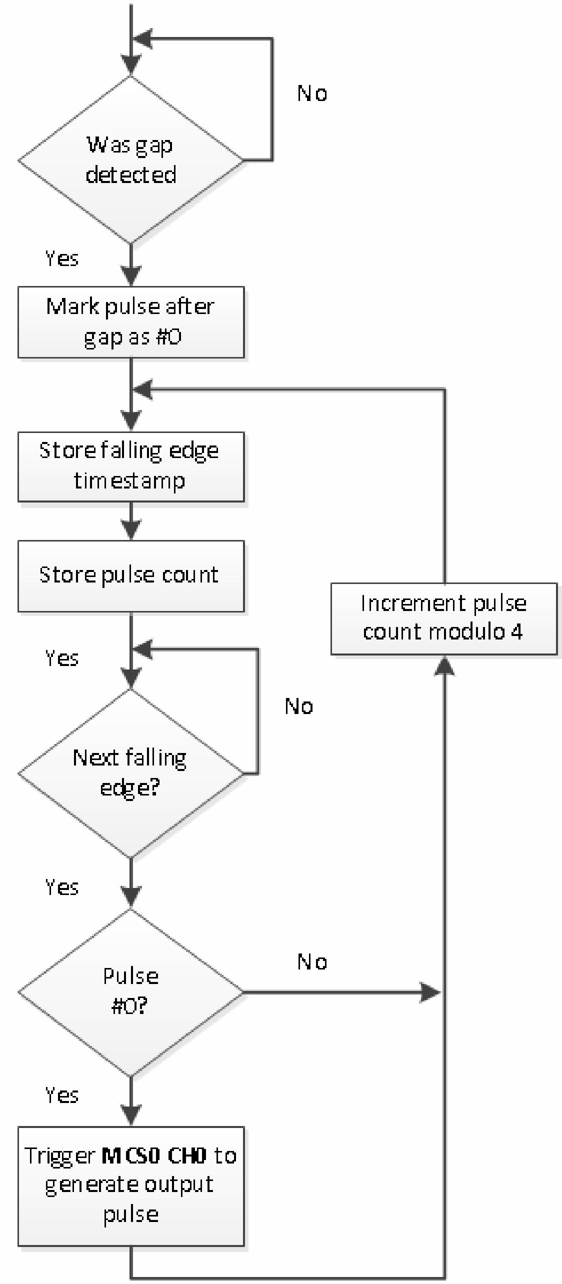
EXAMPLE: 4X1 ROTATION MONITORING

GTM IMPLEMENTATION

Input signal processing

- **TIM0 CH0** TIEM, ARU enable, GPR0=TBU_TS1, GPR1=ECNT
- **MCS0 CH1** accelerated mode
- **TBU_TS1** free running w/ CMU_CLK0 @80MHz





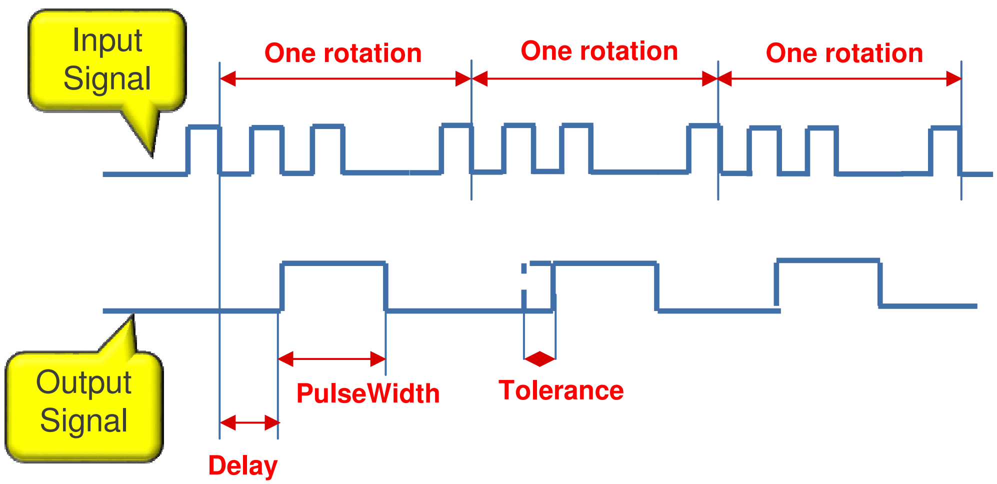
Input Signal Testing Methodology

- Drive an input signal within a script commands file
- The signal will have an acceleration/deceleration curve as follows
  - Accelerate from 3000 us/period 60 us/period to accelerating at ~50%/period
  - Decelerate back to 3000 us /period decelerating at ~50% per period
- 5 micro-seconds after every tooth the tooth number index will be verified

