

Mark Ermolov Maxim Goryachy

POSITIVE TECHNOLOGIES

Intel VISA:

Through the Rabbit Hole



The methods described here are risky and may damage or destroy your computer. We take no responsibility for any attempts inspired by our work and do not guarantee the operability of anything.



blackhat Warning ASIA 2019

- All information provided here is a result of public software, firmware, and hardware analysis. This research may contain true mistakes and inaccuracies.
 - 2. Our access to Intel VISA is made possible by the Intel-SA-00086 vulnerability that we detected, but you can try other methods.;)
 - 3. If you find mistakes, please let us know (email or Twitter).





Deep Intel CPU (Our View)

```
Public Documentation
                                   X86 Instruction set
                                     SMM
                                   MP HT AVX
                                               VT-d
                                      TSX
Partial Documentation
                                 TraceHub
                                              BootGuard
                         MSR Regs UPI VPro
                                                      Probe Mode
                                  CAR PMC Out-of-Order
                                System Agent(SA) ODLA NEB
                                        FUSEs Intel VISA
Private Documentation
                                   IOSF
                                        Side-Band
                                    UNCORE
                                      SAI
                                       Microcode
```

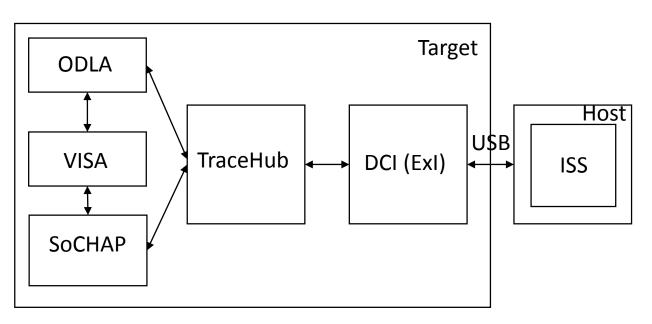


Intel VISA should primarily interest researchers:

- Low-level access to CPU signals on the customer's platform;
- Study of speculative execution and out-of-order;
- Reconstruction of internal architecture.

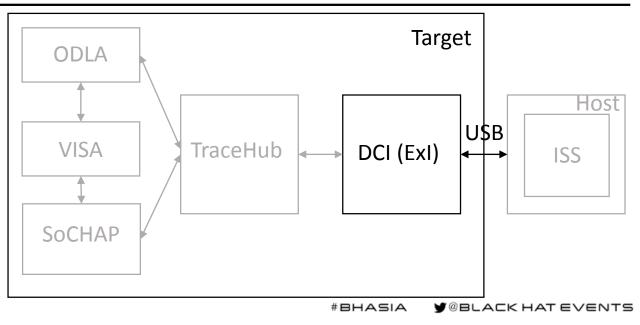
blackhat Agenda ASIA 2019

- Intel Debugging Tools Overview
- What is Intel Trace Hub
- Built-in Logic Analyzer for CPU and PCH
- The Secrets of Intel System Studio
- A Few Words About Security



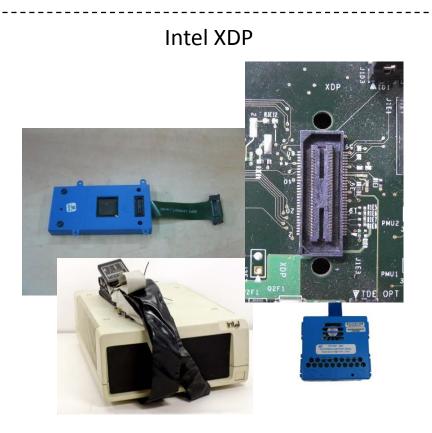


Intel Debugging Tools Overview





blackhat Intel Debugging Tools: Evolution



VS

Intel Direct Connect Interface (DCI)

Proprietary socket, specialized mainboard

USB, consumer mainboard

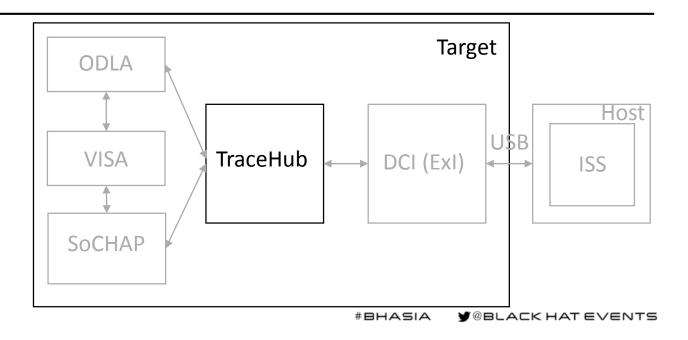


- Intel DFx Abstraction Layout
- Check PCH GPIO pins
- Check DDR memory
- Check USB devices
- Check Direct Media Interface (DMI)
- Check CPU high precision event timer
- Check PCH PCIe bus/drivers
- Check PCH SMBus /devices

