HILEVEL Technology, Inc. Q'n Apps #E55

Q: How can I define ACPMU tests?

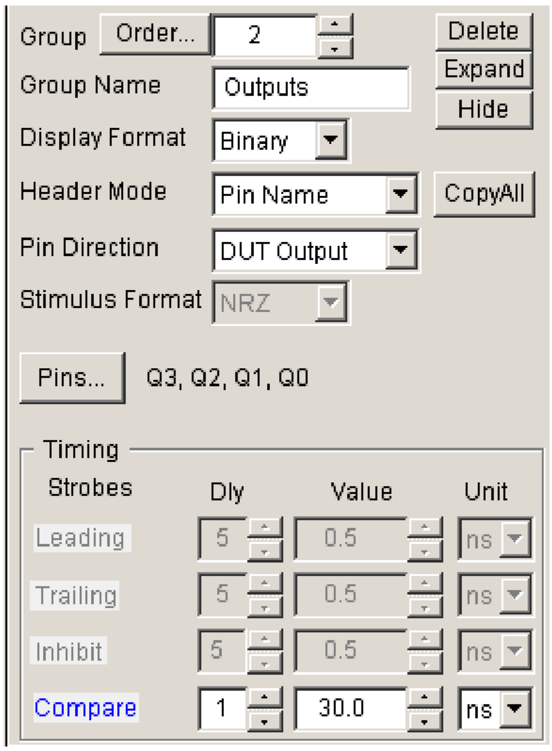
Timing Tests with AC Parametrics

ETS2k software can effectively perform timing measurements with the ACPMU feature, standard on every HiLevel logic test system. You can measure setup time, hold time, access time (or prop delay), and pulse width. You can also create a custom test to measure the time from one edge (timing generator) to another.

Before using the ACPMU it is important to understand a few things.

1. The ACPMU works by using one delay generator as a reference point and varying another delay generator to find the optimum measured interval. In our example to follow, one timing generator creates an event in time as a stimulus edge that is going to an input to the chip, while another timing generator creates a different event in time as it is used to compare the DUT output signal.
2. Proper use of the stimulus formats is crucial to ACPMU effectiveness. NRZ format is generally fine for many pins, but other pins such as clock and enables may need to be dynamic; that is, RZ or R1, so that trailing edges can be controlled for tests like Pulse Width.
3. The vectors play a key role in the operation:
   a. Measurements are made based on the pass/fail point of output pins, so you must have functional vectors that pass in order to use ACPMU.
   b. The entire vector set is executed and so your control of which transition is used for measurements is limited. You may need some special small vector sets for better control of the measurements.
   c. If you make the mistake of using NRZ for your clock with the clock pin toggling at every other vector (i.e., vector data is 1,0,1,0) then you will have to run your test rate at twice the desired test rate, hampering use of the ACPMU. Since ACPMU can only measure within one clock cycle, you will not be able to do most tests because your data requires two cycles in order to change.
4. ACPMU uses binary search to find the measurements, not a linear search. This makes it much faster while providing 100ps granularity in the final measurement.
5. The ACPMU searches for the transition point in the vector run where the output result goes from PASS to FAIL, or from FAIL to PASS. To successfully take the measurement, this Pass/Fail transition point must be within the defined Measurement Range. Defining the proper range enables the ACPMU to search for the first moment when the output data becomes valid (i.e., access time), or the last moment when output data becomes invalid (i.e., hold time).

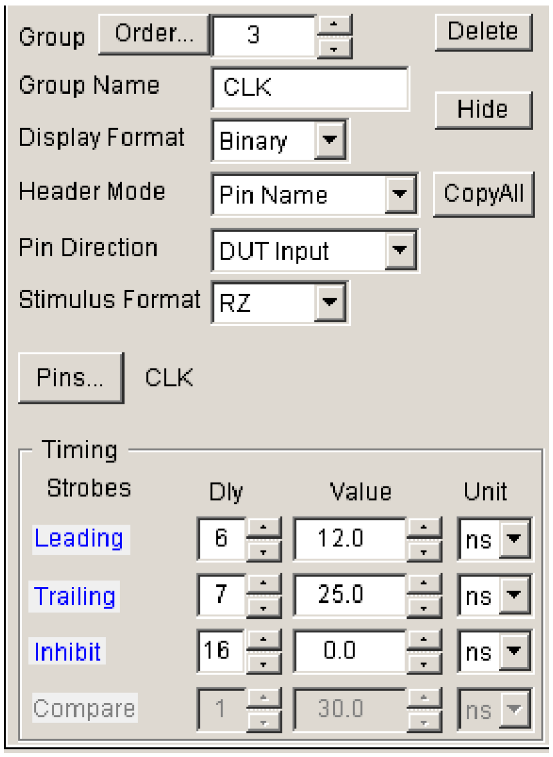
For this example, weôll be making a prop delay test on a 74F163 binary counter. The vectors are simple and easy to follow. Letôs just concentrate on the four output pins and the clock pin, since the goal is to find the interval from the leading edge of the clock to point where the output data becomes valid. Here are the definitions for the two pin groups.



