패턴 지향 프로그램을 이용한 테스트 시간 감소

Test Time Reduction by Pattern Oriented Programming

오강훈, 이용화, 이명구
테라다인 코리아
e-mail 주소가 들어갑니다.

Abstract

This paper represent test time reduction method using pattern oriented programming (POP). POP means the pattern is the center of the test, it controls instrument setups, it performs measurements and all this is done with timing and accuracy typically associated with digital patterns.

I. INTRUDUCTION

Depending on your device type, what it's being compared to, and how the test program was written, we have seen some "huge" test time improvements. Obviously, the improvement will vary, later in the presentation we touch how to assess how applicable POP is to your device and the kind of improvement you could see.

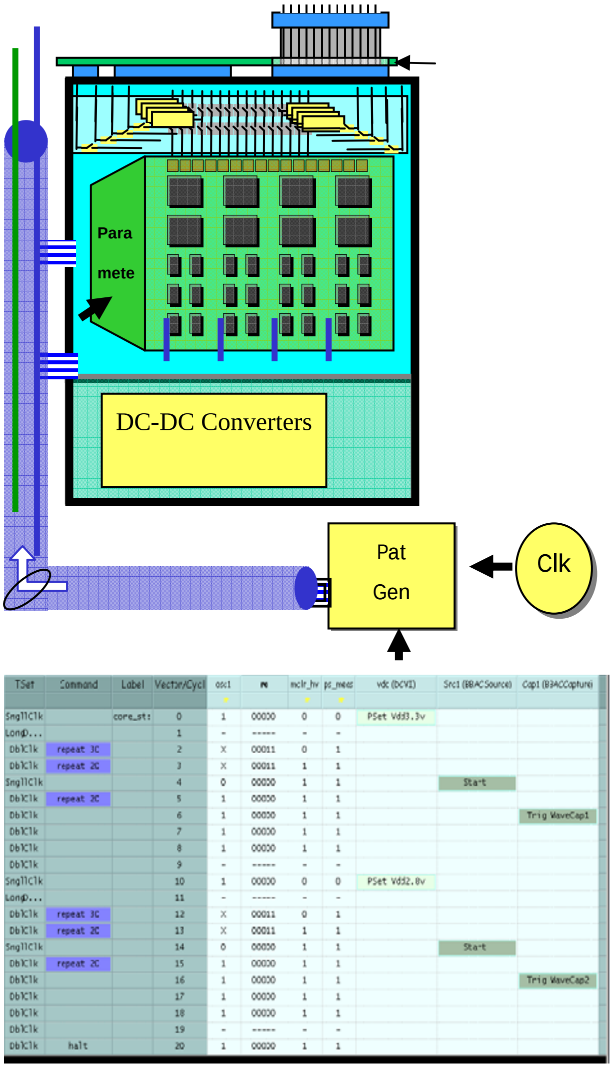
POP benefit is like blow.

1) Test Time
2) Complete Test Integration in Pattern
3) Measurement Stability
4) Production Maintenance

The pattern is where you efficiently setup multiple instruments at the same time across multiple sites. It's where you precisely control the timing between a setup opcode and a measurement. It's where you start on the next test while the instruments moves the captured data to embedded DSP's to perform analysis while the pattern continues to execute (perform the next test). It's where one set of patgen's performs one sequence of instruments while another set performs a different set of instructions to concurrently test different sections of the DUT.

II. TECHNICAL BACKGROUND

In past only digital and AC instruments could be controlled from the pattern. Now most instrument can be be setup (Pset's) and controlled (opcodes) from the pattern