•



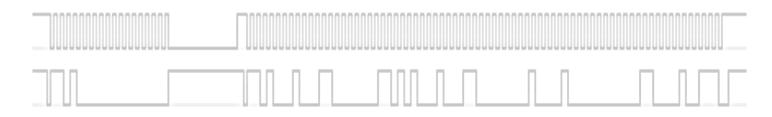
Intel Trace Hub





khat Intel PCH/SoC Hardware Tracing

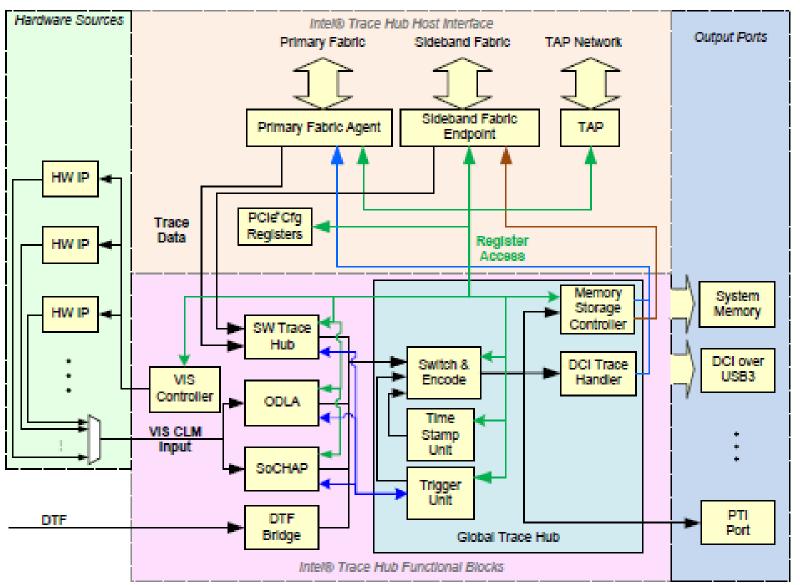
- Integrated into Trace Hub.
- Consists of Intel VISA, ODLA, SoCHAP.
- Controlled by the Trace Hub MMIO CSR range.
- Uses Trace Hub components for trace data collection.
- Uses Common Trigger Sequencer for triggering capabilities.
- Supports all Trace Hub transports (DCI BSSB, DbC, HTI, and PTI via GPIO).



blackhat Intellrace Hub

- A new Intel approach to collection of all trace data in a platform (since Skylake).
- The internal code name is North Peak (NPK).
- Supports software, firmware, and hardware tracing.
- Dedicated IP unit accessible by PCI, IOSF Side Band, JTAG.
- Consists of Trace Sources (Masters) and Global Trace Hub.
- Encodes trace data into the MIPI STPv2 protocol.
- Collects data into RAM, sends it to HW interfaces.
- Supports Intel Processor Trace and Architectural Events Tracing.
- Has Common Trigger Sequencer (CTS) for flexible control of trace masters.

blackhat Intellrace Hub





#BHASIA

¥@BLACK HAT EVENTS

blackhat Intel Trace Hub Software

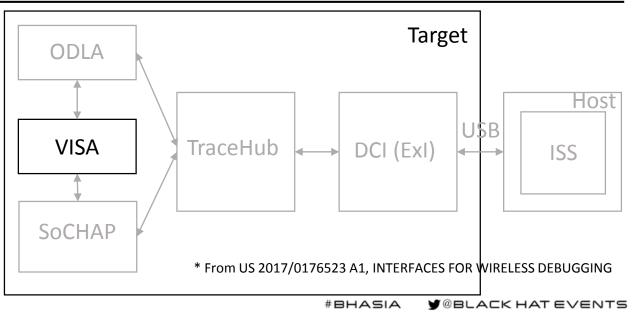
- Software stack of two libraries: Trace Hub API and Trace Decode Engine.
- Each library is provided with native tools: npk_cfg64.exe and tde_boot.exe.
- Both libraries are configured by XMLs: npk.xml, visa.xml, tde.xml.
- Trace Hub API uses special Interview XML for HW events description.
- Trace Hub API serializes its state to *.tracecfg JSON.
- The software is installed as part of the Intel System Trace tool.
- There isn't any public (non-NDA) info about the libraries.
- The Linux version of the libraries has unstripped (function names) binaries.