Group: 2
Group Name: Outputs
Display Format: Binary
Header Mode: Pin Name
Pin Direction: DUT Output
Stimulus Format: NRZ
Pins: Q3, Q2, Q1, Q0

Timing

Strobes	Dly	Value	Unit
Leading	5	0.5	ns
Trailing	5	0.5	ns
Inhibit	5	0.5	ns
Compare	1	30.0	ns

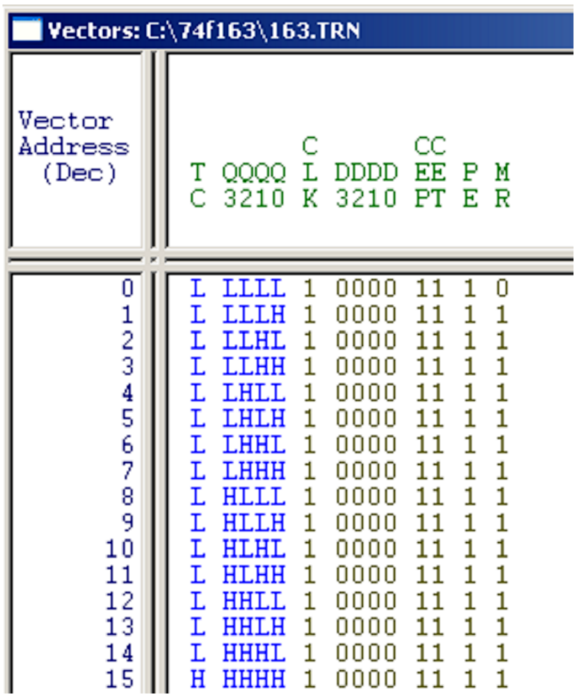


Group: 3
Group Name: CLK
Display Format: Binary
Header Mode: Pin Name
Pin Direction: DUT Input
Stimulus Format: RZ
Pins: CLK

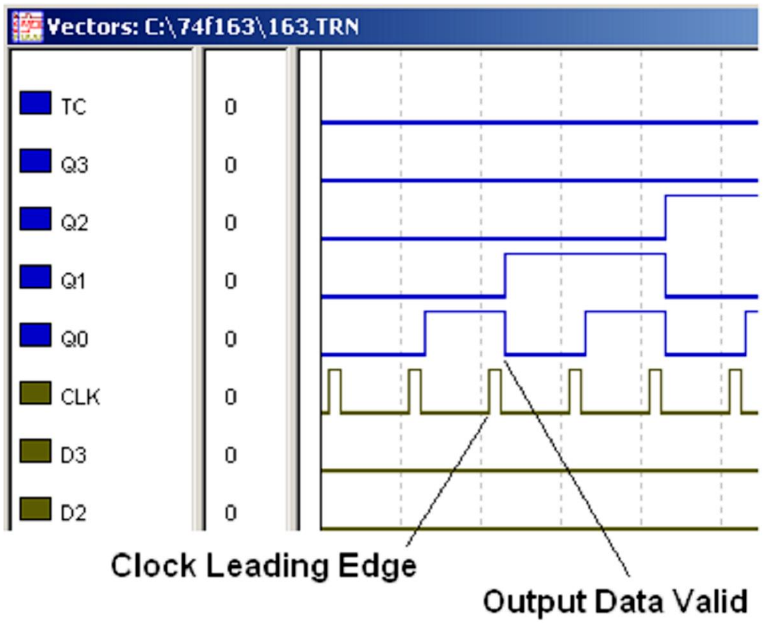
Timing

Strobes	Dly	Value	Unit
Leading	6	12.0	ns
Trailing	7	25.0	ns
Inhibit	16	0.0	ns
Compare	1	30.0	ns