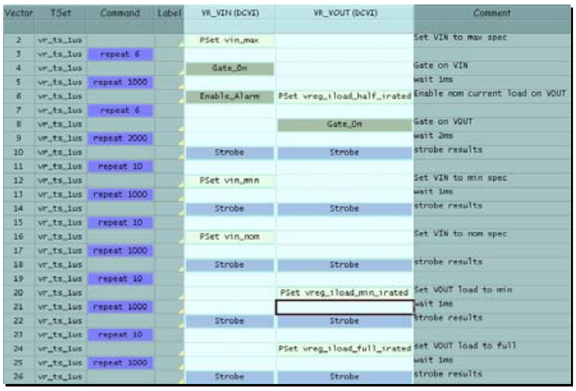
TSet	Command	Label	Vector/Cycl	osc1	[illegible]	mclr_hv	ps_meas	vdc (DCVI)	Src1 (BBACSource)	Cap1 (B3ACCapture)
SnglClk		core_st:	0	1	00000	0	0	PSet Vdd3.3v		
LongD...			1	-	-----	-	-			
DblClk	repeat 3C		2	X	00011	0	1			
DblClk	repeat 2C		3	X	00011	1	1			
SnglClk			4	0	00000	1	1		Start	
DblClk	repeat 2C		5	1	00000	1	1			
DblClk			6	1	00000	1	1			Trig WaveCap1
DblClk			7	1	00000	1	1			
DblClk			8	1	00000	1	1			
DblClk			9	-	-----	-	-			
SnglClk			10	1	00000	0	0	PSet Vdd2.8v		
LongD...			11	-	-----	-	-			
DblClk	repeat 3C		12	X	00011	0	1			
DblClk	repeat 2C		13	X	00011	1	1			
SnglClk			14	0	00000	1	1		Start	
DblClk	repeat 2C		15	1	00000	1	1			
DblClk			16	1	00000	1	1			Trig WaveCap2
DblClk			17	1	00000	1	1			
DblClk			18	1	00000	1	1			
DblClk			19	-	-----	-	-			
DblClk	halt		20	1	00000	1	1			

Fig.1. Pattern Control and Setup Instrument

Now let's look at real example of a controlling an instrument from the pattern using pset's and opcodes.

Voltage Regulator example of a POP test. The pattern shown performs a series of tests on the regulator by applying test conditions initially via Psets and measuring the output's response via Strobe opcode. The scope shot on left hand shows the input waveform to the DUT.

1) PSET VIN_MAX / Gate On
2) Strobe VOUT
3) PSET VIN_MIN
4) Strobe VOUT
5) PSET VIN_NOM
6) Strobe VOUT Min load on VOUT
7) Strobe VOUT Max load on VOUT
8) Strobe VOUT Nom load on VOUT
9) Start PSRR Waveform Source



Vector	TSet	Command	Label	VR_VIN (DCVI)	VR_VOUT (DCVI)	Comment
2	vr_ts_1us			PSet vin_max		Set VIN to max spec
3	vr_ts_1us	repeat 6				
4	vr_ts_1us			Gate_On		Gate on VIN
5	vr_ts_1us	repeat 1000				wait 1ms
6	vr_ts_1us			Enable_Alarm	PSet vreg_iload_half_irated	Enable nom current load on VOUT
7	vr_ts_1us	repeat 6				
8	vr_ts_1us				Gate_On	Gate on VOUT
9	vr_ts_1us	repeat 2000				wait 2ms
10	vr_ts_1us			Strobe	Strobe	strobe results
11	vr_ts_1us	repeat 10				
12	vr_ts_1us			PSet vin_min		Set VIN to min spec
13	vr_ts_1us	repeat 1000				wait 1ms
14	vr_ts_1us			Strobe	Strobe	strobe results
15	vr_ts_1us	repeat 10				
16	vr_ts_1us			PSet vin_nom		Set VIN to nom spec
17	vr_ts_1us	repeat 1000				
18	vr_ts_1us			Strobe	Strobe	strobe results
19	vr_ts_1us	repeat 10				
20	vr_ts_1us				PSet vreg_iload_min_irated	Set VOUT load to min
21	vr_ts_1us	repeat 1000				wait 1ms
22	vr_ts_1us			Strobe	Strobe	strobe results
23	vr_ts_1us	repeat 10				
24	vr_ts_1us				PSet vreg_iload_full_irated	set VOUT load to full
25	vr_ts_1us	repeat 1000				wait 1ms
26	vr_ts_1us			Strobe	Strobe	strobe results

