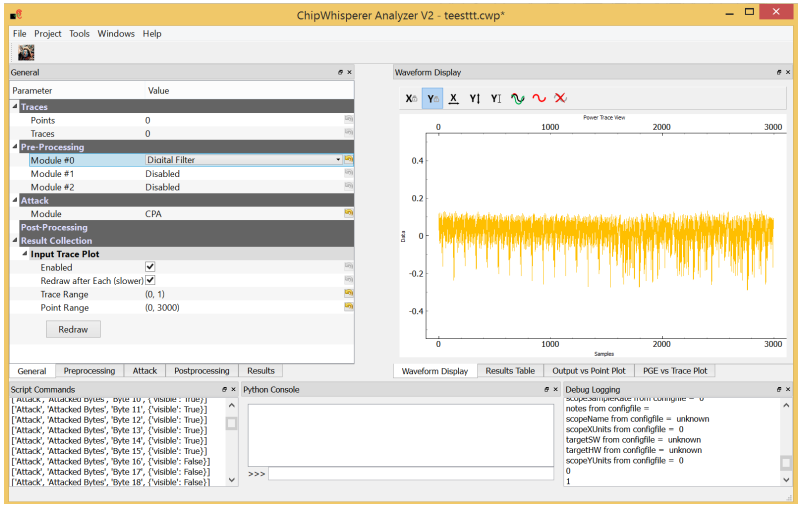
ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research

Colin O'Flynn
Zhizhang Chen
Dalhousie University, Halifax, NS, Canada

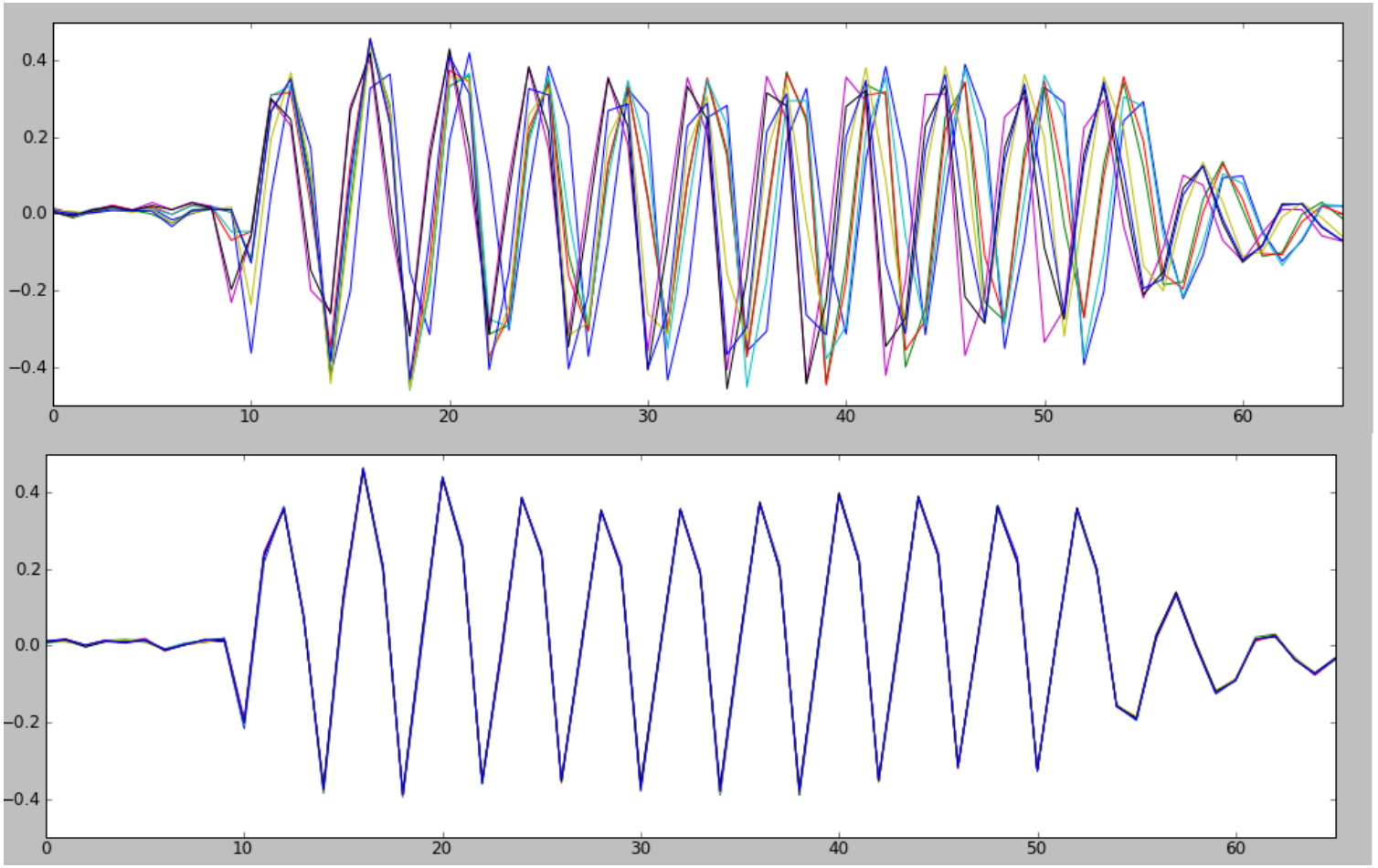
International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)
Paris, France. April 14th-15th.

This Presentation in 60 Seconds

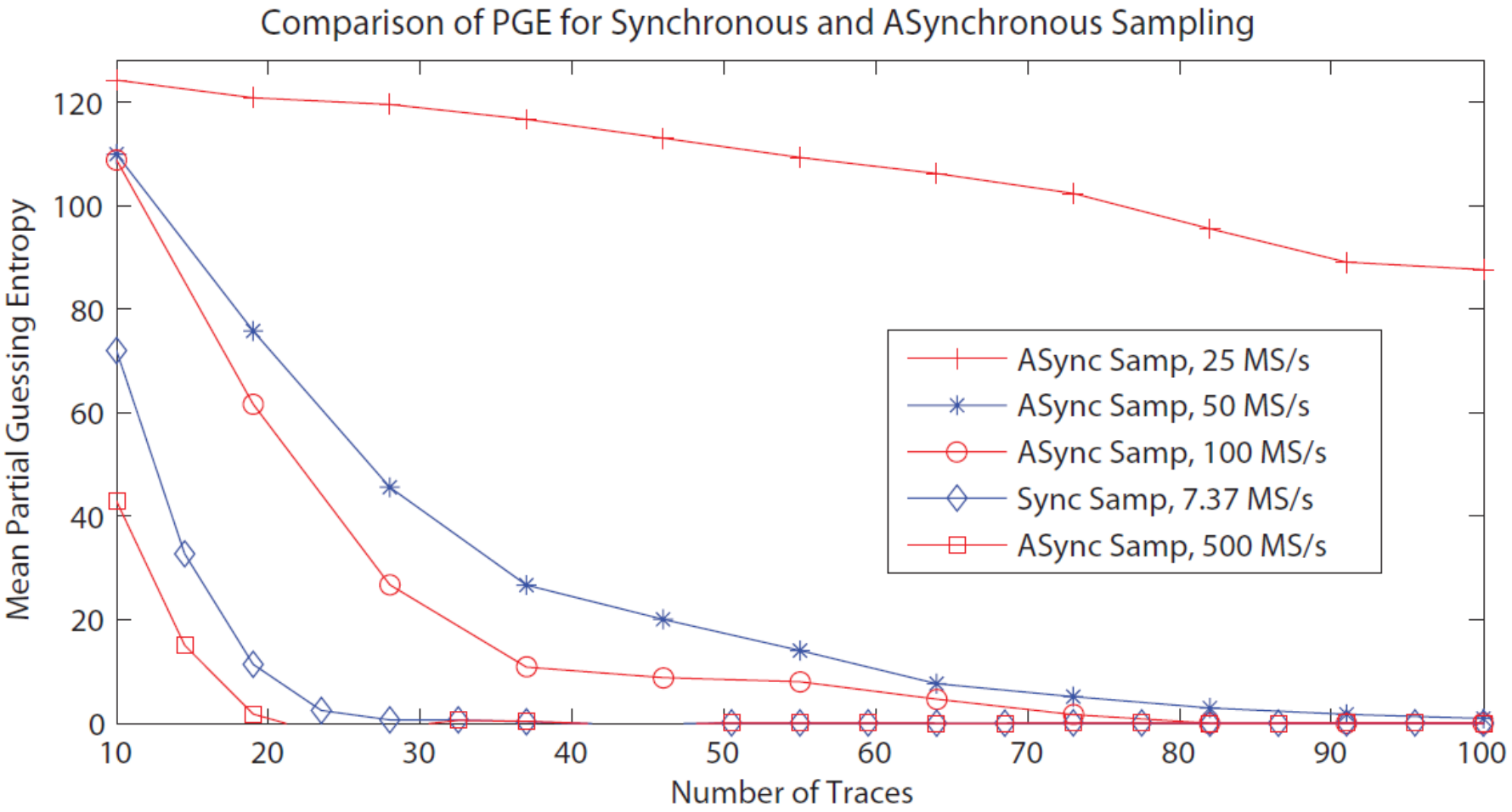


BACKGROUND

Background: Synchronous Capture



Background: Synchronous Capture



Background: Synchronous Capture

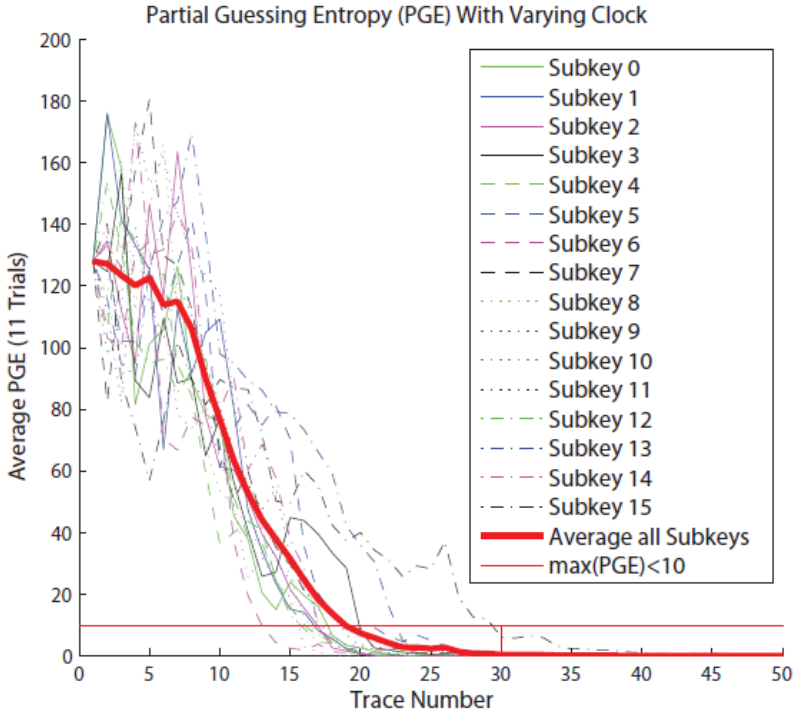
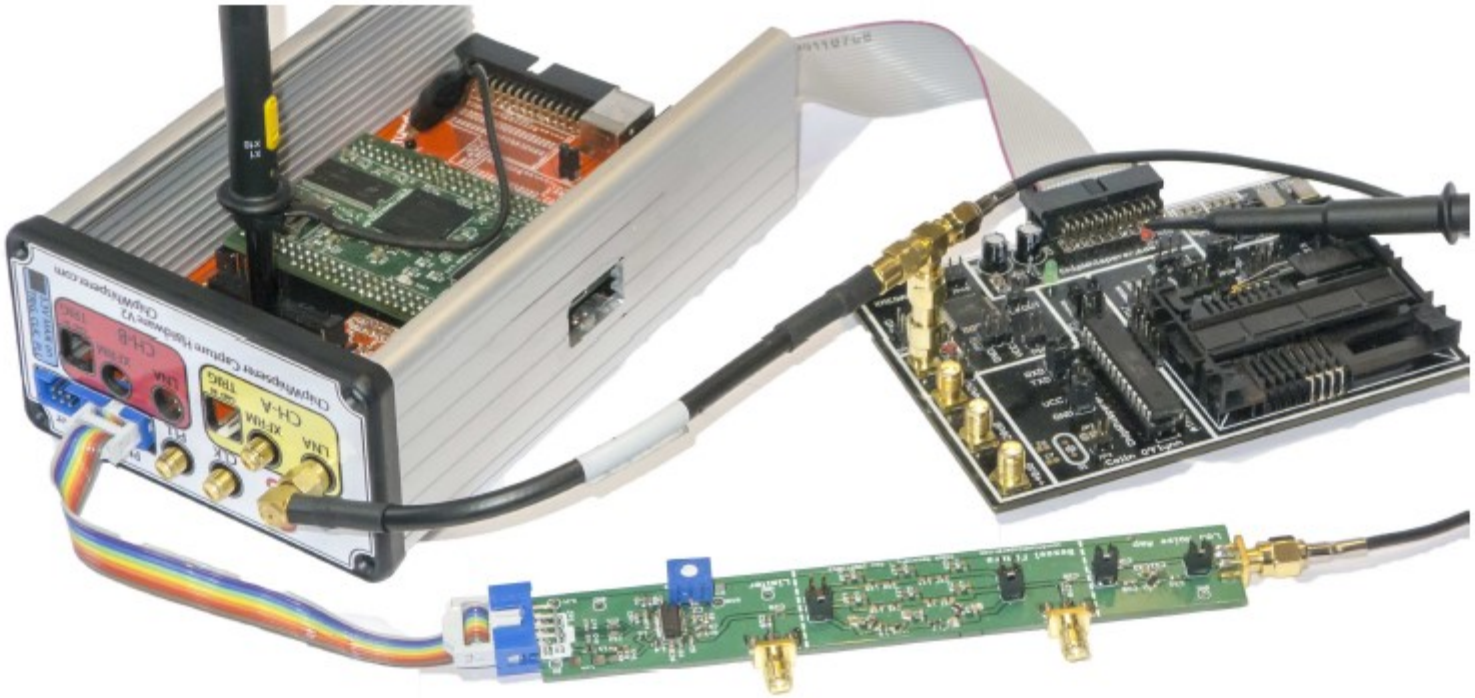