The leading edge of the clock occurs at 12ns into each clock cycle. The outputs become valid at sometime later, so we strobe the output pins at 30ns to compare the actual output data against the expected response. Take a look at the vectors and at the vector graph:

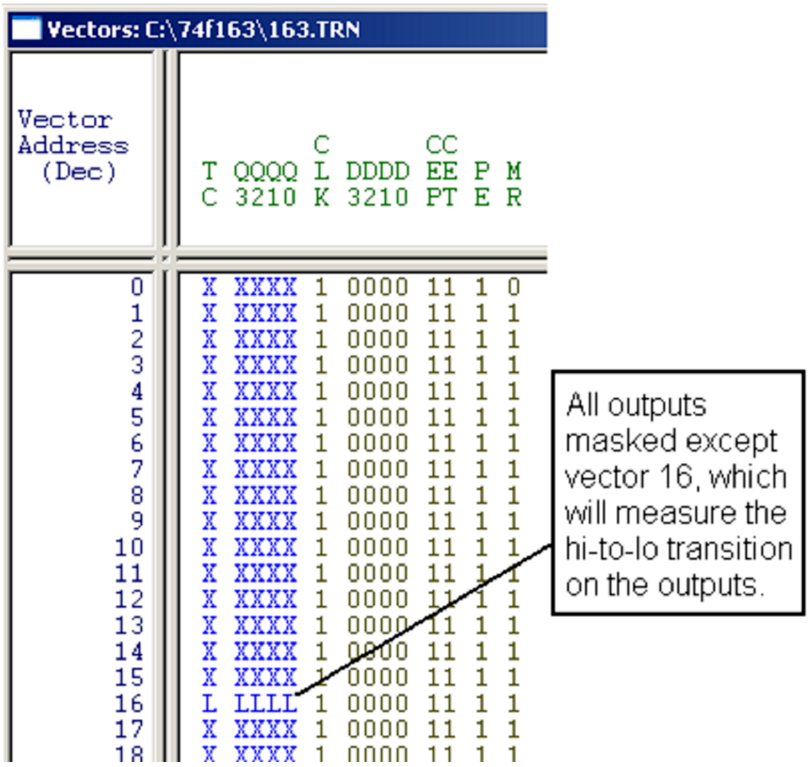


```
Vectors: C:\74f163\163.TRN

Vector
Address            C        CC
 (Dec)     T QQQQ  L  DDDD  EE  P  M
           C 3210  K  3210  PT  E  R

     0     L LLLL  1  0000  11  1  0
     1     L LLLH  1  0000  11  1  1
     2     L LLHL  1  0000  11  1  1
     3     L LLHH  1  0000  11  1  1
     4     L LHLL  1  0000  11  1  1
     5     L LHLH  1  0000  11  1  1
     6     L LHHL  1  0000  11  1  1
     7     L LHHH  1  0000  11  1  1
     8     L HLLL  1  0000  11  1  1
     9     L HLLH  1  0000  11  1  1
    10     L HLHL  1  0000  11  1  1
    11     L HLHH  1  0000  11  1  1
    12     L HHLL  1  0000  11  1  1
    13     L HHLH  1  0000  11  1  1
    14     L HHHL  1  0000  11  1  1
    15     H HHHH  1  0000  11  1  1
```



The 74F163 performs its functions on the leading edge of the clock. But some chips may also operate on the trailing edge, or both edges. Clearly the control of both leading and trailing edges on certain pins is crucial for timing measurements.