EXAMPLE: OUTPUT PULSE GENERATION
REQUIREMENTS

Output signal generation

- The terms for command variables ‘**Delay**’ and ‘**PulseWidth**’ will be in micro-seconds (usec)
- The following requirements will be verified using ‘**Pin Transition Verification**’ (‘Black Box’)



Rotation Monitoring Requirements		
	Min	Max
Delay Tolerance (usec)	-2.1usec	2.1usec
PulseWidth Tolerance (usec)	-2.1usec	2.1usec

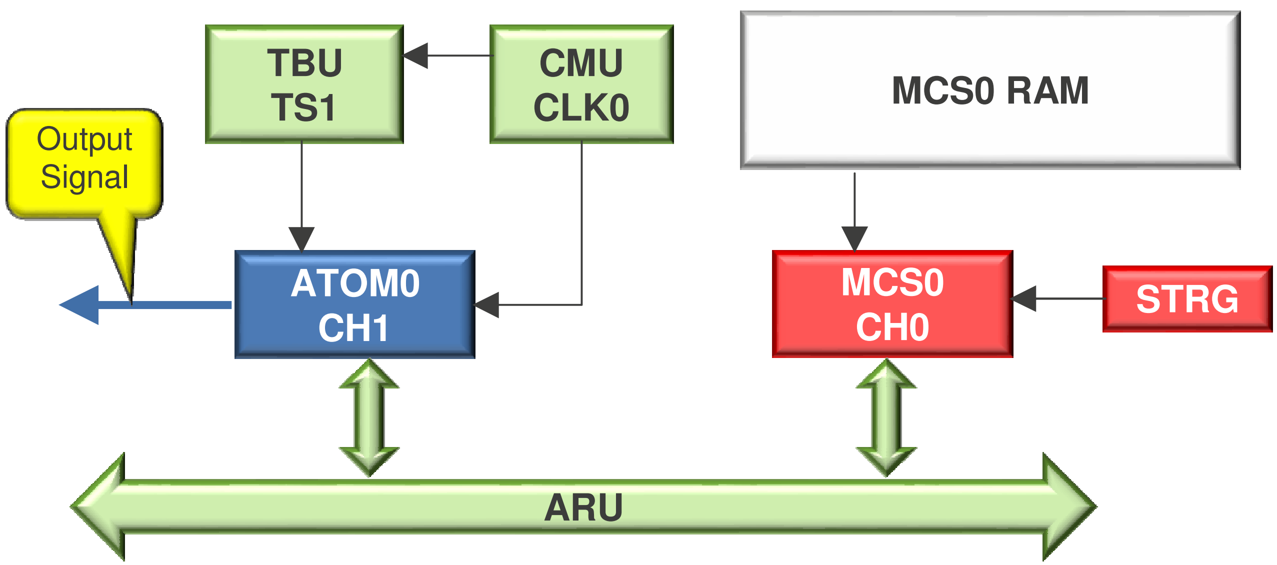
WWW.ASHWARE.COM

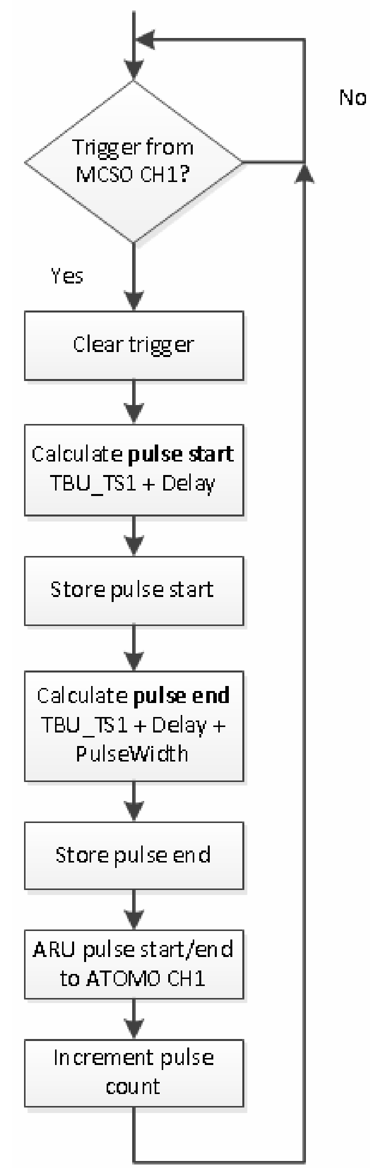
EXAMPLE: OUTPUT PULSE GENERATION

GTM IMPLEMENTATION

Output signal generation

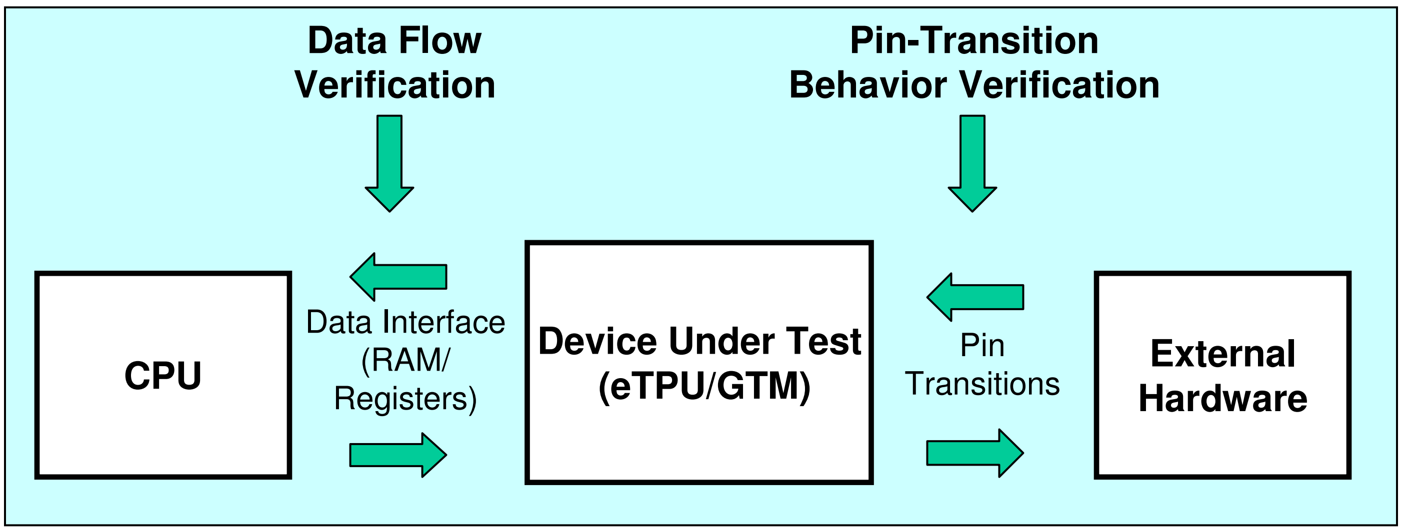
- **ATOM0 CH1** SOMB serve last w/ TBU_TS1
- **MCS0 CH0** accelerated mode, w/ STRG trigger
- **TBU_TS1** free running w/ CMU_CLK0 @80MHz