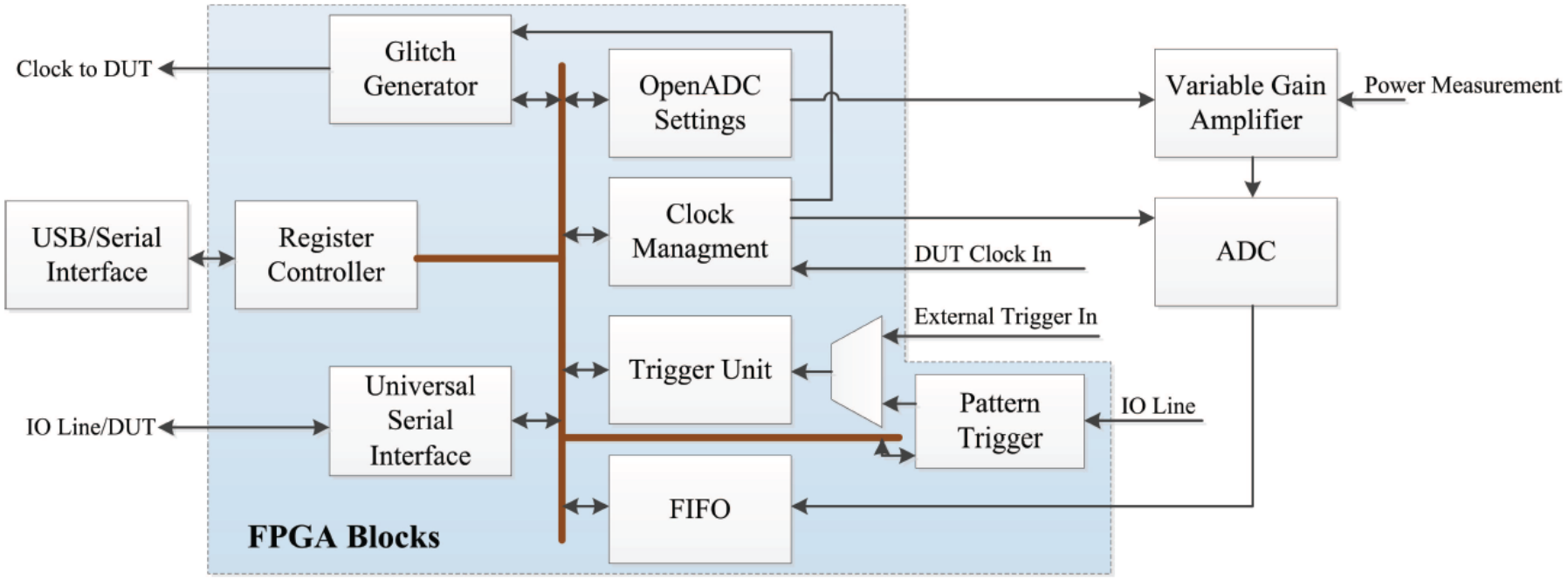
Fig. 8 Results of a CPA attack on a device with oscillator frequency randomly varying between 3.9 MHz–13 MHz on each encryption, and no trace synchronization being performed. The *Byte N* refer to the subkey Partial Guessing Entropy(PGE), *Average* refers to the average of all 16 subkeys. $max(PGE) < 10$ shows the metric used in Table 2.