Fig.2. Pattern Example for POP

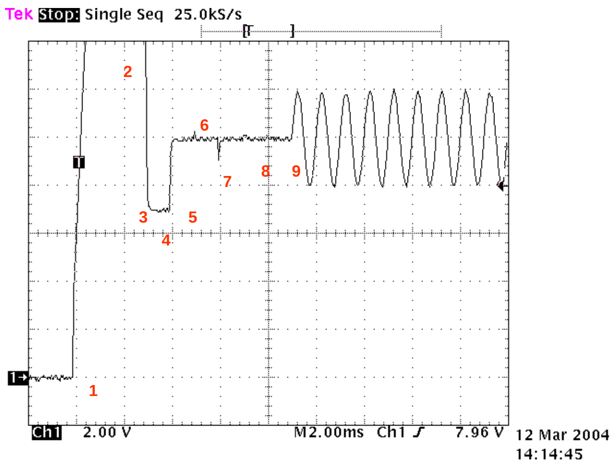
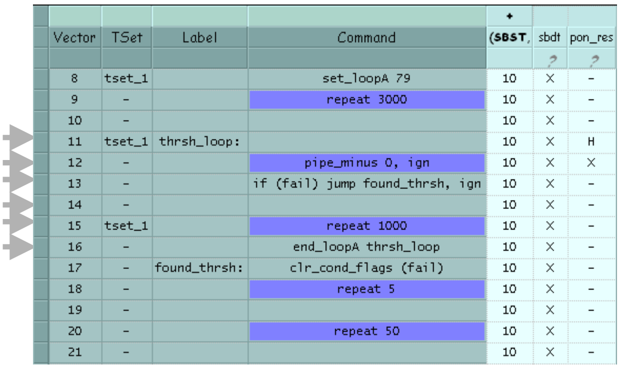


Fig.3. Capture Wave Form by Oscilloscope



Vector	TSet	Label	Command	(SBST,	sbdt	pon_res
8	tset_1		set_loopA 79	10	X	-
9	-		repeat 3000	10	X	-
10	-			10	X	-
11	tset_1	thrsh_loop:		10	X	H
12	-		pipe_minus 0, ign	10	X	X
13	-		if (fail) jump found_thrsh, ign	10	X	-
14	-			10	X	-
15	tset_1		repeat 1000	10	X	-
16	-		end_loopA thrsh_loop	10	X	-
17	-	found_thrsh:	clr_cond_flags (fail)	10	X	-
18	-		repeat 5	10	X	-
19	-			10	X	-
20	-		repeat 50	10	X	-
21	-			10	X	-