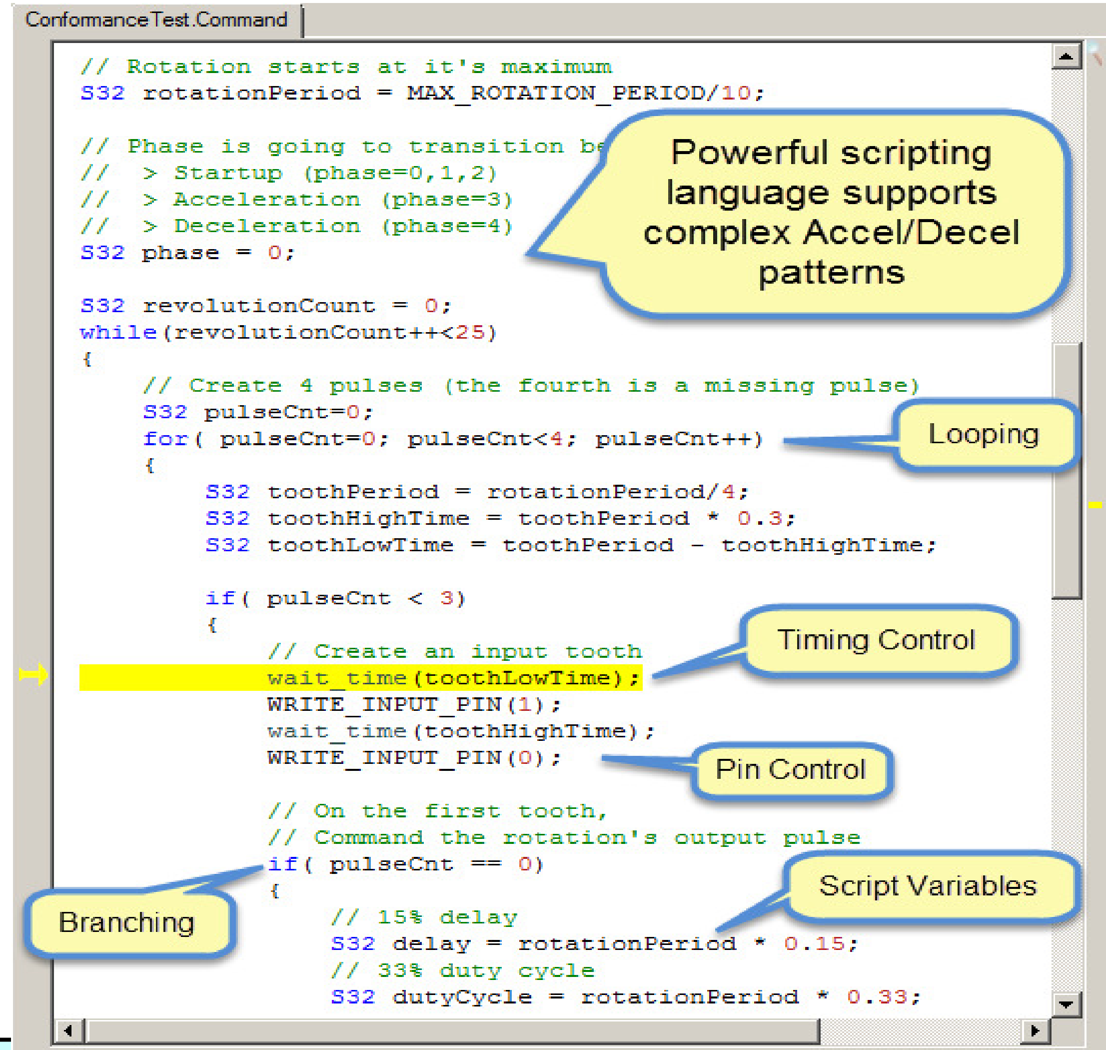
Data Flow and Output Signal Testing

- Data flow testing verifies a series of values over time in a simulation run. A simple acceleration/deceleration example consisting of crank example will be presented in which the measured tooth indices are validated over a series of time points
- Pin behavior files are a recording of output pin transitions and times. They can be a result of a combination of input pin and data-driven stimuli over time. A methodology for applying the same data and input pin stimuli to both a GTM and an eTPU system along with conformance to the same pin behavior file will be shown.
  - A tolerance band will be used to handle skew.