Background: Clock Recovery

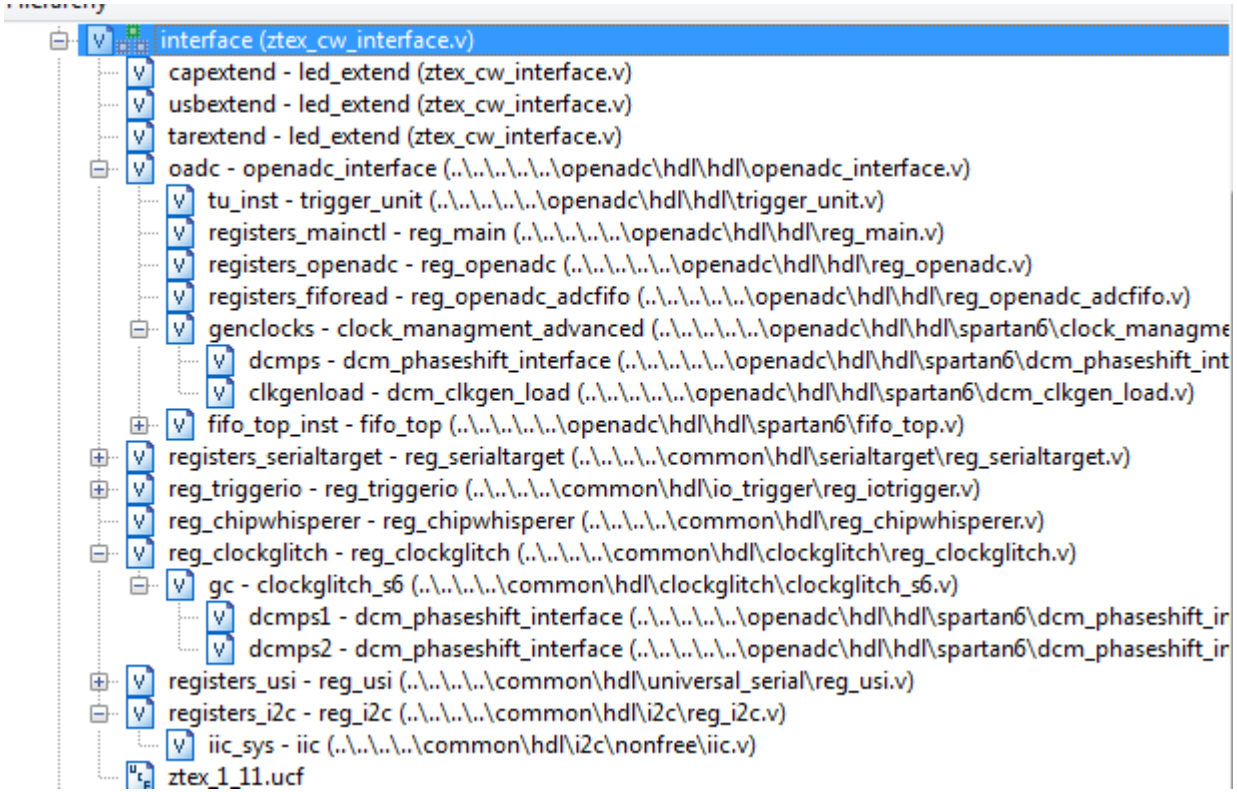


HARDWARE DESIGN

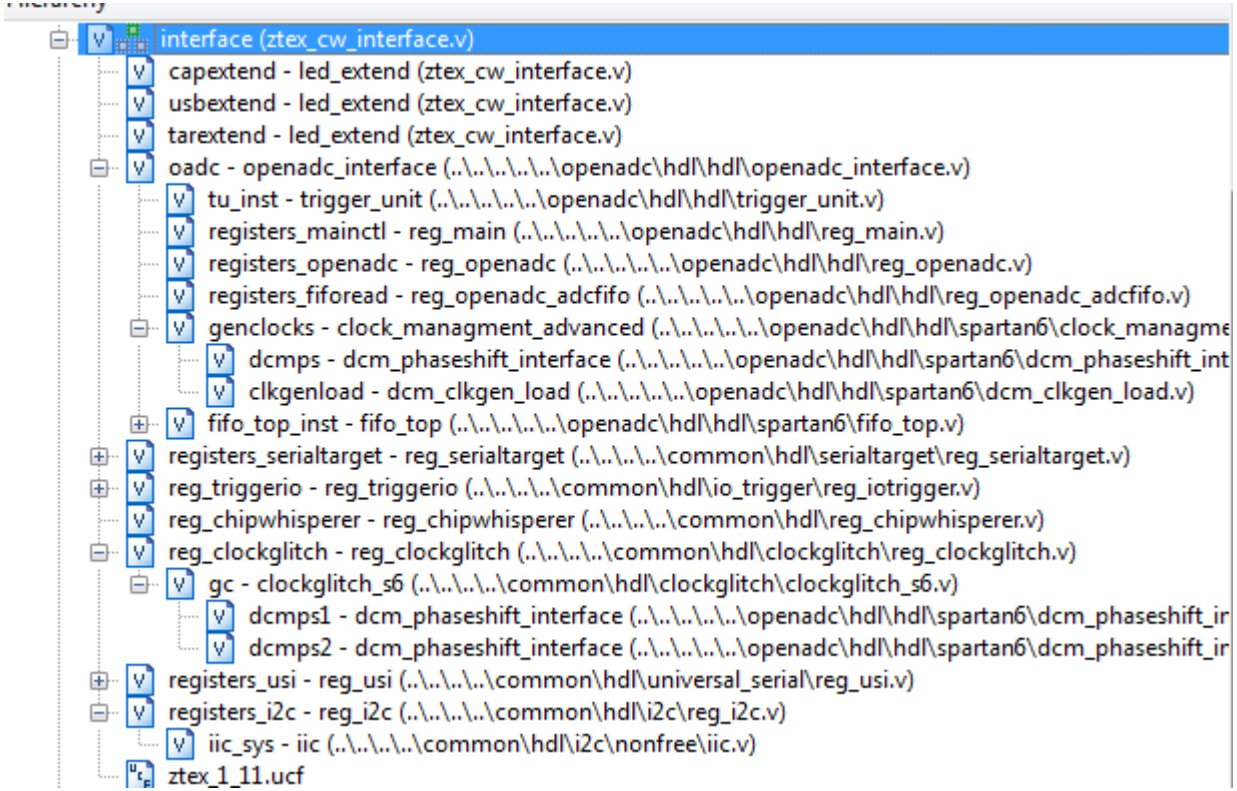
ChipWhisperer Block Diagram



Modular FPGA Design

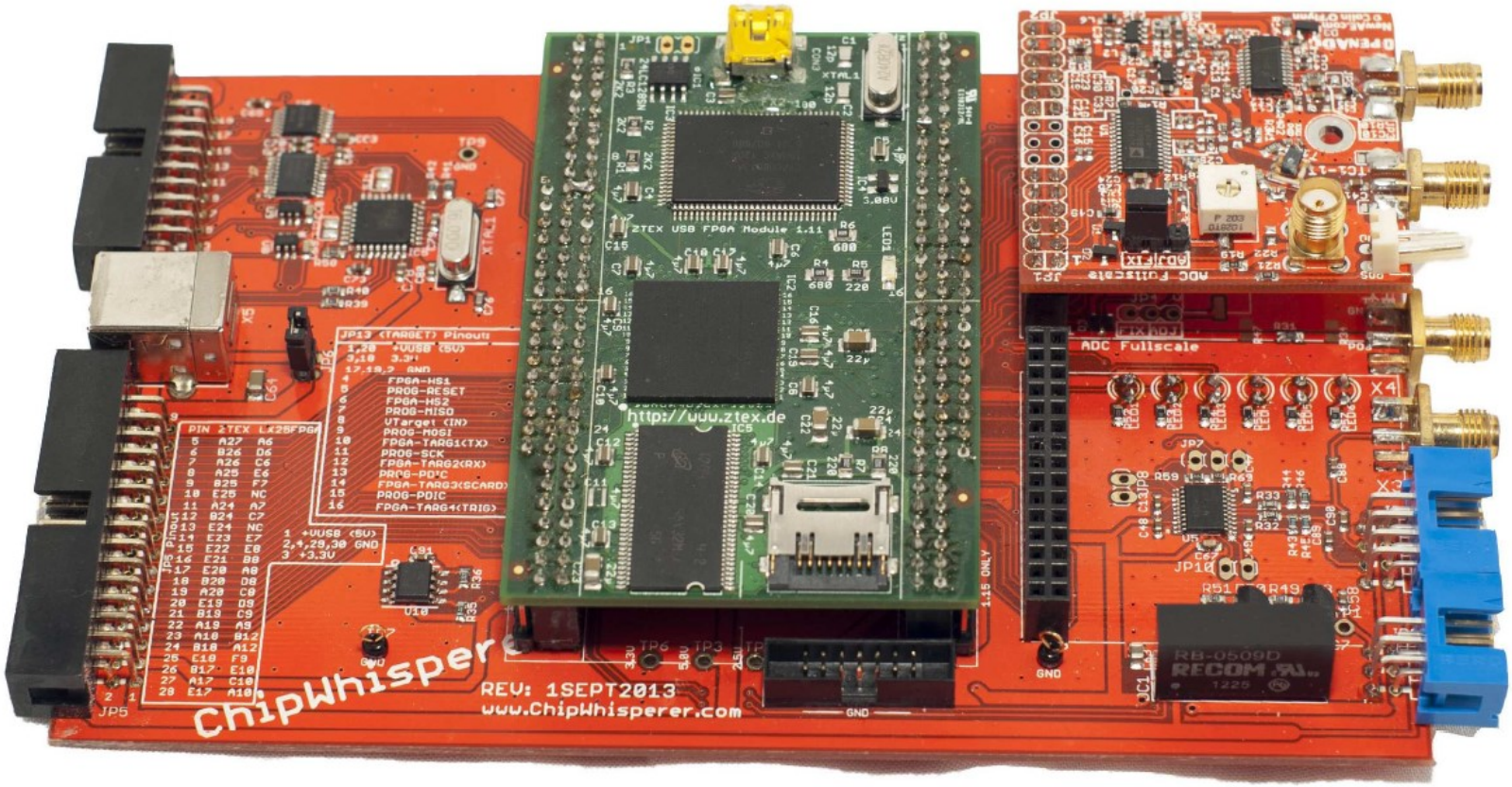


Modular FPGA Design

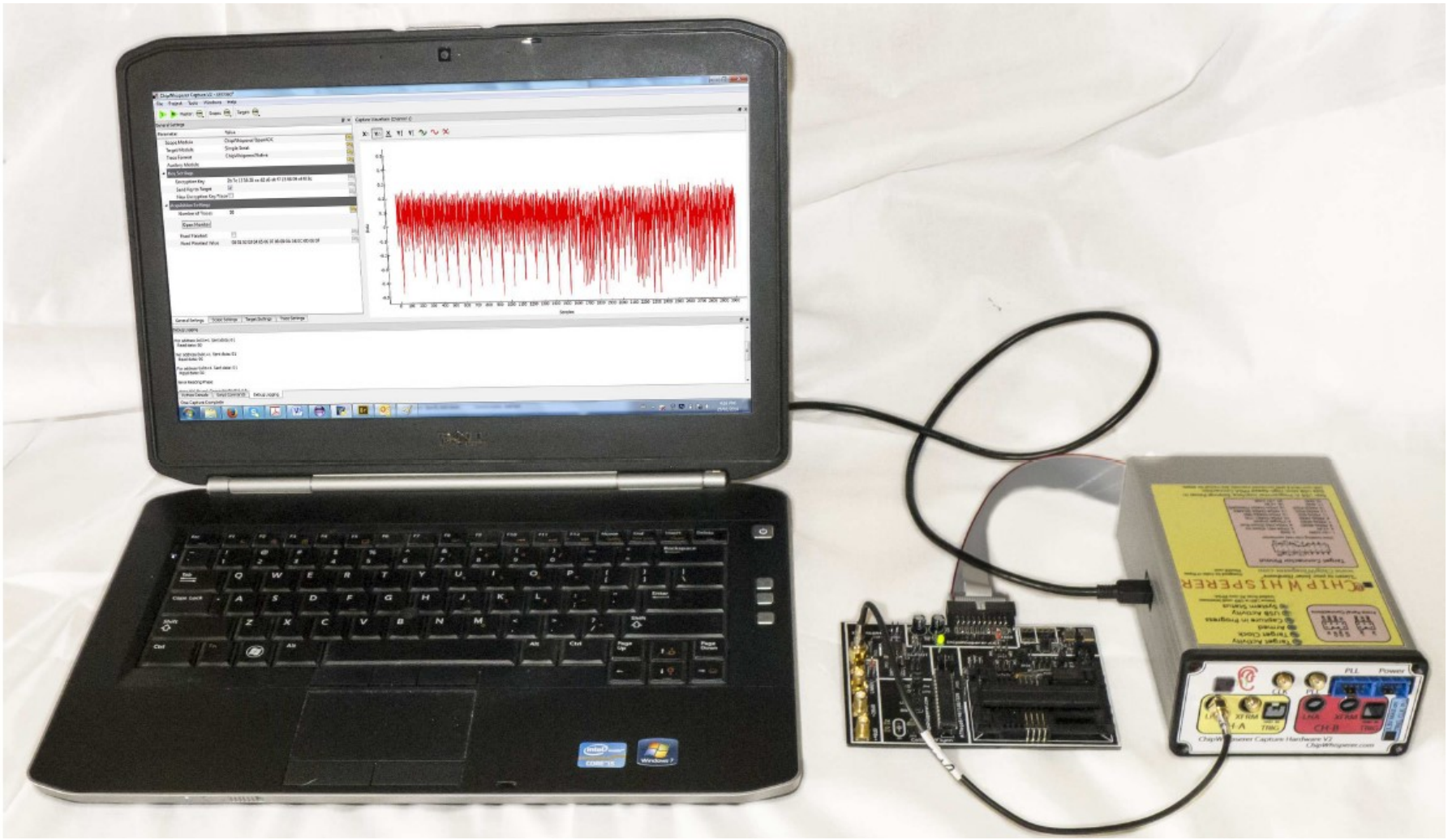



```
reg_usi registers_usi (
    .reset_i(reg_rst),
    .clk(ifclk_buf),
    .reg_address(reg_addr),
    .reg_bytecnt(reg_bcnt),
    .reg_datao(reg_datai_usi),
    .reg_datai(reg_datao),
    .reg_size(reg_size),
    .reg_read(reg_read),
    .reg_write(reg_write),
    .reg_addrvalid(reg_addrvalid),
    .reg_hypaddress(reg_hypaddr),
    .reg_hyplen(reg_hyplen_usi),
    .reg_stream(),
    .usi_out(usi_out),
    .usi_in(usi_in)
);

reg_i2c registers_i2c (
    .reset_i(reg_rst),
    .clk(ifclk_buf),
    .reg_address(reg_addr),
    .reg_bytecnt(reg_bcnt),
    .reg_datao(reg_datai_i2c),
    .reg_datai(reg_datao),
    .reg_size(reg_size),
    .reg_read(reg_read),
    .reg_write(reg_write),
    .reg_addrvalid(reg_addrvalid),
    .reg_hypaddress(reg_hypaddr),
    .reg_hyplen(reg_hyplen_i2c),
    .reg_stream(),
    .scl(pll_scl),
    .sda(pll_sda)
);
```