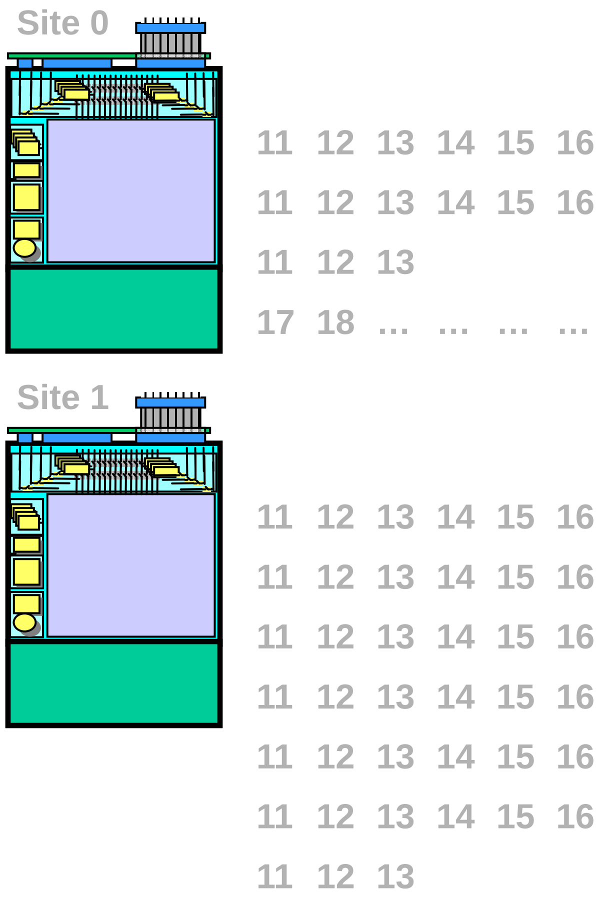


Fig.4. Pattern Threading

Since each board has it's own patgen you can support applications like pattern threading where you want to run the same pattern on every site but need the flexibility of each site responding differently to conditional branching. The example above shows how site 0 requires 3 times thru the loop before meeting the jump condition while site 1 requires 6 times thru the loop. Obviously, the key requirement for this to work is that site can NOT share any boards (i.e. patgens), said another way channels from instrument can only go to a single site.

SOC/MCM devices have multiple independent cores capable of being tested at the same time the exact manner in which they operate in the end application.

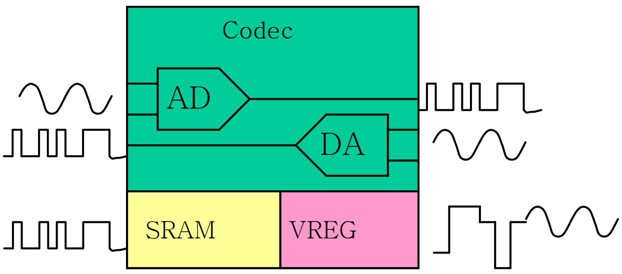


Fig.5. Several Core Inside Device

Conventional Testing does tests in a sequential manner. First one test is performed, then another, etc.

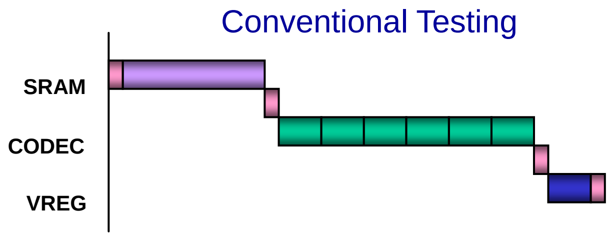


Fig.6. Conventional Testing

In Concurrent Testing, tests are run in parallel within the same device. The elapsed time to run all tests is the time required to run the longest test.

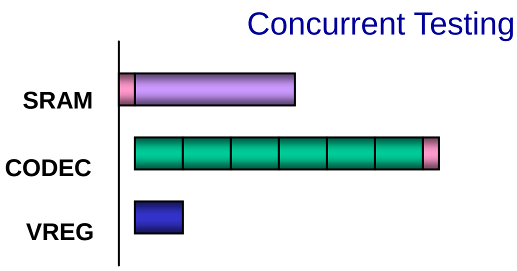


Fig.7. Concurrent Testing

Next picture demonstrate Background DSP and IIM. The key point is that while the pattern is executing data is moved from the instrument (IIM) to the DSP for processing, and this can happen multiple times in a pattern.

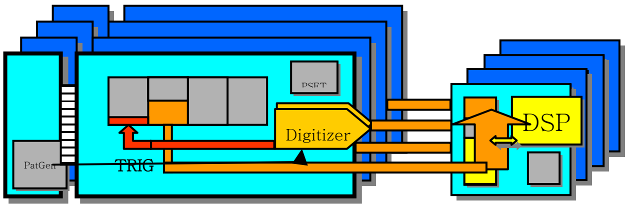


Fig.8. Background DSP & IIM

III. EXAMPLES

Just to state it one more time. POP stands for Pattern Oriented Programming. POP tests are pattern based instead of code based. This is important be POP is a key enabler for making effective use of Flex's architectural features/benefits.

To help make this a little more concrete we're going to review two applications that use POP, give a basic overview of the tests and the benefits derived from using POP.