CSME Tracing



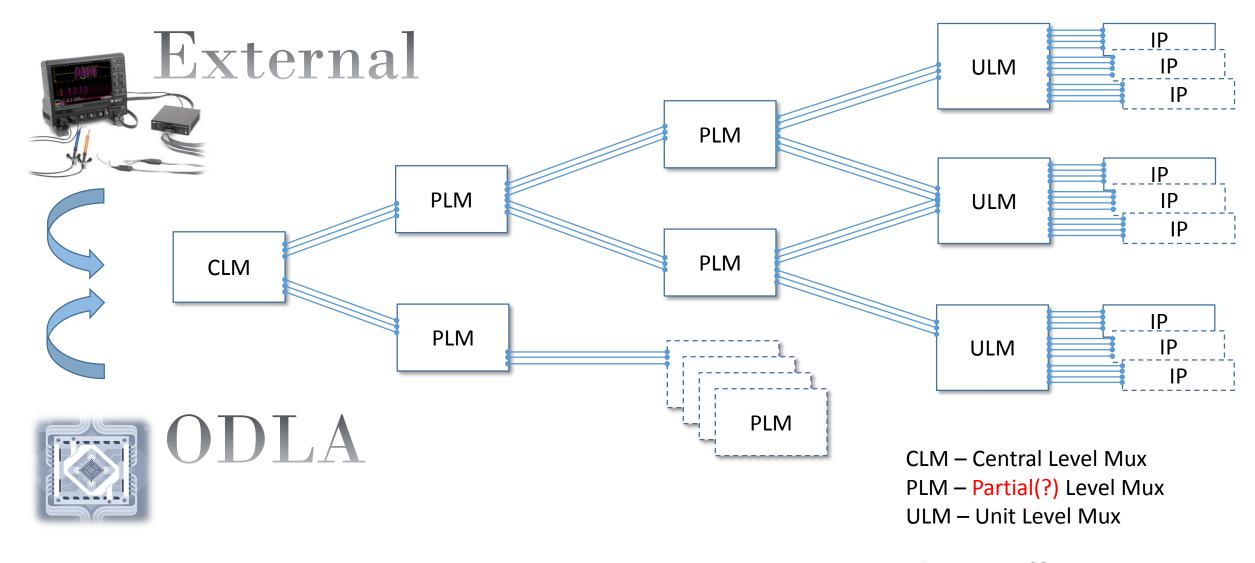
Intel Visualization of Internal Signals Architecture (VISA*)





- Stands for Visualization of Internal Signals Architecture.
- A network of hundreds of multiplexers routing debug signals to ODLA.
- Each IP unit selects a set of signals to be routed to VISA.
- Dedicated VISA Registers Controller (VRC) managing the network.
- Three types of multiplexers: CLM, PLM, and ULM.
- VISA uses native internal clocks.
- VISA can route IP signals and native clocks to external Logic Analyzer.
- Each multiplexer supports an advanced pattern generator.

blackhat Intel VISA Schematic



ackhat Intel VISA Registers Controller (VRC)

- VRC defines fixed MMIO ranges for each Mux register.
- Each Mux has one control register for every output lane.
- VRC defines selectors for native clocks and for IP units signal groups.
- ULM lanes transmit one native clock and a group of IP signals.
- CLM and PLM lanes transmit lanes of other PLMs and ULMs.
- CLM and some ULMs have special xbar lanes for CTS and SoCHAP.
- VRC controls access to lane control registers.
- VRC has Replay RAM for automatic Intel VISA reconfiguration.