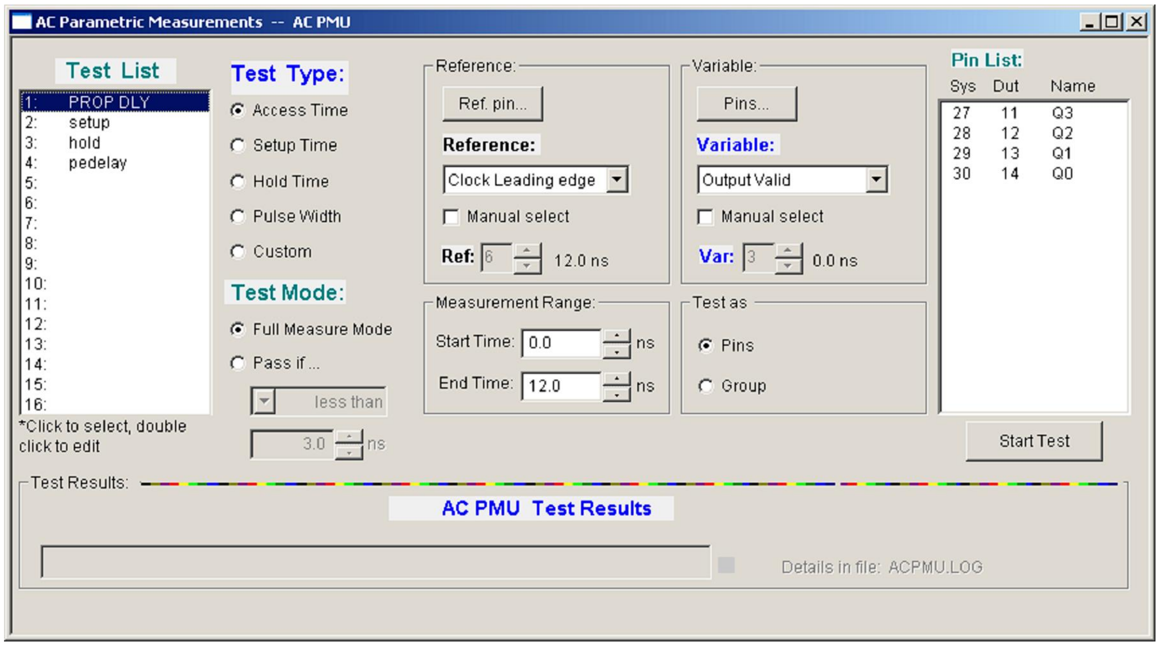
For more precise control of the exact transition you wish to measure, it is better to create some smaller vector sets that are subsets of your main vector file. In DCPMU, you select the vector address for the measurement, but in ACPMU, all of the vectors are run until Last Vector as defined in your Test Setup window. By masking all outputs except the transition you wish to measure, you have more control of your measurements. Here are the vectors shown above after masking:



```
Vectors: C:\74f163\163.TRN

Vector
Address            C        CC
 (Dec)     T QQQQ  L  DDDD  EE  P  M
           C 3210  K  3210  PT  E  R

     0     X XXXX  1  0000  11  1  0
     1     X XXXX  1  0000  11  1  1
     2     X XXXX  1  0000  11  1  1
     3     X XXXX  1  0000  11  1  1
     4     X XXXX  1  0000  11  1  1
     5     X XXXX  1  0000  11  1  1
     6     X XXXX  1  0000  11  1  1
     7     X XXXX  1  0000  11  1  1
     8     X XXXX  1  0000  11  1  1
     9     X XXXX  1  0000  11  1  1
    10     X XXXX  1  0000  11  1  1
    11     X XXXX  1  0000  11  1  1
    12     X XXXX  1  0000  11  1  1
    13     X XXXX  1  0000  11  1  1
    14     X XXXX  1  0000  11  1  1
    15     X XXXX  1  0000  11  1  1
    16     L LLLL  1  0000  11  1  1
    17     X XXXX  1  0000  11  1  1
    18     X XXXX  1  0000  11  1  1
```

All outputs masked except vector 16, which will measure the hi-to-lo transition on the outputs.

Now letôs look at the AC test definition for this test.


AC Parametric Measurements -- AC PMU

Test List:
1: PROP DLY
2: setup
3: hold
4: pedelay
5:
6:
7:
8:
9:
10:
11:
12:
13:
14:
15:
16:
*Click to select, double click to edit

Test Type: [✓ Access Time] Setup Time / Hold Time / Pulse Width / Custom

Test Mode: [✓ Full Measure Mode] Pass if ... less than 3.0 ns

Reference: Ref. pin... Reference: Clock Leading edge; Manual select ☐; Ref: 6 12.0 ns

Variable: Pins... Variable: Output Valid; Manual select ☐; Var: 3 0.0 ns

Measurement Range: Start Time: 0.0 ns; End Time: 12.0 ns

Test as: [✓ Pins] Group

Pin List:

Sys	Dut	Name
27	11	Q3
28	12	Q2
29	13	Q1
30	14	Q0

Start Test

Test Results: AC PMU Test Results

Details in file: ACPMU.LOG

The test type is ñAccess Timeò, but weôll call our test ñPROP DLYò; for our purposes it essentially means the same thing. When we click the ñRef. Piné ò button we select the clock pin, and the reference delay generator (6) is automatically selected. The currently assigned value for the leading edge is displayed.