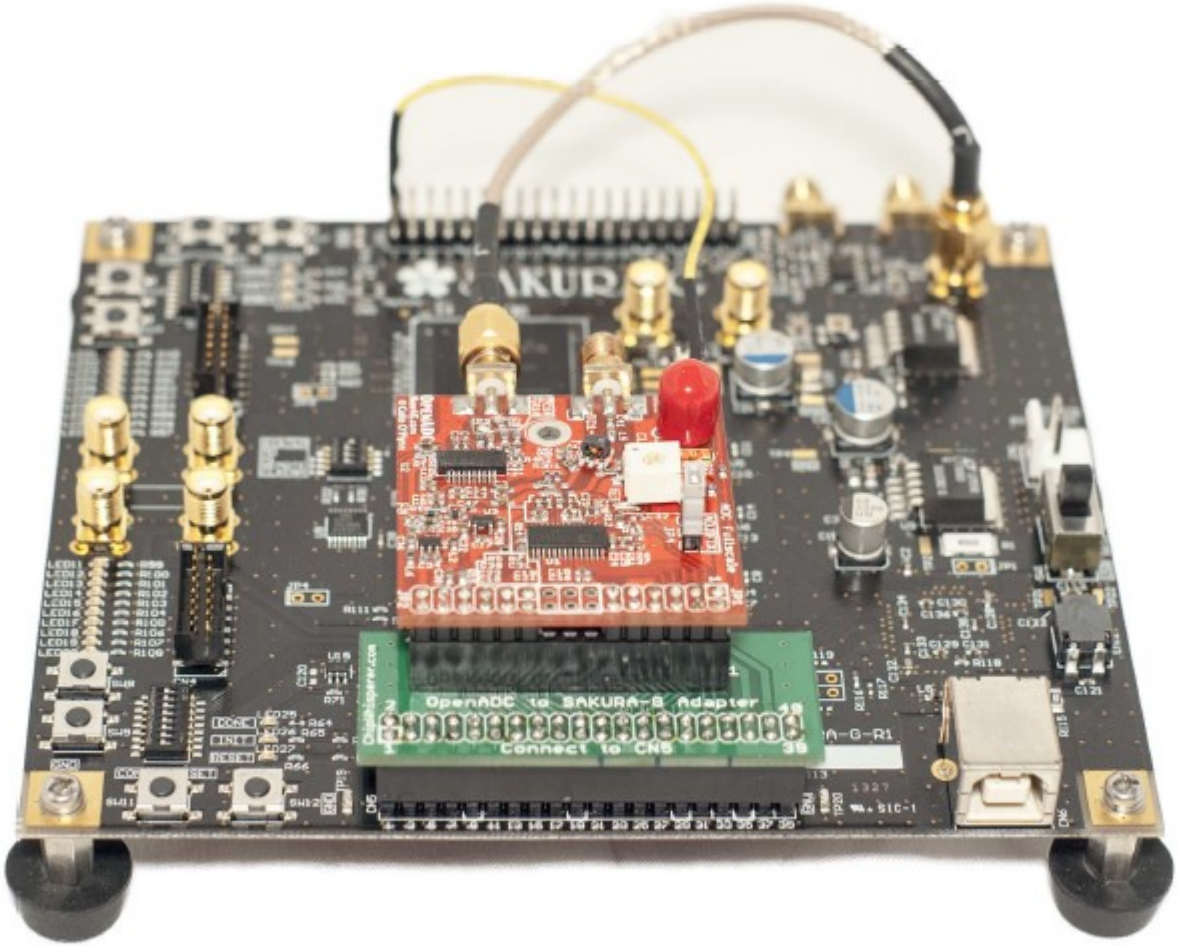
Hardware Implementation



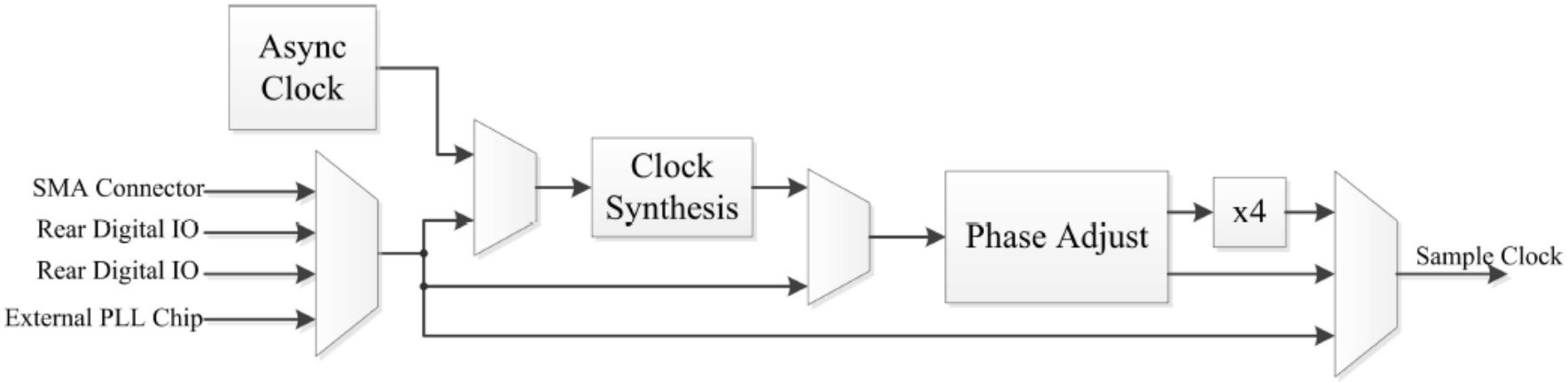
Hardware Implementation



SAKURA-G Version



Clock Routing



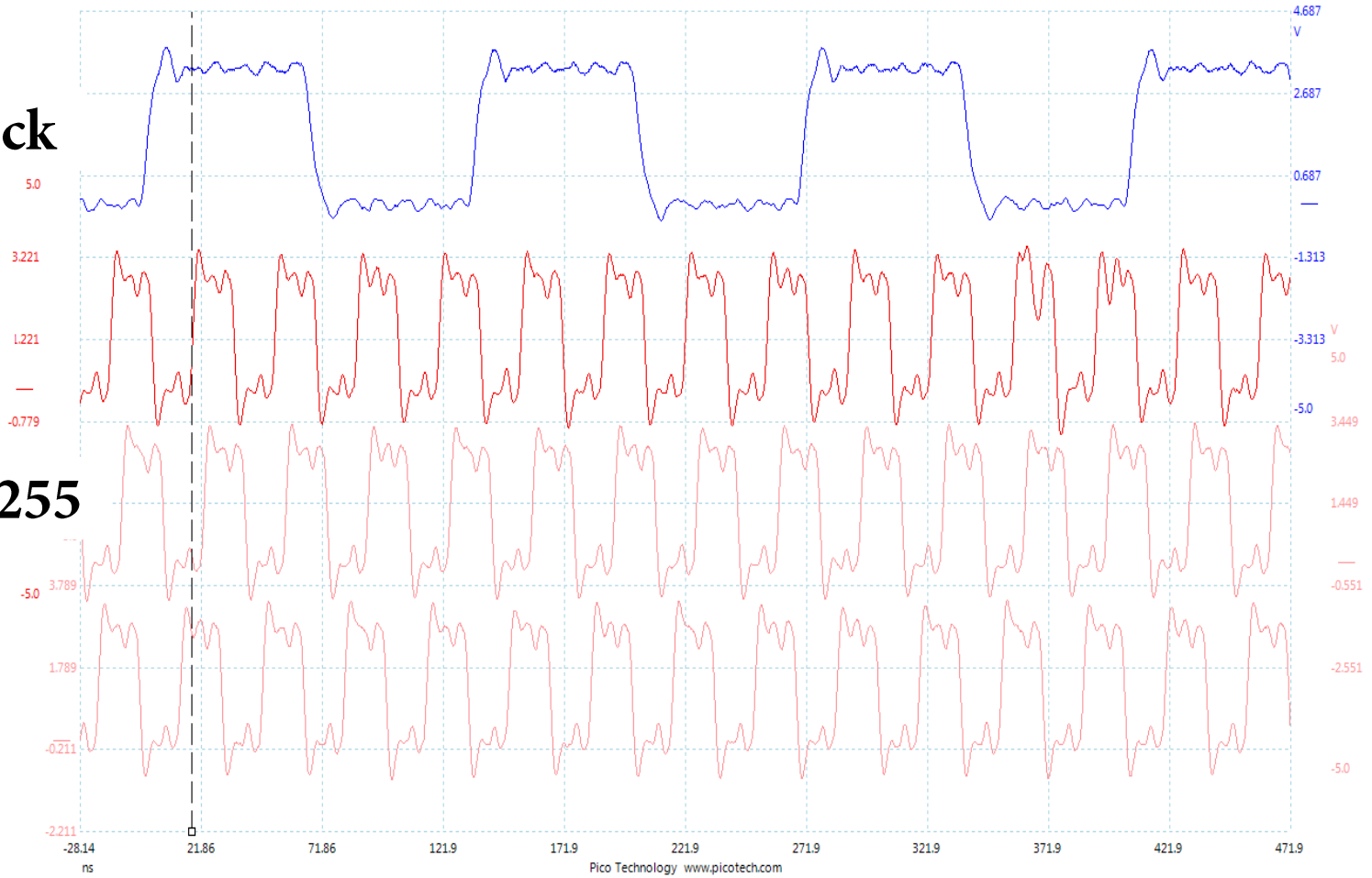
Clock Phase Adjustment

DUT Clock

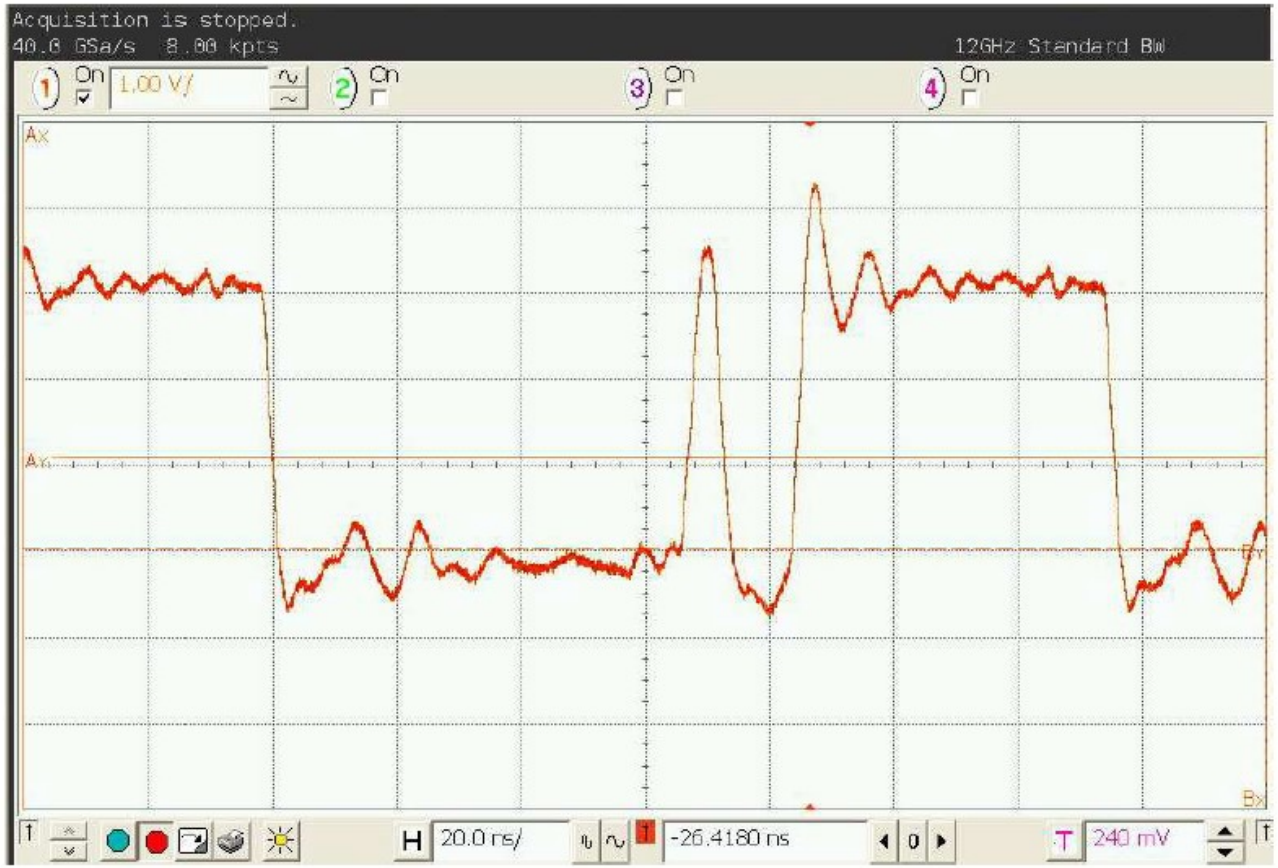
Phase=0

Phase=+255

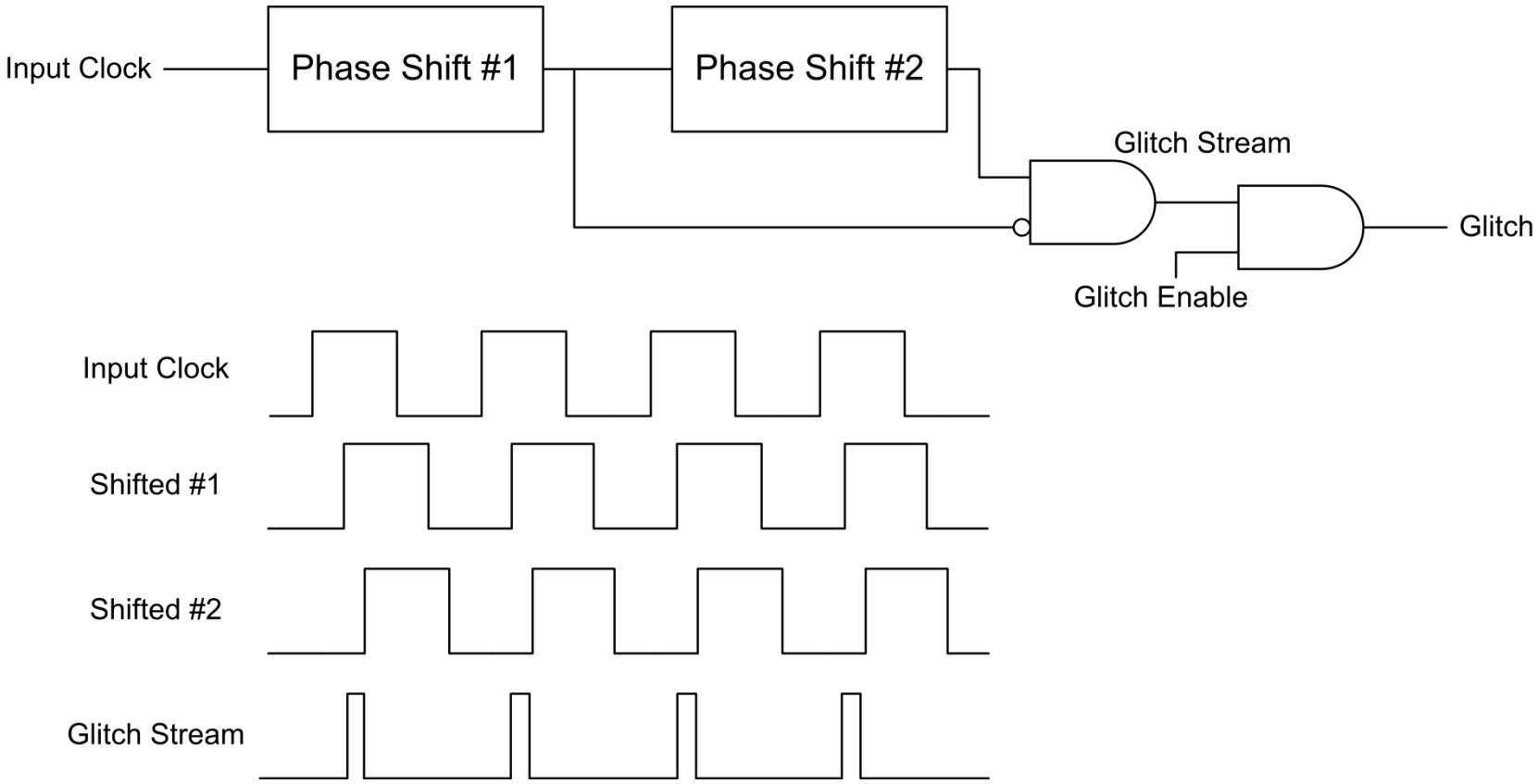
Phase=-255



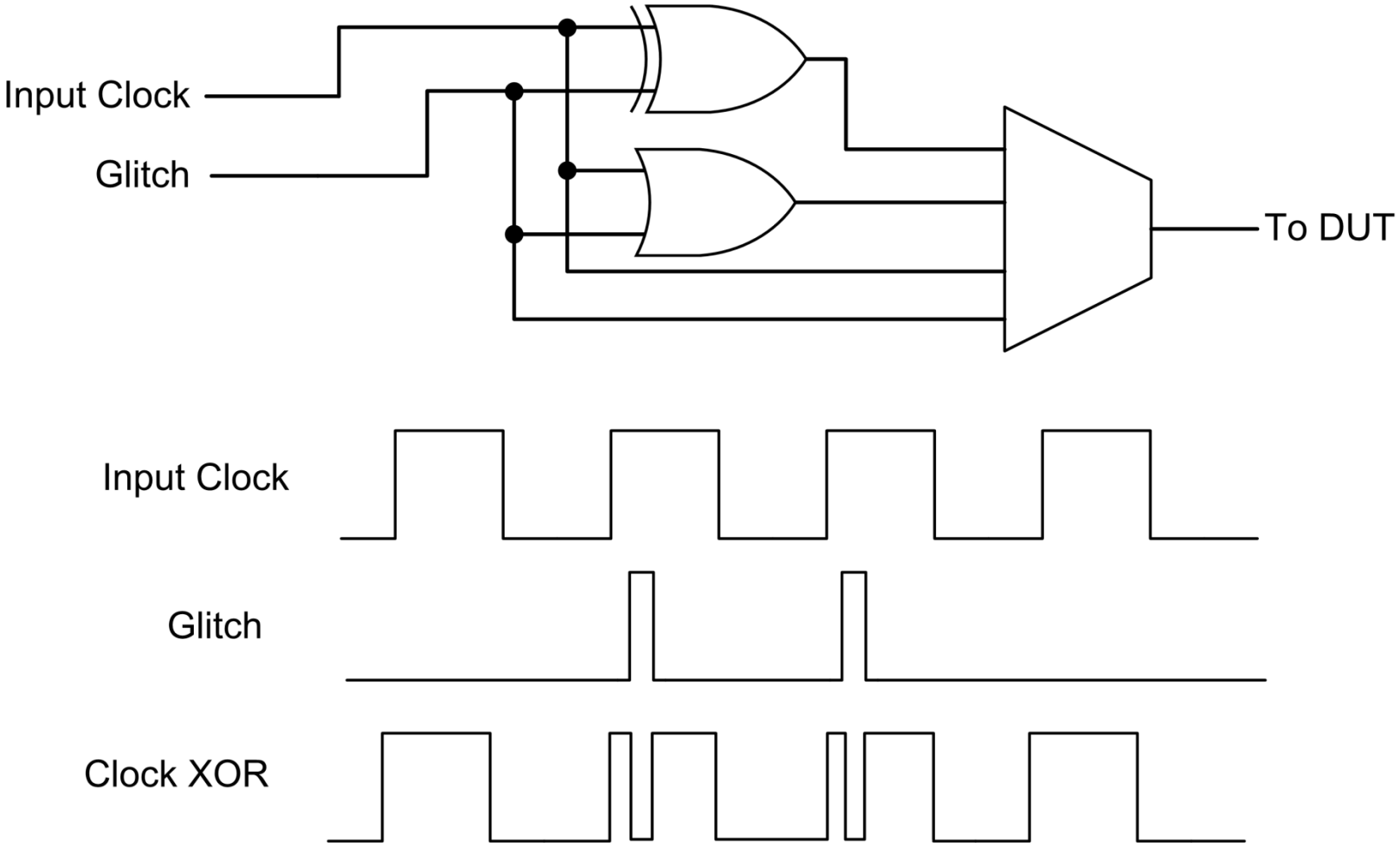
Glitching Support



Glitching Support

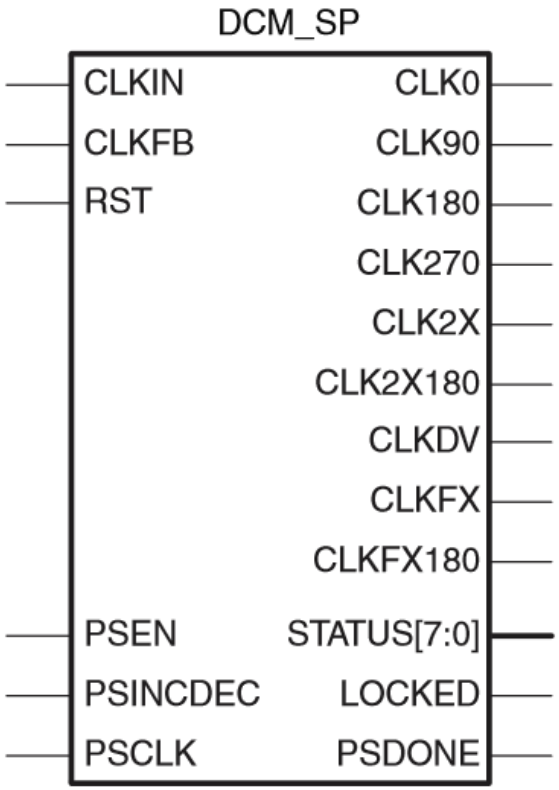


Glitching Support



Partial Reconfiguration

- DCM Blocks:



Partial Reconfiguration

Delay Lines										
DCM_DELAY_STEP(5)	Finest delay resolution, averaged over all steps.	10	40	10	40	10	40	10	40	ps

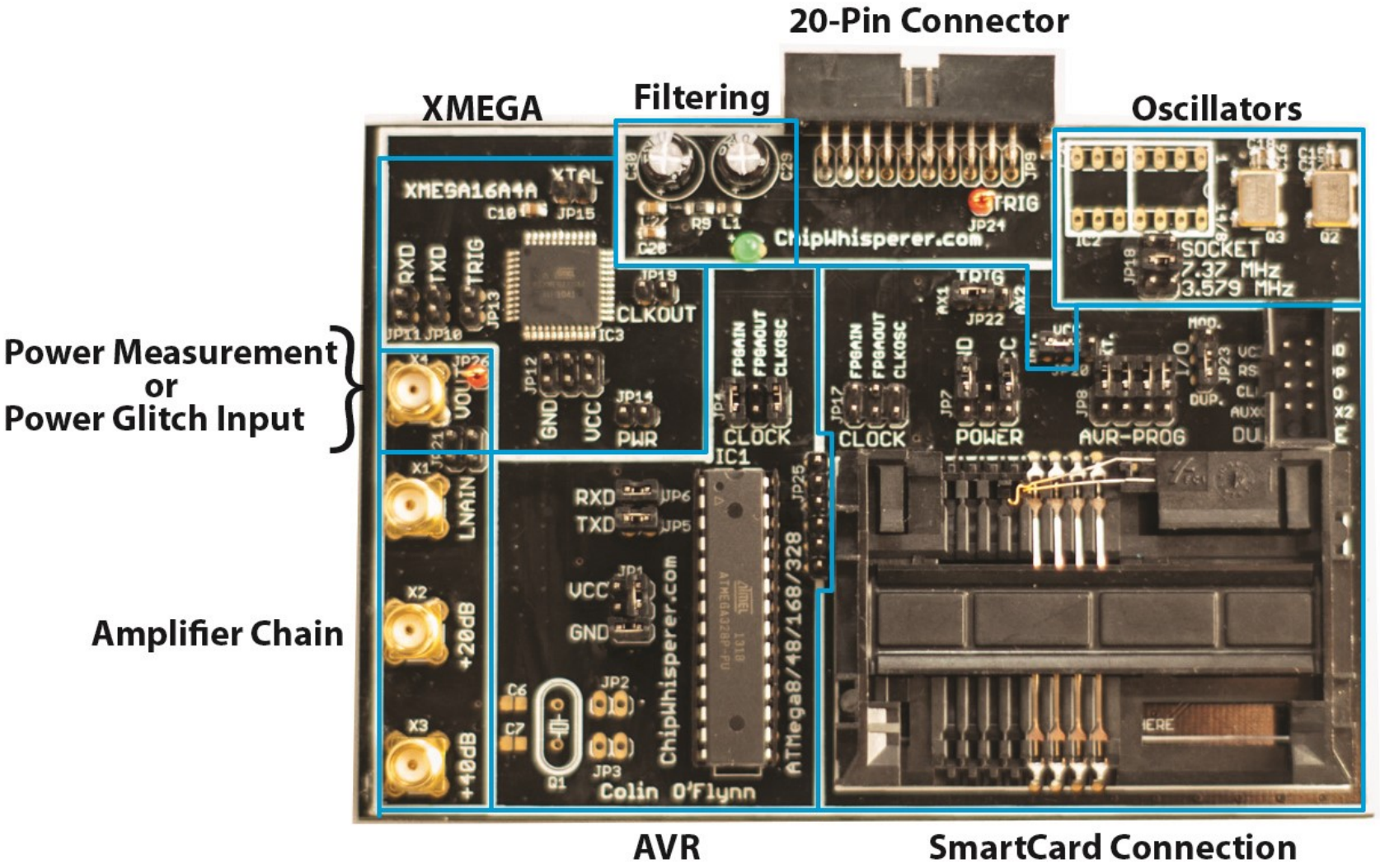
Table 59: **Switching Characteristics for the Phase-Shift Clock in Variable Phase Mode(1)**

Symbol	Description	Amount of Phase Shift	Units
Phase Shifting Range			