The Script Commands File

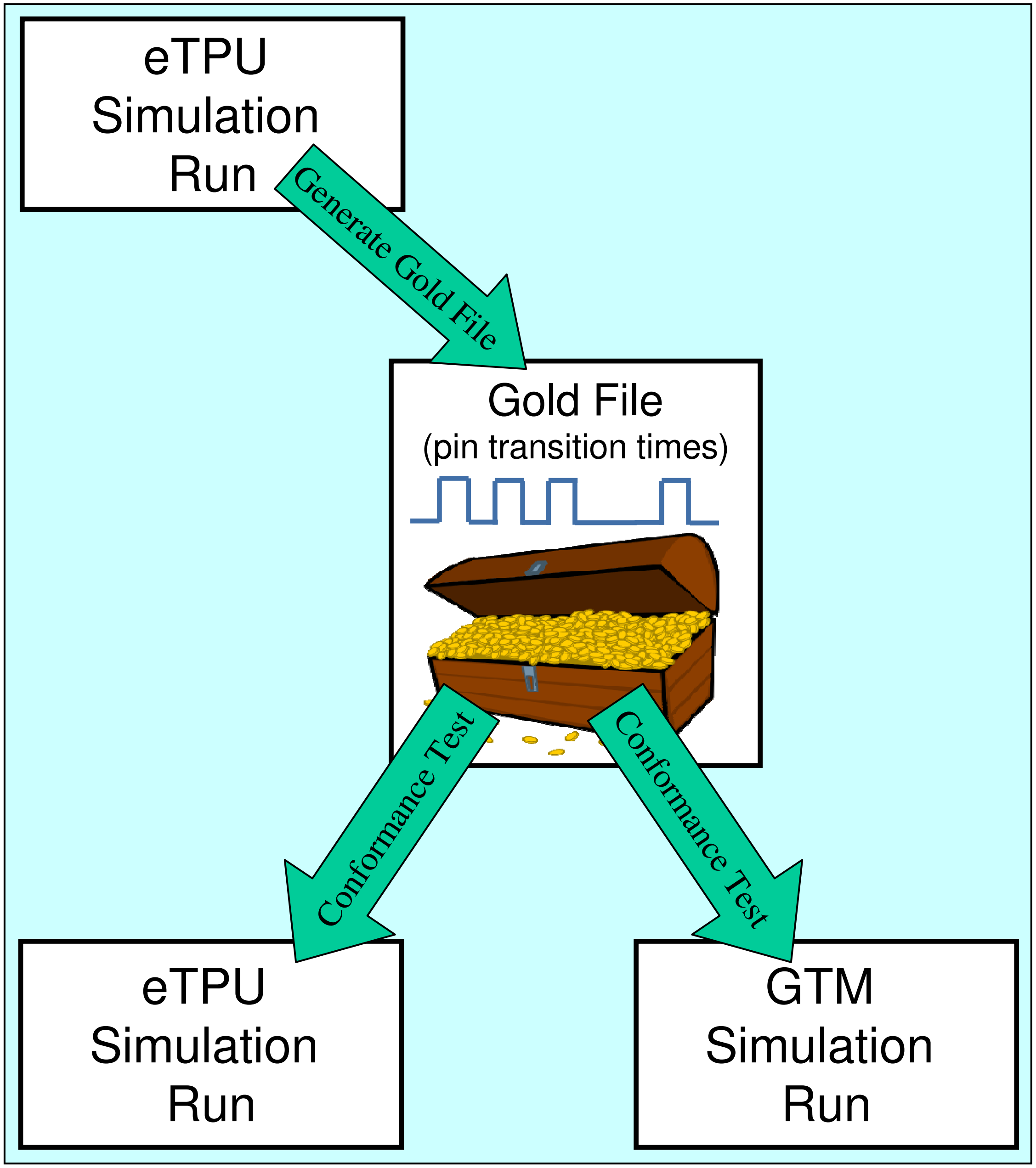
- A common script file is used to exercise (the tests) on both the eTPU Simulator and the GTM Simulator.
- Platform Hardware differences are handled using #ifdefs.
  - Example … handling of writing input pins and testing variable values is shown below.




```
#ifdef TEST_GTM
#define WRITE_INPUT_PIN(val) GTM.write_input_pin(TIM0, CH1, val).
#define VERIFY_TOOTH_INDEX(index) \
     verify_mem_u24(GTM_RAM_SPACE, _AWGTM_VAR_ADDR_currentToothIndex_+1, 0xFFFFFF, index);
#else // TEST ETPU
#define WRITE_INPUT_PIN(val) write_chan_input_pin(INPUT_DETECT_CHAN, val );
#define VERIFY_TOOTH_INDEX(index)  \
    verify_chan_data24(INPUT_DETECT_CHAN, _CPBA24_InputDetect__currentToothIndex_, index);
#endif
```

Output Signal Testing Methodology

- A 'Gold' file is generated that contains all the recorded pin transitions
- In an automated regression testing environment, the same gold file and script command file will be used to test both the GTM as well as the eTPU solution