Multiplexor descriptor

```
<mux name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/chfuscntl ulm" num lanes="2" num xbar lanes="0"</pre>
   <output lane num="0" output lane num="0" />
   <output lane num="1" output lane num="1" />
   <clock clk="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/fuse ro clk gated" sel="0" ss="0" />
   <clock clk="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/fuse ref clk gated" sel="1" ss="0" />
   <clock clk="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/ftap tck" sel="2" ss="0" />
</mux>
                                                          IP unit signals group descriptor
<group clk="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/fuse ro clk gated" name="/pch/parleg/parleg pwell</pre>
   <mux path clk sel="0" lane sel="20" mux name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/chfuscntl ulm'
   <mux path clk sel="4" lane sel="5" mux name="/pch/parleg/parleg pwell wrapper/plm parleg/plm" />
   <mux path clk sel="0" lane sel="1" mux name="/pch/parlpss/parlpss pwell wrapper/plm parlpss/plm" />
   <mux path clk sel="0" lane sel="1" mux name="/pch/parsdx/parsdx pwell wrapper/plm parsdx/plm" />
   <mux path clk sel="8" lane sel="9" mux name="/pch/parcavshub/parcavshub pwell wrapper/plm parcavshub/plm" />
   <mux path clk sel="0" lane sel="1" mux name="/pch/parnpk/parnpk pwell wrapper/plm parnpk 1/plm" />
   <mux path clk sel="0" lane sel="1" mux name="/pch/parnpk/parnpk pwell wrapper/clm/clm" />
   <signal bit="0" name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/i fuse mem if arb/gfsm outaddr[0]" />
   <signal bit="1" name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/i fuse mem if arb/gfsm outaddr[1]" />
   <signal bit="2" name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/i fuse mem if arb/gfsm outaddr[2]" />
   <signal bit="3" name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/i fuse mem if arb/gfsm outaddr[3]" />
   <signal bit="4" name="/pch/parleg/parleg_pwell_wrapper/fuse top1/i chassis fuse controller top/i fuse mem if arb/gfsm outaddr[4]" />
   <signal bit="5" name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/i fuse mem if arb/gfsm outaddr[5]" />
  <signal bit="6" name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/i fuse mem if arb/gfsm outaddr[6]" />
   <signal bit="7" name="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/i fuse mem if arb/gfsm outaddr[7]" />
</group>
```

Registers descriptor

<register mux="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/chfuscntl ulm" offset="0x21F80" standard="visa ctrl">VISA Control <register mux="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/chfuscntl ulm" offset="0x21F84" reg num="0" standard="visa lane"> <register mux="/pch/parleg/parleg pwell wrapper/fuse top1/i chassis fuse controller top/chfuscntl ulm" offset="0x21F88" reg num="1" standard="visa lane">



blackhat Our collection (Intel VISA XMLs)