	Test Time
Non POP	4.0 S
POP	0.5 S

Table.1 Test Time Comparison between POP and Non-POP by Automotive Engine Control Device

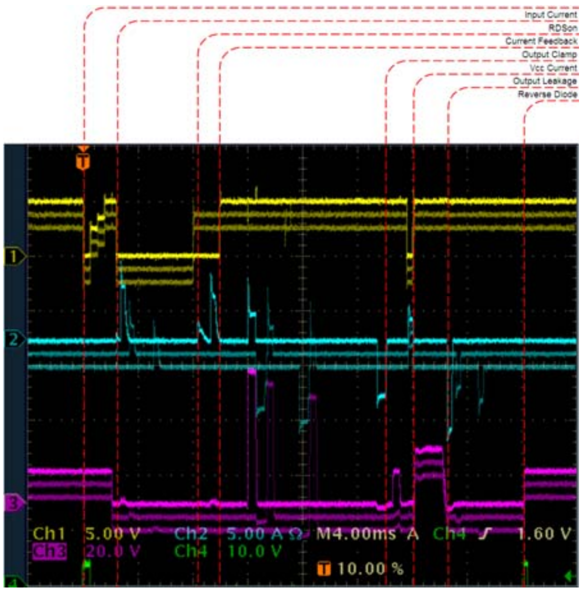


Fig.9. Capture Wave Form by Oscilloscope with Automotive Engine Control Device

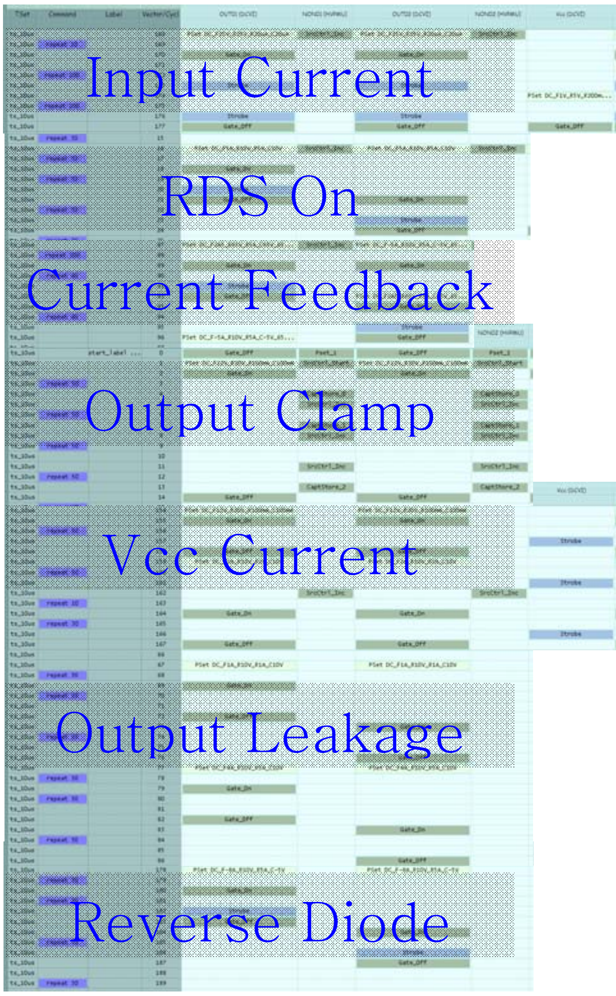


Fig.10. Pattern Example of POP with Automotive Engine Control Device

	Test Time
Non POP	3.4 S
POP	1.9 S

Table.2 Test Time Comparison between POP and Non-POP by BaseBand Processor for GSM Cell Phones