Automation: **Snippets From File 'Test.Bat'**

```
Rem File: 'Test.bat'

echo .
echo Start by deleting the Enhanced Behavior Verification 'GOLD' file
del GoldPinTransitions.ebv
echo .


echo .
echo Create a new GOLD file 'GoldPinTransitions.ebv' from the eTPU
%SIM% -p=Project_ETpu.ETpuIdeProj -AutoBuild -AutoRun -d=_CREATE_EBV_FILE_ -d=TEST_ETPU
if  %ERRORLEVEL% NEQ 0 ( goto errors )
if not exist GoldPinTransitions.ebv (goto errors)
echo .


echo .
echo Test the GOLD file 'GoldPinTransitions.ebv' in the GTM
%SIM% -p=Project_Gtm.GtmIdeProj -AutoBuild -AutoRun  -d=TEST_GTM
if  %ERRORLEVEL% NEQ 0 ( goto errors )
if not exist GoldPinTransitions.ebv (goto errors)
echo .
```

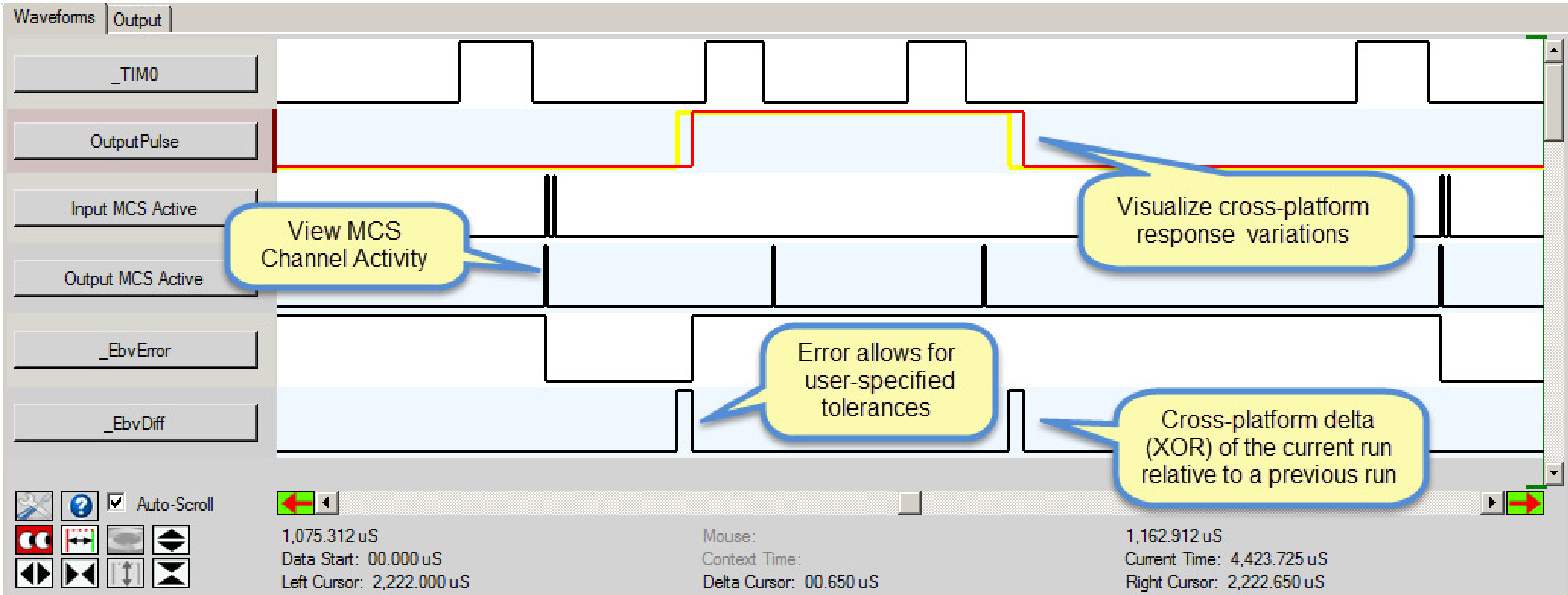
Initially …
Delete any previously-created pin-transition file

Next, …
An eTPU simulation run creates the pin-transition file

Finally …
Run the conformance test in the GTM simulator using the pin transition file created by the eTPU simulator

Debugging and Handling Skew

- The ASH WARE GTM Simulator provides a number of visualizations and other capabilities for analyzing and debugging



- A tolerance band can be specified in the script commands file for handling signal skews

```
// Apply allowed tolerance band to signal ‘OutputPulse’
set_ebehavior_pin_tolerance(“OutputPulse", EBV_ABSOLUTE, 0.0, 2.1);
```