MAX_STEPS(2)	When CLKIN < 60 MHz, the maximum allowed number of DCM_DELAY_STEP steps for a given CLKIN clock period, where T = CLKIN clock period in ns. When using CLKIN_DIVIDE_BY_2 = TRUE, double the clock-effective clock period.	±(INTEGER(10 x (TCLKIN – 3 ns)))	steps
	When CLKIN ≥ 60 MHz, the maximum allowed number of DCM_DELAY_STEP steps for a given CLKIN clock period, where T = CLKIN clock period in ns. When using CLKIN_DIVIDE_BY_2 = TRUE, double the clock-effective clock period.	±(INTEGER(15 x (TCLKIN – 3 ns)))	steps
FINE_SHIFT_RANGE_MIN	Minimum guaranteed delay for variable phase shifting.	±(MAX_STEPS x DCM_DELAY_STEP_MIN)	ps
FINE_SHIFT_RANGE_MAX	Maximum guaranteed delay for variable phase shifting	±(MAX_STEPS x DCM_DELAY_STEP_MAX)	ps

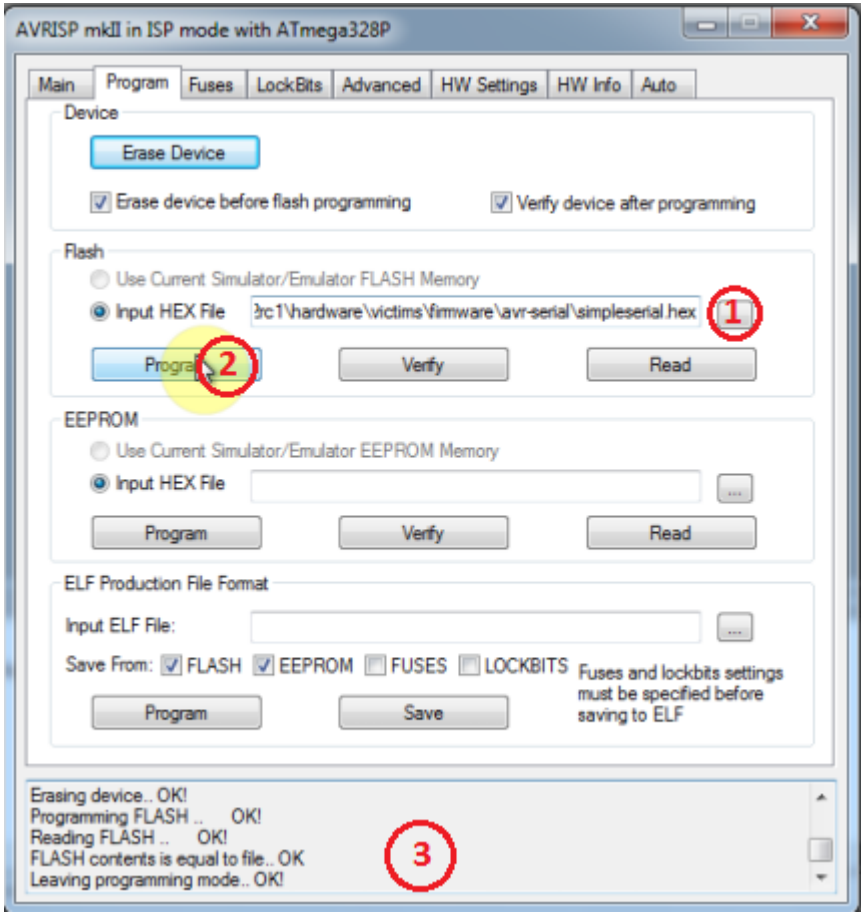
Partial Reconfiguration

- Generate bitstreams for fixed phase shift
  - 256 options for each DCM to cover -50% to +50%
  - 2 DCMs
- Generate ‘Difference’ Files for internal Partial Reconfiguration module