45	Α	<u> </u>	15	

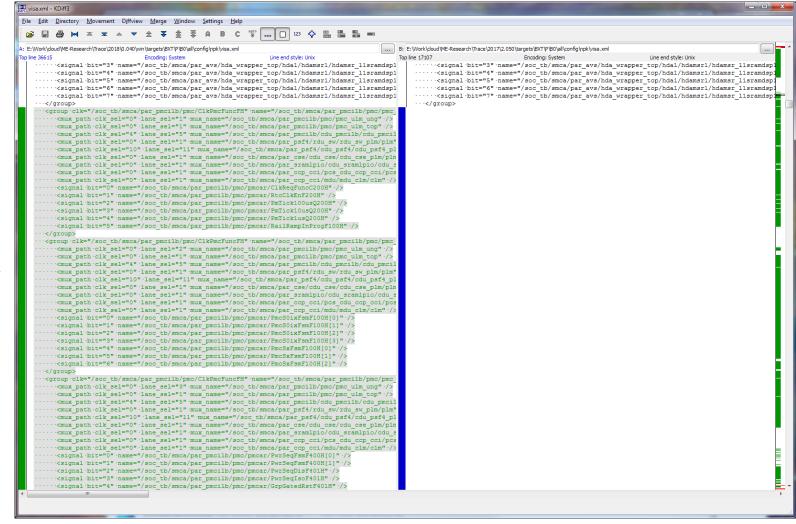
Name	Size
E:\visa\Trace\2019\0.050\win\targets\CNP\WHL\D0\green\config\npk\visa.xml	82610604
E:\visa\Trace\2019\0.050\win\targets\CNP\CFL-S\B1\green\config\npk\visa.xml	82610604
E:\visa\Trace\2019\0.033\win\targets\CNP\WHL\D0\green\config\npk\visa.xml	82610604
E:\visa\Trace\2019\0.033\win\targets\CNP\CFL-S\B1\green\config\npk\visa.xml	82610604
E:\visa\Trace\2019\0.014\win\targets\CNP\CFL-S\B1\green\config\npk\visa.xml	82610604
E:\visa\Trace\2018\0.051\targets\BXT\P\D0\all\config\npk\visa.xml	43068808
E:\visa\Trace\2018\0.051\targets\BXT\P\B0\all\config\npk\visa.xml	43068808
E:\visa\Trace\2018\0.040\win\targets\BXT\P\D0\all\config\npk\visa.xml	43068808
E:\visa\Trace\2018\0.040\win\targets\BXT\P\B0\all\config\npk\visa.xml	43068808
E:\visa\Trace\2018\0.040\lin\targets\BXT\P\D0\all\config\npk\visa.xml	43068808
E:\visa\Trace\2018\0.040\lin\targets\BXT\P\B0\all\config\npk\visa.xml	43068808
E:\visa\Trace\2018\0.051\targets\LBG\SKX\B1\green\config\npk\visa.xml	35177433
E:\visa\Trace\2018\0.051\targets\LBG\SKX\B0\green\config\npk\visa.xml	35177433
E:\visa\Trace\2018\0.040\win\targets\LBG\SKX\B1\green\config\npk\visa.xml	35177433
E:\visa\Trace\2018\0.040\win\targets\LBG\SKX\B0\green\config\npk\visa.xml	35177433
E:\visa\Trace\2018\0.028\targets\LBG\SKX\B1\green\config\npk\visa.xml	35177433
E:\visa\Trace\2018\0.028\targets\LBG\SKX\B0\green\config\npk\visa.xml	35177433
E:\visa\Trace\2018\0.051\targets\LBG\SKX\A0\green\config\npk\visa.xml	35050304
E:\visa\Trace\2018\0.040\win\targets\LBG\SKX\A0\green\config\npk\visa.xml	35050304
E:\visa\Trace\2018\0.028\targets\LBG\SKX\A0\green\config\npk\visa.xml	35050304
E:\visa\Trace\2018\0.051\targets\LBG\SKX\S0\green\config\npk\visa.xml	32140480
E:\visa\Trace\2018\0.040\win\targets\LBG\SKX\S0\green\config\npk\visa.xml	32140480
E:\visa\Trace\2018\0.028\targets\LBG\SKX\S0\green\config\npk\visa.xml	32140480
E:\visa\Trace\2018\0.051\targets\KBP\X\H0\green\config\npk\visa.xml	21603327
E:\visa\Trace\2018\0.051\targets\KBP\W\M0\green\config\npk\visa.xml	21603327
E:\visa\Trace\2018\0.051\targets\KBP\H\A0\green\config\npk\visa.xml	21603327



blackhat Multiplexers & Groups

Platform	Count	Туре	Version
CNP	780	mux	2019
BXT	390	mux	2018
GLK	837	mux	2018
GLK	4022	group	2018
BXT	17095	group	2018
BXT	4822	group	2017







blackhat Intel VISA Control Registers

visa_lane / visa_clm_lane

Field	Bits
sel	07
clksel	815
bypass	16
fsphsel	1719
vsgmode	2021
dskgrp	2528
vsyncen	29
dsken	30
div2	31