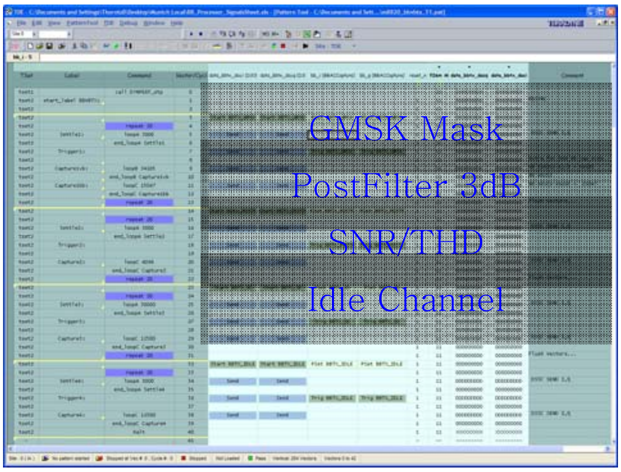


Fig.11. Pattern Example of POP with BaseBand Processor for GSM Cell Phones

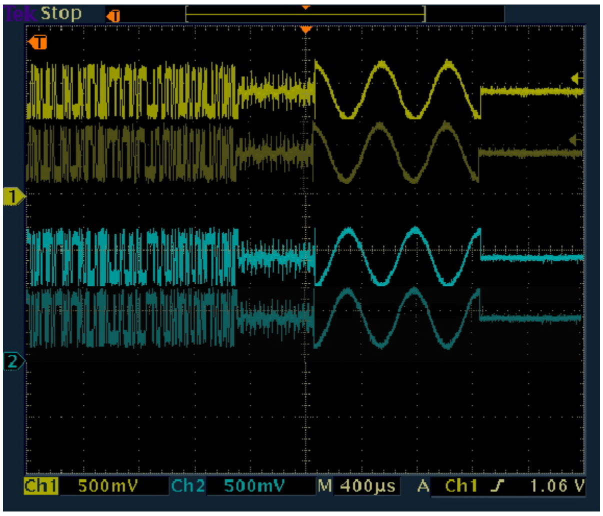


Fig.12. Capture Wave Form by Oscilloscope with BaseBand Processor for GSM Cell Phones

IV. CONCLUSION

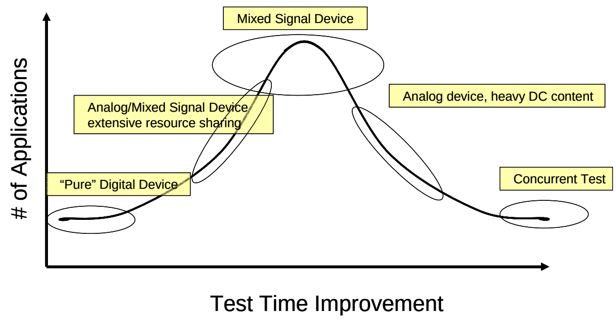


Fig.13. Test Time Improvement by a Kind of Devices

You will see very little test time improvement if you simply take an existing program and convert it line for line. Effective POP programming requires you to step back and understand the tests and the DUT and how to best apply the instrumentation features. This understanding allows you to apply pin conditions in parallel and group tests that use the same connection and basic conditions within a single pattern.

These are just examples, obviously results for individual devices will vary depending on a number of factors. The idea of this slide is just to talk about the key points that affect the effectiveness of POP and how they vary from one device type to the next.

It take a couple of tests from a working test program and convert them to use POP. It will give you some idea of the benefits that can be derived from POP and get your feet wet with some of the development issues and challenges.

If you have a new effort staring soon, or just started, and you have spent time to review the DUT, test list and system configuration, and it looks like POP could be applied more fully. Then start from scratch with POP in mind – or even coherent POP. Frankly, we're still at an early stage in POP and we expect to learn as much from you as you'll learn from us.

REFERENCES

[1] Cutting Test Times by Utilizing All HVD Features at Once by R Gruber (TUG 2004)

[2] Speeding Up DC and Timing Tests Using FLEX DCVI and HVD PSET Feature by T Friedmann (TUG 2004)

[3] High Performance Testing with the FLEX Pattern Control Architecture by J Rowe (TUG 2004)

[4] Solving Problems with Signals: A Signals FAQ by B Van Wagenen (TUG 2004)

[5] PSets and Signals A Match Made in Heaven by Dan Thornton (TUG 2004)