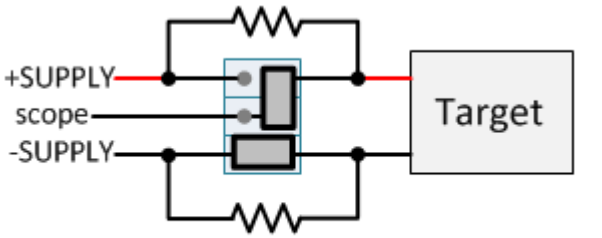
Multi-Target Board



Programming AVR/Xmega/MegaCard

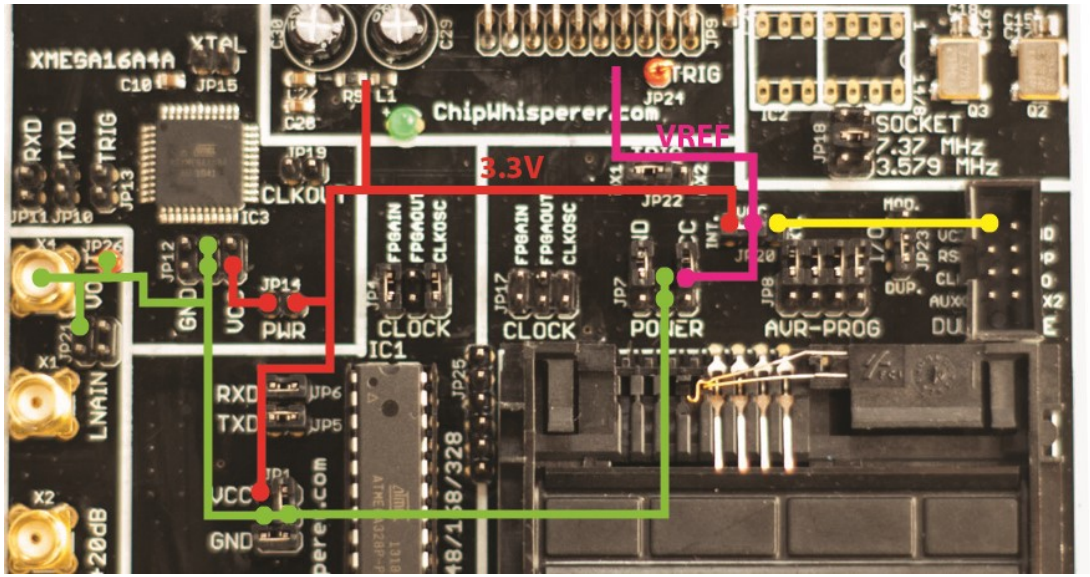


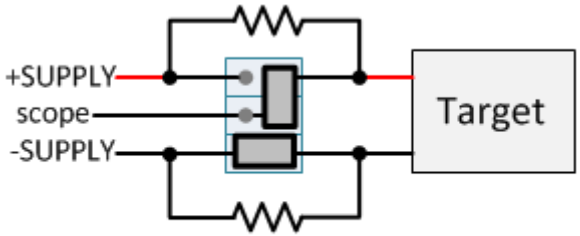
Multi-Target Board



Measuring across VCC Shunt

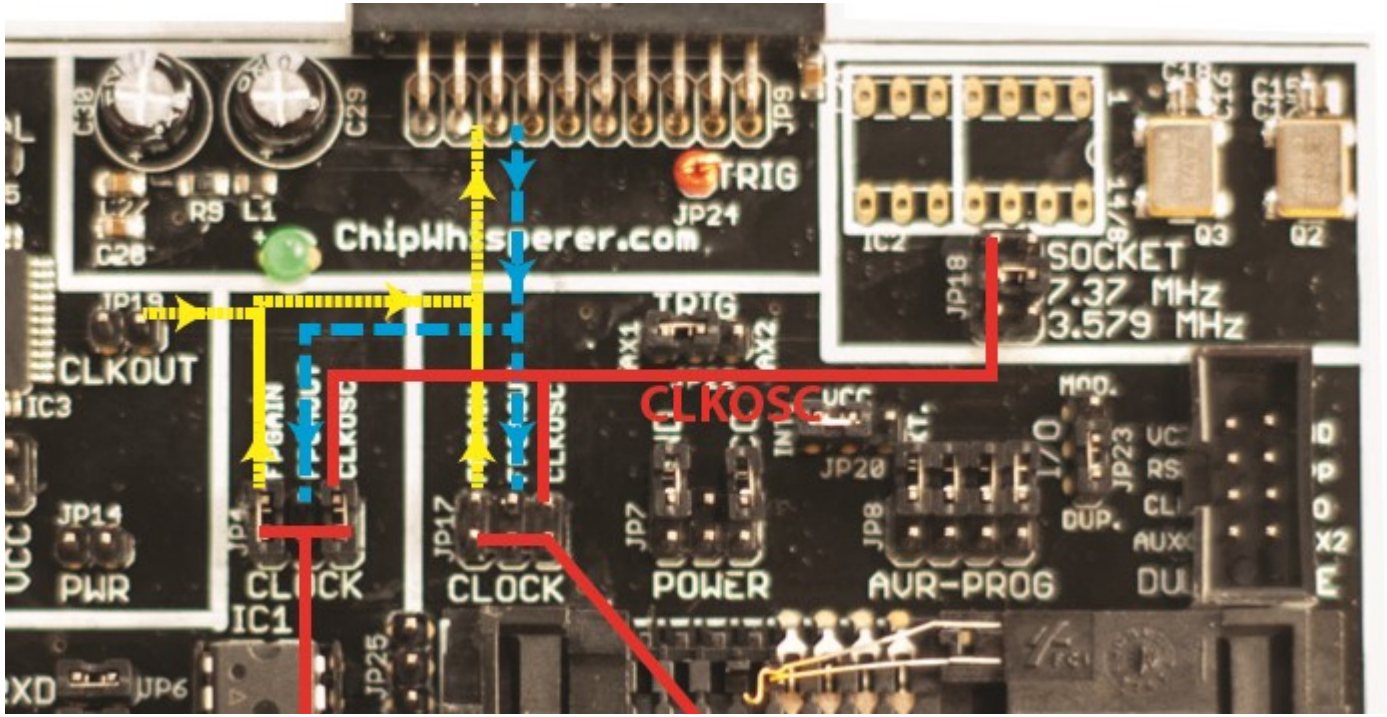
Multi-Target Board





Measuring across VCC Shunt

Multi-Target Board



SOFTWARE DESIGN

Design Requirements

- Allow Scripting
  - Load script into program for initial configuration
  - Run entire program from script
- Run Capture vs. Analysis Separately
  - Allow saving captured traces to custom formats (e.g. MATLAB)

Scripting

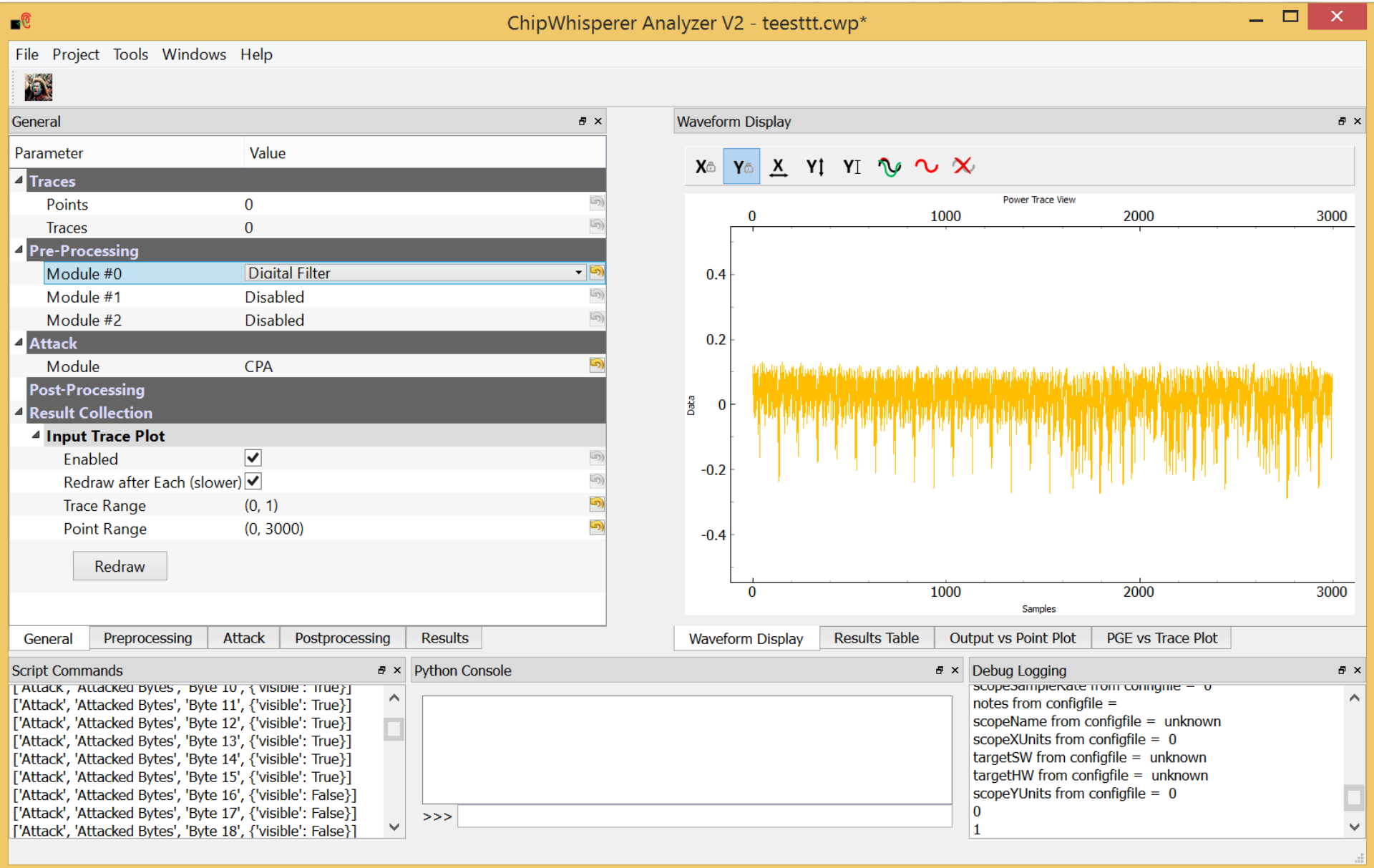
```
#Make the application
app = cwc.makeApplication()

#Get main module
capture = cwc.ChipWhispererCapture()

#Show window - even if not used
capture.show()
pe()

#Call user-specific commands
usercommands = userScript(capture)

usercommands.run()
```