visa_slide

Field	Bits	
sel	07	
clksel	815	
bypass	16	
window_slide	2022	
alt_sel	2431	

visa_stepdown

Field	Bits	
sel	07	
clksel	815	
bypass	16	
stepdown_en	17	
subrange	1819	

visa_ctrl

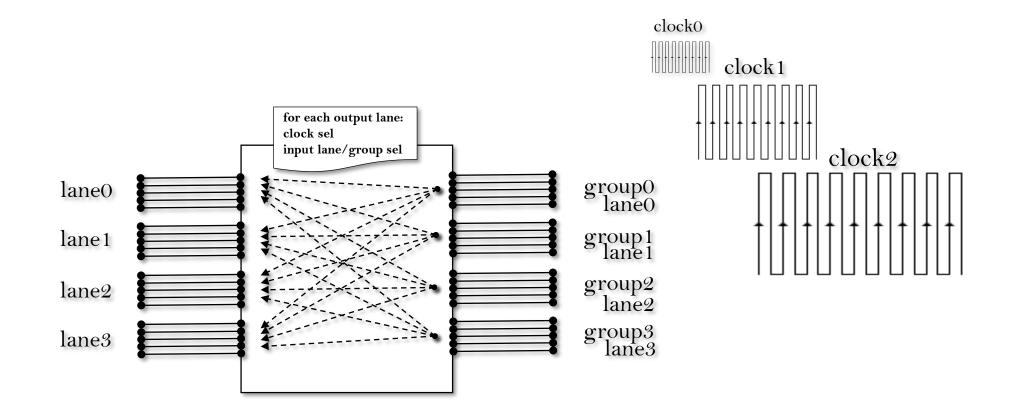
Field	Bits
enable	0
pgmode	47

visa_xbar

Field	Bits
bit_sel	02
byte_sel	815
bypass	9
bit_sel	1012
byte_sel	1318
bypass	19
bit_sel	2224
byte_sel	2530
bypass	31

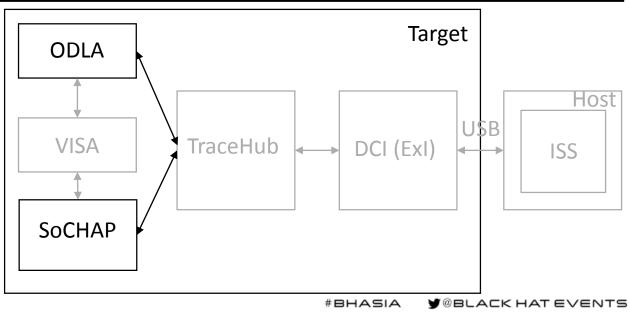


blackhat Intel VISA Multiplexer





Intel On-Die Logic Analyzer (ODLA)



blackhat Intel ODLA

- Internal sampling and timestamp generation at ~400 MHz.
- Uses native SoC/PCH clocks to validate lanes (clock-off detection).
- Can sample up to 16 lanes (128 signals) per clock.
- Has per-lane and global buffer resources.
- Arbitrates lanes based on 2-bit weights.
- Supports store-on-change compression.
- Encodes sampled signal data into a private format.
- Sends encoded data to GTH (Master ID = 1) to be wrapped into STP format.
- TDE ODLA decoder log (in debug mode) allows analyzing the private format.

blackhat TD=ODLA Output Example

```
$timescale
  1ps
$end
Scomment
   Overloaded symbol values:
     u = CLOCK OFF
      w = FILTERED VALUE
      - = BUFFER OVERFLOW
      z = UNKNOWN TRANSITION
$scope module VISA $end
  $var wire 1 A trace trigger $end
$scope module default $end
  $var wire 8 E lane3[7:0] $end
  $var wire 8 D lane2[7:0] $end
  $var wire 8 C lane1[7:0] $end
  $var wire 8 B lane0[7:0] $end
Supscope Send
Senddefinitions Send
b0 A
b0 B
b0 C
b0 D
b0 E
#163802500
b0 B
b0 C
b0 D
b0 E
#4150817035000
b11110101 D
b111 E
#4150817037500
b0 B
b0 C
#4150817052500
b10110111 B
b10100100 C
#4150817055000
#18446567560093026615
b0 D
b0 E
#8710235452500
bx B
bx C
bx D
bx E
#8710235455000
bu B
bu C
bu D
bu E
```

- Default file name capture_vcd_decoder.vcd.
- Placed at the path of the tde_boot64.exe session XML.
- Contains a timestamp in ps for each sample interval.
- Displays a special 1-bit lane A trigger.
- Lanes data are only in binary format.
- Marks special conditions with the symbols **u**, **w**, -, **x**, **z**.