For our variable, we have selected the output pins Q0-Q3 as displayed in the Pin List. ETS2k automatically selects an unused delay generator so that no other pins are affected, but you can click ñManual selectò and choose another delay if desired.

Next we set our measurement range. To determine the optimum range, we refer to the specifications for the 74F163, which states that the maximum allowed propagation delay is 10.0ns. We know that our device passes functional vectors with the output compare delay generator set at 30ns. Since our clock leading edge occurs at 12.0ns, this means that we are allowing 18ns for prop delay during functional test. So our actual prop delay is somewhere between 0ns and 18ns (18ns is actually too ñgenerousò for this chip). The device is considered ñbadò if the measured value is more than 10ns, so we will set our range at 0 to 12ns. The start and end times of the range are *not* relative to the ñtime zeroò of the test rate, but are relative to the leading edge of the clock pin.

Again, it is important that our Measurement Range is set so that the output of the device will pass at one end of the range and fail at the other end. In our case, 0ns will cause vectors to fail because it is too near to the clock transition. Surely it will be passing before it reaches 12ns. Negative values are allowed in the range setting as long as the range still fits within a single vector timeframe.

When we start the test ETS2k will repeatedly run the functional vectors while adjusting the compare strobe delay generator across the measurement range in a binary search algorithm. The results appear in the file ACPMU.LOG.

```
HILEVEL -- AC Parametric Measurement Unit -- RESULTS
-------------------------------------------------------------
Test: 1 Name: PROP DLY Test Type: Access Time.
(Pins are tested individually, and timing results & PASS/FAIL
indications are listed.)
REFERENCE PARAMETER: Clock Leading Edge @ 12.0 ns.
VARIABLE PARAMETER: Output Valid.
Range: from 0.0ns to 12.0ns.

---------------------------- RESULT(S) -------------------
SysChan   DutPin   PinName   Access Time
--------- -------- --------- -------------
27          11       Q3       4.0 ns      Passed!
28          12       Q2       3.6 ns      Passed!
29          13       Q1       3.6 ns      Passed!
30          14       Q0       3.9 ns      Passed!

-------------------------------------------------------------
Total Pins Tested: 4 Pin(s) Passed: 4 Failure(s): 0
=============================================================
```

Example of ACPMU Test Results file, ACPMU.LOG

Other tests in ACPMU

In the case of a Setup Time measurement, we measure for the minimum amount of time required for input data to be asserted before it is clocked in by the leading edge of the clock pin. The reference pin is the clock pin, and the variable is the leading edge of the data (data inputs should be defined as RZ for this test so that the trailing edge of the data does not move as a function of ñVariableò). For the type of variable you can select ñInput Transitionò. The range will be from the start of the Test Rate cycle (time zero) up to the leading edge of the clock pin. When the data is asserted ñtoo nearò to the clock pin leading edge, the vectors fail because there is not enough setup time before the clock. ACPMU will search the range between T_0 and CLK_{LE} to find the optimum setup time.

Hold Time refers to the amount of time we must ñholdò the input data valid *after* the clock edge (in the case of 74F163, leading clock edge). So our reference is still Clock Leading Edge and our Variable is Input Transition, but for the variable delay we select the delay generator that is being used for the trailing edge of our RZ data input pins. The measurement range must be set so that this trailing edge ranges from ñnormalò to the clock leading edge. ACPMU will report the minimum amount of hold time measured.

Pulse Width is really for determining the minimum width of your clock pulse that will allow the device to function. The reference is the clock leading edge and the variable is the clock trailing edge. Simply set meaningful ranges according to the device specification in order to find the minimum pulse width.

Also see section 15 of the User Manual.

Also See:
Q'nApp #E16: ETS2k Files
Q'nApp #E17: Stimulus formats
Q'nApp #E23: Inhibit delay
Q'nApp #E24: Data vs. Clock
Q'nApp #E26: Functional test preparation