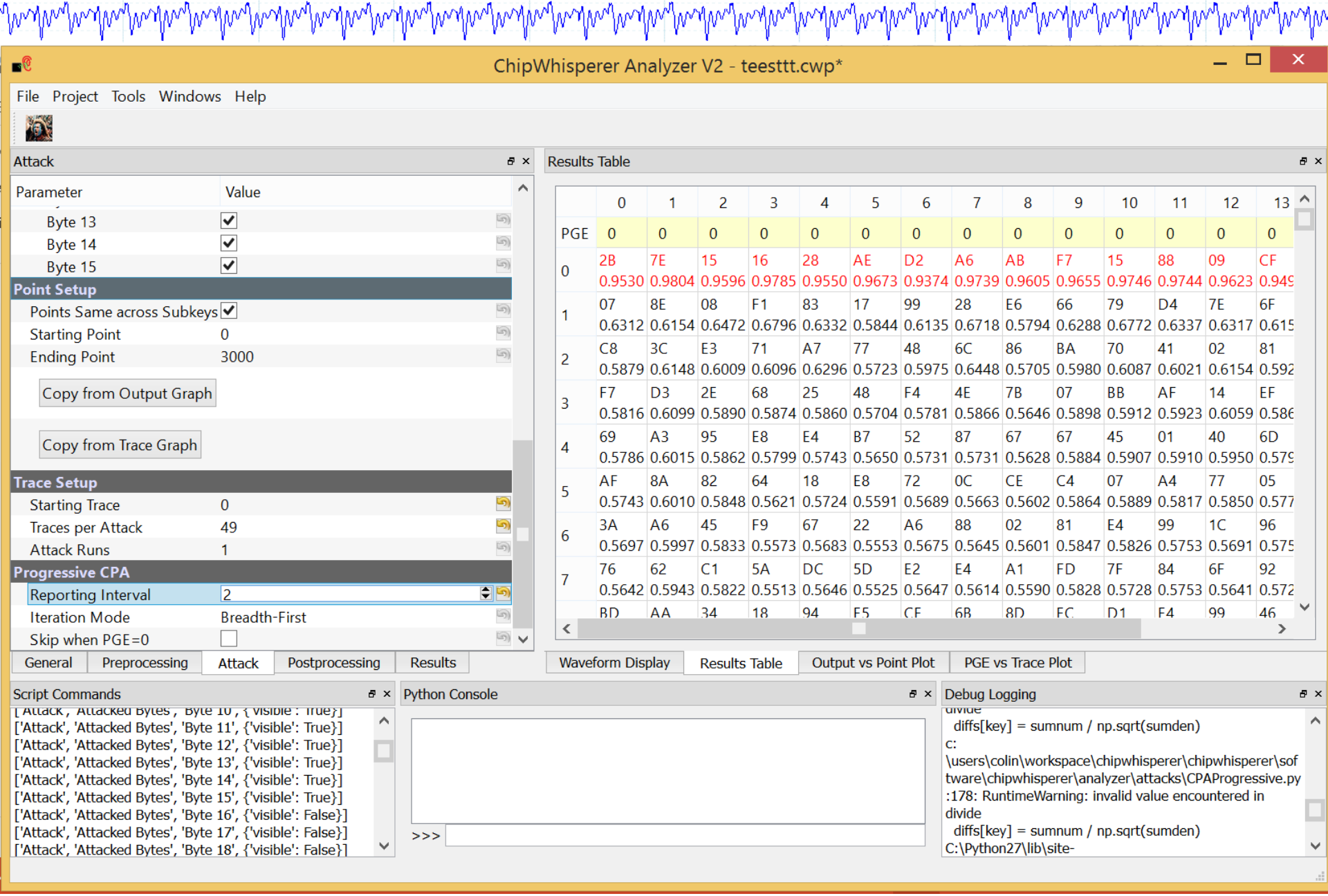
Scripting

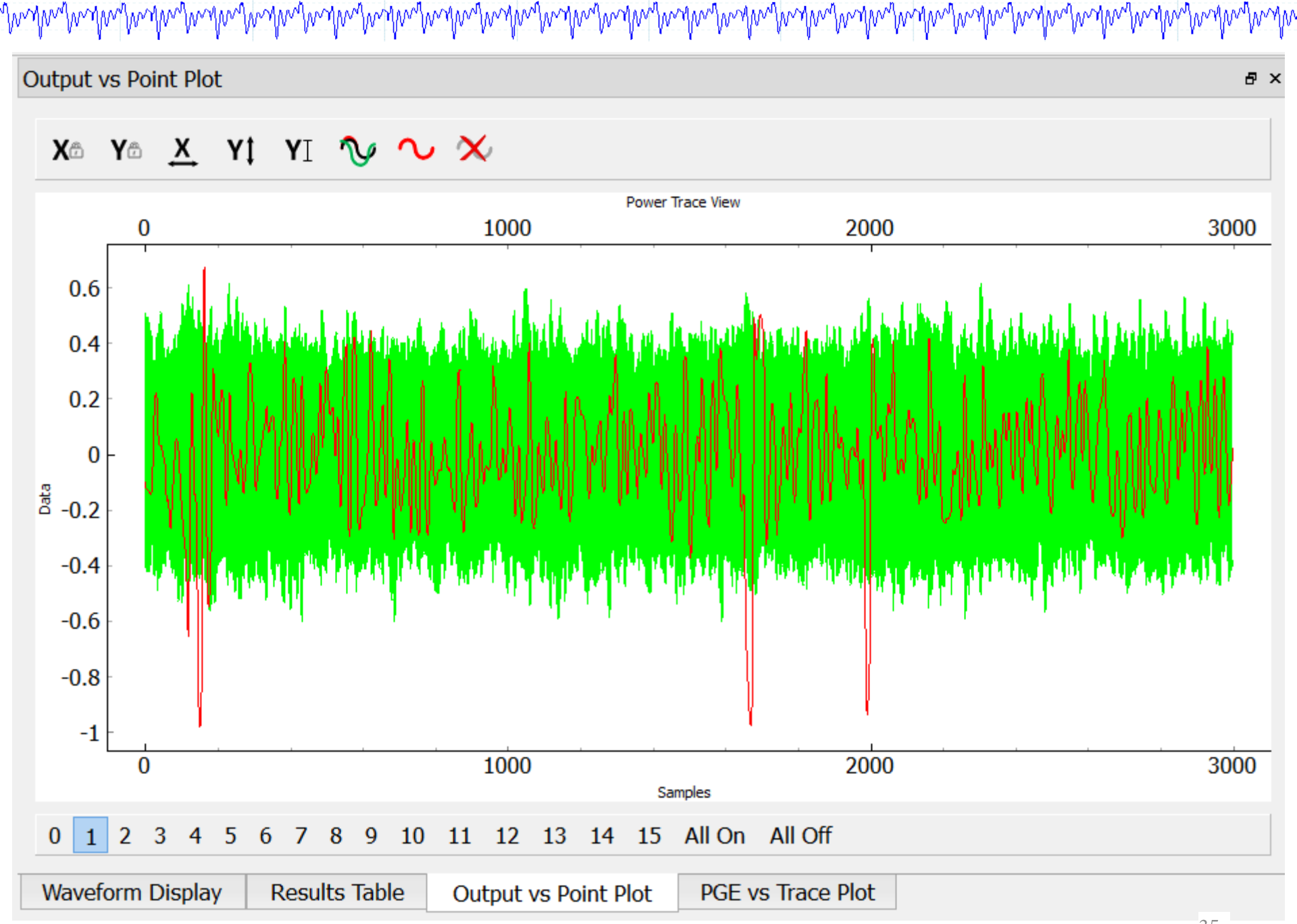
```
#Example of using a list to set parameters. Slightly easier to copy/paste in
this format
lstexample = [['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Front Panel
A', False],
  ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger
Line)', True],
  ['CW Extra', 'CW Extra Settings', 'Clock Source', 'Target IO-IN'],
  ['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'EXTCLK x4 via DCM'],
  ['OpenADC', 'Trigger Setup', 'Total Samples', 3000],
  ['OpenADC', 'Trigger Setup', 'Offset', 1500],
  ['OpenADC', 'Gain Setting', 'Setting', 45],
  ['OpenADC', 'Trigger Setup', 'Mode', 'rising edge'],
  #Final step: make DCMs relock in case they are lost
  ['OpenADC', 'Clock Setup', 'Reset DCMs', None],
  ]

#Download all hardware setup parameters
for cmd in lstexample: cap.setParameter(cmd)
```

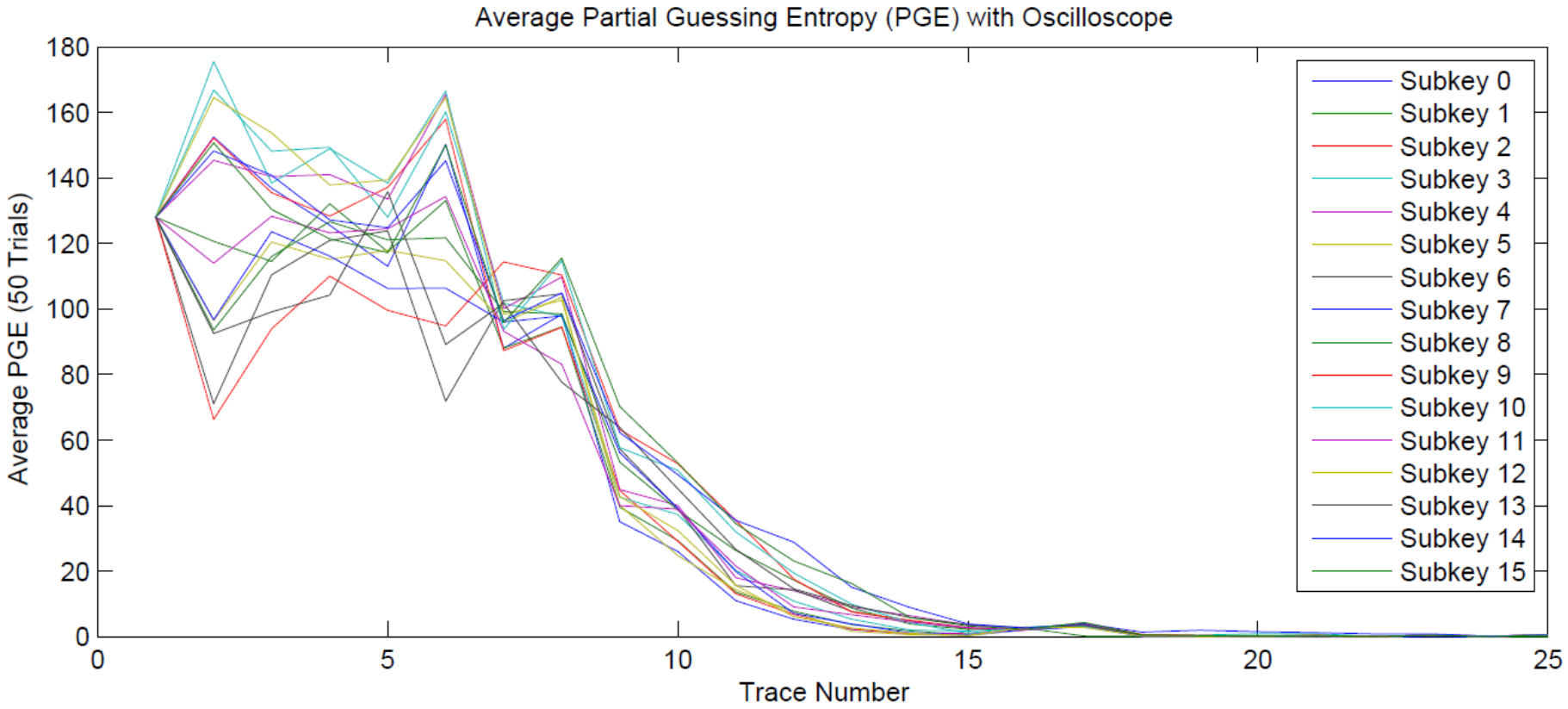
Scripting

```
print "Loading CW-Extra Module"
usi = cwe.CWUniversalSerial()
usi.con(cap.scope.qtadc.sc)

usi.setBaud(9600)
usi.setParity("even")
usi.setStopbits(2)
usi.write([0x80,0x04, 0x04, 0x00, 0x10, None, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01])

time.sleep(0.05)

usi.read(16, startonly=True)
usi.write([0x80,0xC0, 0x00, 0x00, 0x10])
p = bytearray(usi.read(16, waitonly=True))
for t in p:
    print "%2x "%t,
```


ChipWhisperer Analyzer V2 - teesttt.cwp*
File Project Tools Windows Help
General
Parameter
Value
Traces
Points 0
Traces 0
Pre-Processing
Module #0 Digital Filter
Module #1 Disabled
Module #2 Disabled
Attack
Module CPA
Post-Processing
Result Collection
Input Trace Plot
Enabled
Redraw after Each (slower)
Trace Range (0, 1)
Point Range (0, 3000)
Redraw
General Preprocessing Attack Postprocessing Results
Waveform Display
Power Trace View
0 1000 2000 3000
0.4 0.2 0 -0.2 -0.4
Data
Samples
Waveform Display Results Table Output vs Point Plot PGE vs Trace Plot
Script Commands
['Attack', 'Attacked Bytes', 'Byte 11', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 12', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 13', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 14', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 15', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 16', {'visible': False}]
['Attack', 'Attacked Bytes', 'Byte 17', {'visible': False}]
['Attack', 'Attacked Bytes', 'Byte 18', {'visible': False}]
Python Console
>>>
Debug Logging
notes from configfile =
scopeName from configfile = unknown
scopeXUnits from configfile = 0
targetSW from configfile = unknown
targetHW from configfile = unknown
scopeYUnits from configfile = 0
0
1