blackhat ODLA Packets Log

```
RefNo:3 TS:INJ BO:F GDC:T Trig:F MemWrp:F TS:0x00000a03c800
PacketNdx: 0x00014a0e f84000000a03c800 G TS COUNTER PACKET
PacketNdx: 0x00014a0f f9001db1db0000000 G SAMPLE INDEX PACKET RefNo:3 TS:INJ L0:000 L1:000 L2:1db L3:1db
PacketNdx: 0x00014a10 fa00fff100fff100 G DC SAMPLE PACKET
                                                             RefNo:3 TS:INJ LO CLK:fff1 L1 CLK:fff1 L0 VAL:00 L1 VAL:00
PacketNdx: 0x00014a11 980000000a07f56f G TS COUNTER PACKET
                                                             RefNo:0 TS:INJ BO:F GDC:F Trig:F MemWrp:F TS:0x00000a07f56f
PacketNdx: 0x00014a12 9900001001000000 G SAMPLE INDEX PACKET RefNo:0 TS:INJ L0:000 L1:000 L2:001 L3:001
PacketNdx: 0x00014a13 027f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:T D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a14 037f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:T D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a15 9a002d6f002d6f00 G DC SAMPLE PACKET
                                                             RefNo:0 TS:INJ LO CLK:2d6f L1 CLK:2d6f L0 VAL:00 L1 VAL:00
                                                             RefNo:0 TS:INJ L2 EDC L3 EDC L2 CLK:2d6d L3_CLK:2d6d L2_VAL:00 L3_VAL:00
PacketNdx: 0x00014a16 9a072d6d002d6d00 G DC SAMPLE PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a17 023f00e100e100e1 L DATA PACKET
PacketNdx: 0x00014a18 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a19 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a1a 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a1b 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a1c 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a1d 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a1e 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a1f 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a20 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a21 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a22 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a23 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a24 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a25 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a26 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a27 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a28 033f000300030003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a29 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a2a 033f000300030003 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a2b 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a2c 033f000300030003 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:001 D2:e1 C:001 D3:00 C:001 D4:e1 C:001 D5:00 C:001
PacketNdx: 0x00014a2d 023f00e100e100e1 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:001 D2:03 C:001 D3:00 C:001 D4:03 C:001 D5:00 C:001
PacketNdx: 0x00014a2e 033f000300030003 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:e1 C:001 D1:00 C:201
PacketNdx: 0x00014a2f 022b00ff00ff00e1 L DATA PACKET
PacketNdx: 0x00014a30 032b00ff00ff0003 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:03 C:001 D1:00 C:201
PacketNdx: 0x00014a31 023ae100e15200ff L DATA PACKET
                                                             RefNo:0 L2 TSI:F C:0ff D1:00 C:053 D2:e1 C:001 D3:00 C:001 D4:e1 C:001
PacketNdx: 0x00014a32 033a0300035200ff L DATA PACKET
                                                             RefNo:0 L3 TSI:F C:0ff D1:00 C:053 D2:03 C:001 D3:00 C:001 D4:03 C:001
PacketNdx: 0x00014a33 023fe100e100e100 L DATA PACKET
                                                             RefNo:0 L2 TSI:F D0:00 C:001 D1:e1 C:001 D2:00 C:001 D3:e1 C:001 D4:00 C:001 D5:e1 C:001
PacketNdx: 0x00014a34 033f030003000300 L DATA PACKET
                                                             RefNo:0 L3 TSI:F D0:00 C:001 D1:03 C:001 D2:00 C:001 D3:03 C:001 D4:00 C:001 D5:03 C:001
```

- Time stamp injectionLanes data
- Clock-off example

blackhat ODLA Tracing: Step by Step

- ✓ Ensure that you have *npk.xml* and *visa.xml* for your platform.
 - Check the following path: <Program Files>\IntelSWTools\system_debugger_[version]\system_trace\targets\[your platform]\[stepping]\[green|all]\config\npk
- ✓ Create *interview.xml* describing the desired hardware events.

 See the next slide for an example of interview.xml.
- ✓ Create *tde.xml* describing TDE session parameters.

 Hint: The System Trace tool from System Studio can do it for you.
- ✓ Run *tde_boot64.exe* with the appropriate tde.xml.
- ✓ Run npk_cfg64.exe to start a trace session.
 Example command:
 npk_cfg64.exe --ipcapi --xml-file <npk.xml> --access-profile TAP_ONLY --trace-src HW --dest-port BSSB --event-list /Events//event1
- ✓ Wait for or reproduce the hardware event.
- ✓ Stop the trace session by pressing <*Ctrl+C>* in the *npk_cfg* console.
- ✓ Stop the trace decode session by pressing <*Ctrl+C>* in the *tde_boot* console.
- ✓ Check capture_vcd_decoder.vcd in the trace session folder (near tde.xml).

blackhat Interview.xml

* Expressions contain full signal names of output groups from visa.xml

blackhat Intel SoCHAP

- Stands for SoC CHAP (Chipset Hardware Architecture Performance).
- Designed to count HW events based on signal patterns.
- Samples signals from Intel VISA xbars at the rate of hundreds of MHz.
- Uses native clocks for lanes clock-off detection.
- Up to 16 mask/match registers (HW events).
- Up to 16 counters with advanced control (inc/dec, preload).
- Can count event occurrence or event duration.
- Counters can be read from MMIO registers or sent to GTH (Master ID = 2).