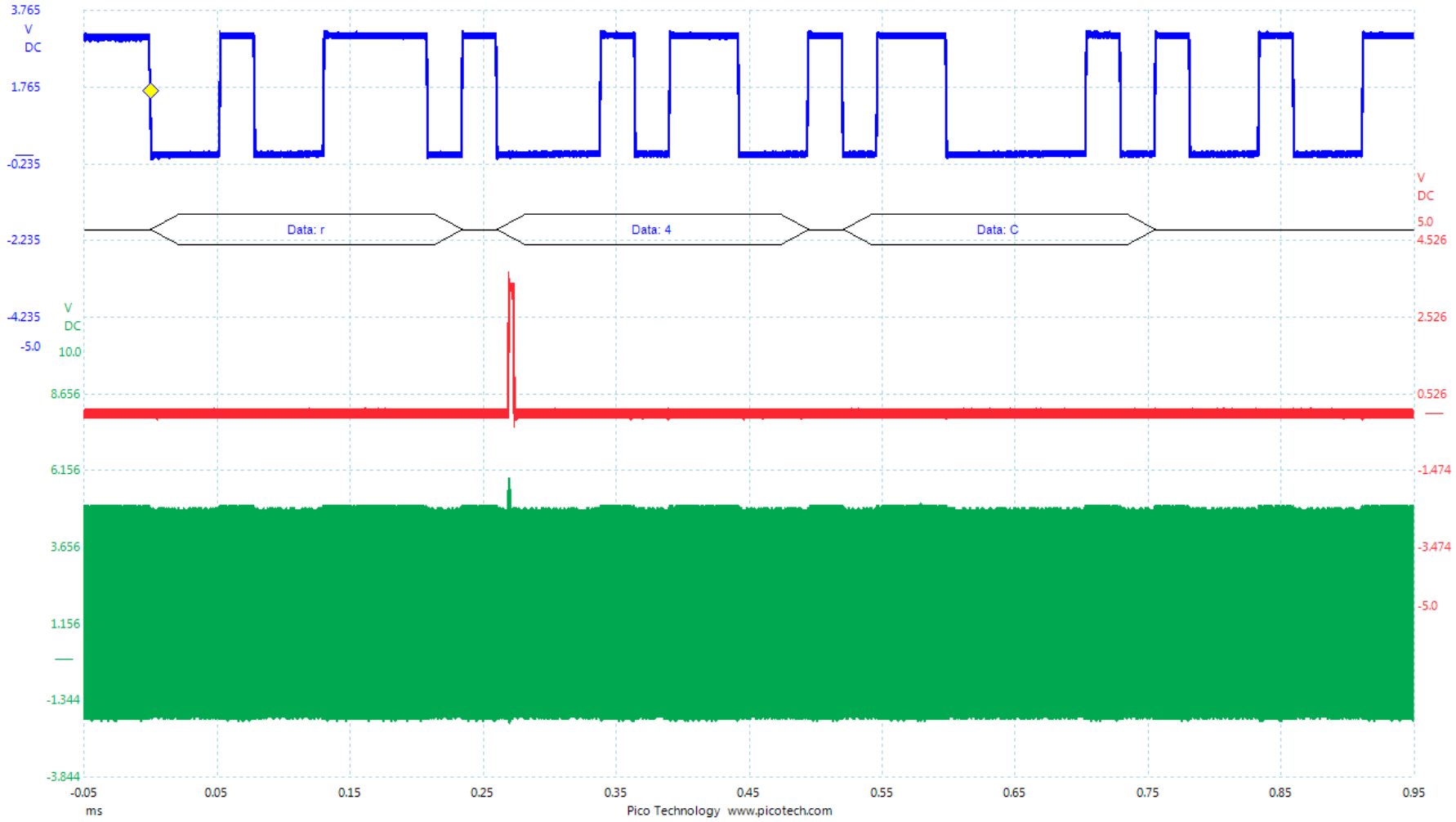

ChipWhisperer Analyzer V2 - teesttt.cwp*
File Project Tools Windows Help
Attack
Parameter Value
Byte 13
Byte 14
Byte 15
Point Setup
Points Same across Subkeys
Starting Point 0
Ending Point 3000
Copy from Output Graph
Copy from Trace Graph
Trace Setup
Starting Trace 0
Traces per Attack 49
Attack Runs 1
Progressive CPA
Reporting Interval 2
Iteration Mode Breadth-First
Skip when PGE=0
General Preprocessing Attack Postprocessing Results
Results Table
0 1 2 3 4 5 6 7 8 9 10 11 12 13
PGE 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF
0.9530 0.9804 0.9596 0.9785 0.9550 0.9673 0.9374 0.9739 0.9605 0.9655 0.9746 0.9744 0.9623 0.949
1 07 8E 08 F1 83 17 99 28 E6 66 79 D4 7E 6F
0.6312 0.6154 0.6472 0.6796 0.6332 0.5844 0.6135 0.6718 0.5794 0.6288 0.6772 0.6337 0.6317 0.615
2 C8 3C E3 71 A7 77 48 6C 86 BA 70 41 02 81
0.5879 0.6148 0.6009 0.6096 0.6296 0.5723 0.5975 0.6448 0.5705 0.5980 0.6087 0.6021 0.6154 0.592
3 F7 D3 2E 68 25 48 F4 4E 7B 07 BB AF 14 EF
0.5816 0.6099 0.5890 0.5874 0.5860 0.5704 0.5781 0.5866 0.5646 0.5898 0.5912 0.5923 0.6059 0.586
4 69 A3 95 E8 E4 B7 52 87 67 67 45 01 40 6D
0.5786 0.6015 0.5862 0.5799 0.5743 0.5650 0.5731 0.5731 0.5628 0.5884 0.5907 0.5910 0.5950 0.579
5 AF 8A 82 64 18 E8 72 0C CE C4 07 A4 77 05
0.5743 0.6010 0.5848 0.5621 0.5724 0.5591 0.5689 0.5663 0.5602 0.5864 0.5889 0.5817 0.5850 0.577
6 3A A6 45 F9 67 22 A6 88 02 81 E4 99 1C 96
0.5697 0.5997 0.5833 0.5573 0.5683 0.5553 0.5675 0.5645 0.5601 0.5847 0.5826 0.5753 0.5691 0.575
7 76 62 C1 5A DC 5D E2 E4 A1 FD 7F 84 6F 92
0.5642 0.5943 0.5822 0.5513 0.5646 0.5525 0.5647 0.5614 0.5590 0.5828 0.5728 0.5753 0.5641 0.572
BD AA 34 18 94 F5 CF 6B 8D FC D1 F4 99 46
Waveform Display Results Table Output vs Point Plot PGE vs Trace Plot
Script Commands
['Attack', 'Attacked Bytes', 'Byte 10', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 11', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 12', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 13', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 14', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 15', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 16', {'visible': False}]
['Attack', 'Attacked Bytes', 'Byte 17', {'visible': False}]
['Attack', 'Attacked Bytes', 'Byte 18', {'visible': False}]
Python Console
>>>
Debug Logging
diffs[key] = sumnum / np.sqrt(sumden)
c:
\users\colin\workspace\chipwhisperer\chipwhisperer\software\chipwhisperer\analyzer\attacks\CPAProgressive.py
:178: RuntimeWarning: invalid value encountered in
divide
diffs[key] = sumnum / np.sqrt(sumden)
C:\Python27\lib\site-


Output vs Point Plot
Power Trace View
0
1000
2000
3000
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Data
Samples
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 All On All Off
Waveform Display
Results Table
Output vs Point Plot
PGE vs Trace Plot


Average Partial Guessing Entropy (PGE) with Oscilloscope
Average PGE (50 Trials)
Trace Number
0
20
40
60
80
100
120
140
160
180
0
5
10
15
20
25
Subkey 0
Subkey 1
Subkey 2
Subkey 3
Subkey 4
Subkey 5
Subkey 6
Subkey 7
Subkey 8
Subkey 9
Subkey 10
Subkey 11
Subkey 12
Subkey 13
Subkey 14
Subkey 15

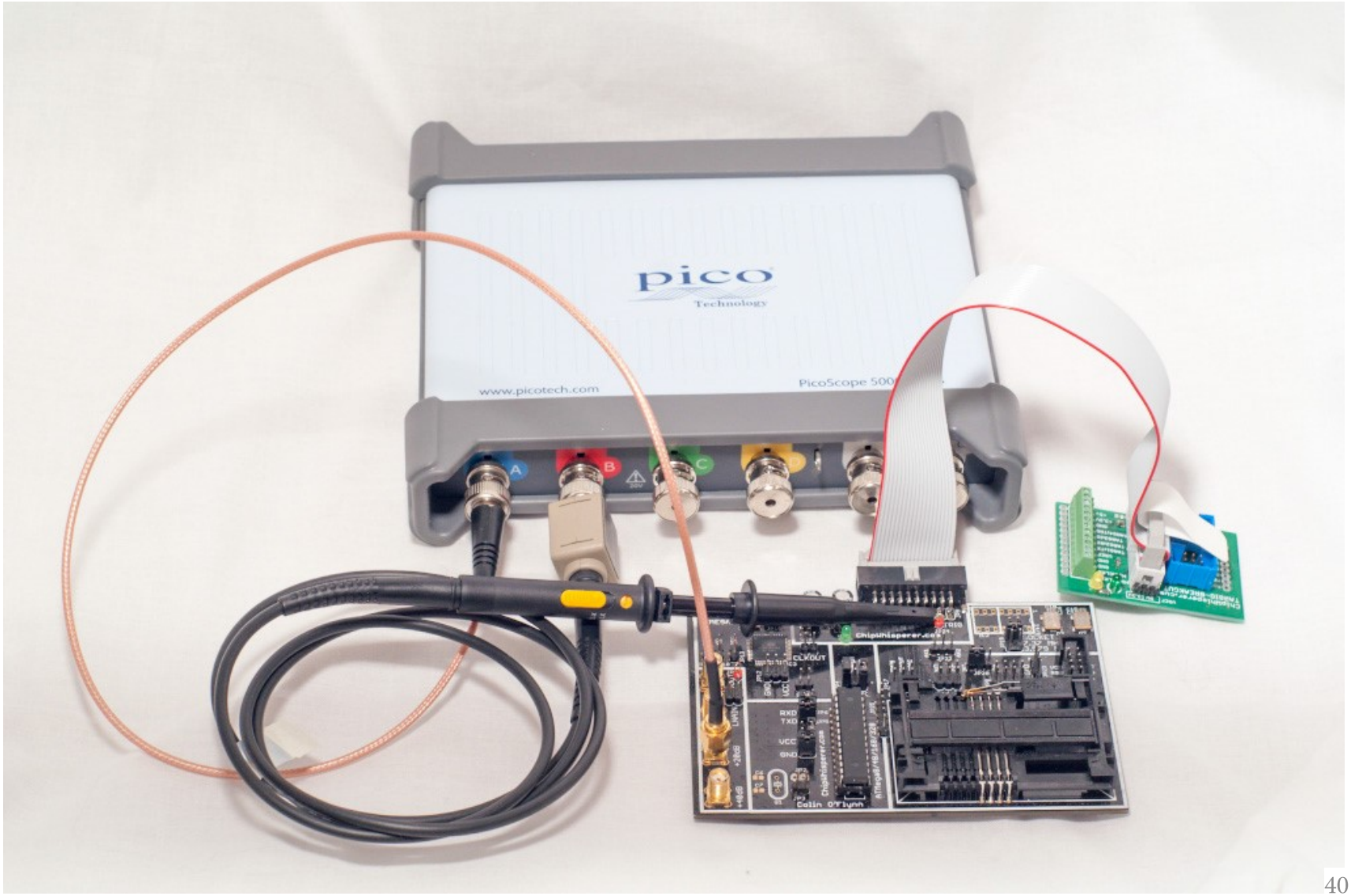
USAGE EXAMPLES

Example: Glitching on UART Byte

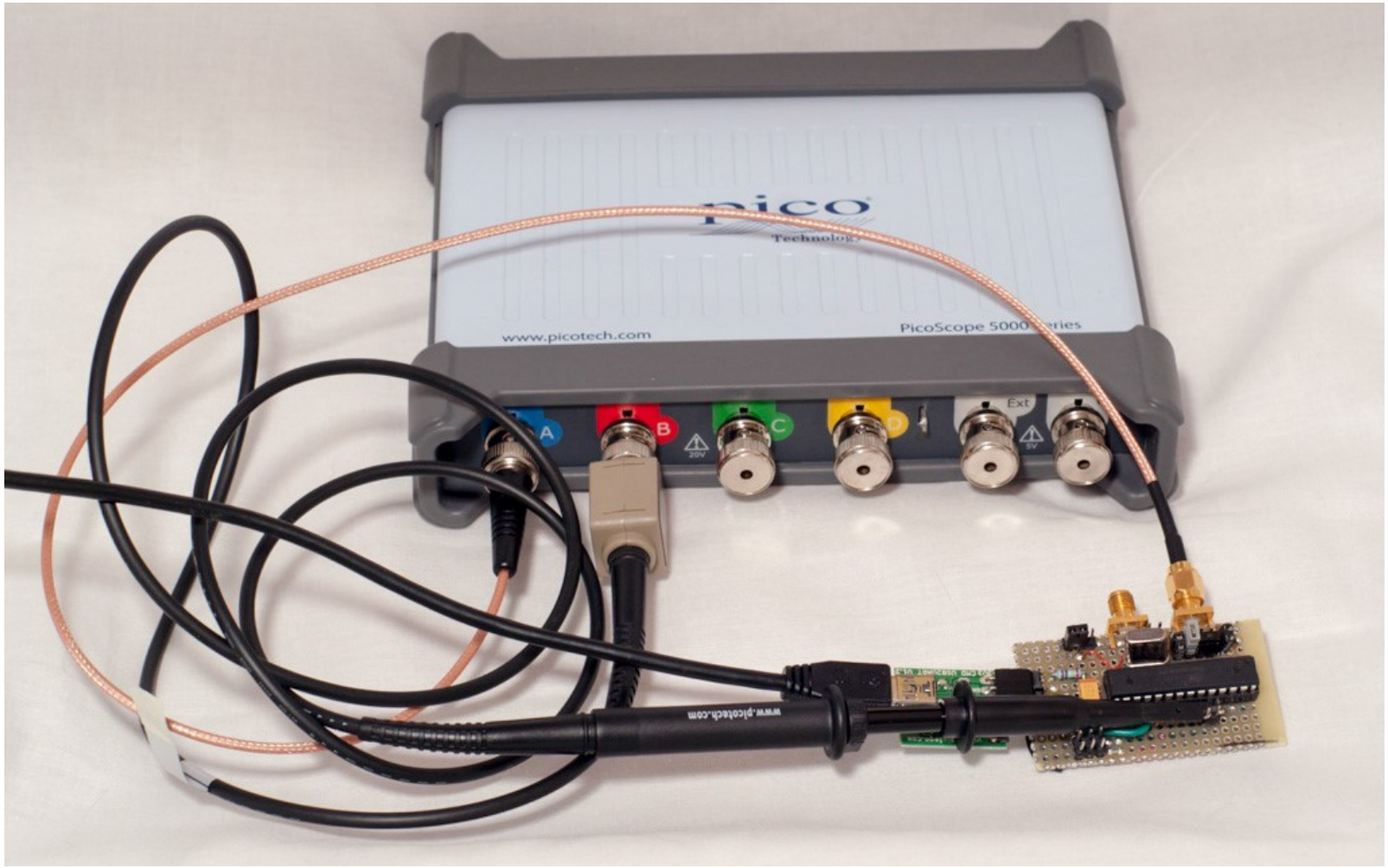


Example: Glitching on UART Byte

PicoScope Version #1



PicoScope Version #2



Capture Speed Results

Capture Hardware	Attack Target	Target Connection	Points /Trace	Traces/Second Win7	Linux
ChipWhisperer Rev2	SASEBO-GII	USB	100	14.8	28.3
ChipWhisperer Rev2	AVR, 38400 Baud	FPGA-Serial	3000	11.3	3.91
ChipWhisperer Rev2	AVR, 38400 Baud	FPGA-Serial	20000	7.04	3.78
ChipWhisperer Rev2	AVR, 38400 Baud	USB-Serial	3000	18.2	18.9
ChipWhisperer Rev2	SmartCard	PS/SC Reader	3000	7.40	6.62
SAKURA-G	SAKURA-G	Integrated	400	6.67	7.18
SASEBO-W	SmartCard	FPGA-USI	3000	0.271	0.279
SASEBO-W	SmartCard	FPGA-SmartCard	3000	1.49	1.52
Agilent MSO54831D	SASEBO-GII	USB	1500	8.01	—
PicoScope 6403D	SASEBO-GII	USB	1500	12.1	43.6
PicoScope 6403D	SAKURA-G	USB	1500	15.4	29.6
PicoScope 5444B	AVR, 38400 Baud	USB-Serial	12000	16.4	5.63

Questions?

Code: www.ChipWhisperer.com

Docs: www.newae.com/sidechannel/cwdocs

Contact Me: coflynn@newae.com