What we've got

Intel On-Chip System Fabric (IOSF)

Skhat Intel On-Chip System Fabric

- Intel private IP units on-die interconnect specification.
- Primarily designed for SoCs and later carried over to PCH.
- Allows interchangeable IP units design.
- Allows seamless third-party IP units integration.
- Defines three types of interconnect buses: Primary, Side Band, and DFx (JTAG).
- Defines a bridge specification for the OCP (Open Core Protocol) bus.
- Defines bridges for legacy/external buses: PCI-E, SPXB, NEB, FSB.
- Uniquely identifies each IP unit by Security Attribute of Initiator (SAI).
- Defines access control rules for IPs based on SAI.
- Propagates SAI between all buses.

blackhat IOSE Primary Bus

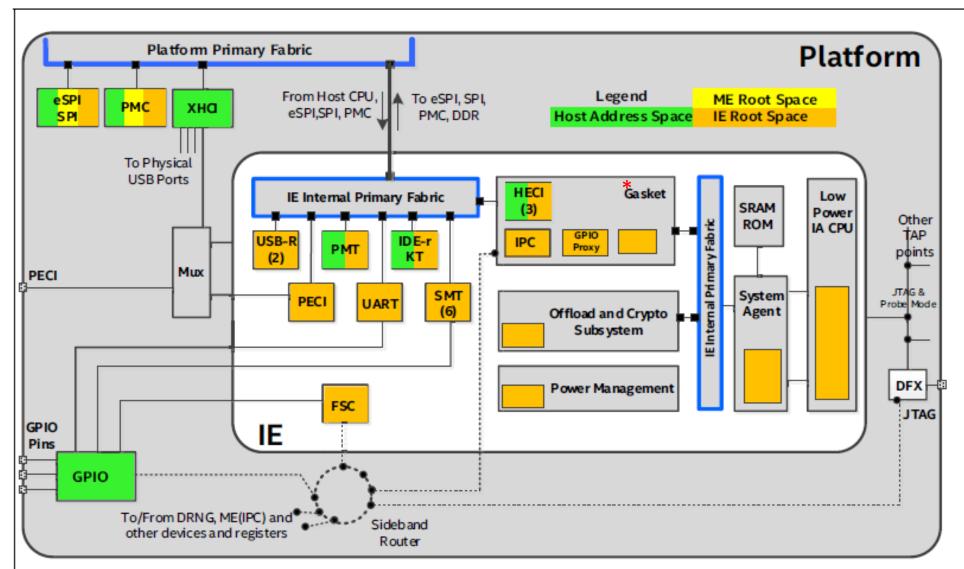
- Fast-speed, parallel bus for primary data flows within a platform.
- Greatly extends the PCI-E transaction layer specification.
- Variable parameters within each segment (bus width, frequency).
- Based on Primary Scalable Fabric (PSF) IP unit.
- PSF routes transactions between upstream and downstream ports.
- Supports different root spaces with independent address ranges.
- Supports point-to-point transaction of IP units.
- Supports different decode (routing) modes (address, source, target).
- Despite BDF, each IP unit is addressable by Dest ID (PSF#, port group, and port ID).
- Intel CSME has its own bridge (ATT) to IOSF Primary that allows specifying Dest ID.
- PSF allows controlling VTd for its transaction in each root space.

blackhat IOSF Side Band Bus

- Low-speed bus for all out-of-band communications.
- Used for private configuration, fuse distribution, error reporting.
- Uniquely identifies each IP unit by global Port ID.
- There are multiple Side Band Routers (SBR) that make up the bus.
- Defines opcodes to access the IPs configuration space, IO/Memory BARs.
- Defines an opcode for the special messaging (private configuration) address space.
- Some IP units (for example, fuse controller) support custom opcodes.
- Has Primary-to-Side-Band (P2SB) and DFx-to-Side-Band (TPSB) bridges.
- IOSF Primary PSF units are controlled by SB.
- Different SoCs/PCH unlock modes give different access levels to SB via TPSB.
- A P2SB bridge can give special access to SB for devices with a special SAI (for example, TAM).
- Intel CSME has full access to SB (via its own ATT bridge with a privileged SAI).
- DFx Aggregator, a special device managing all platform security, is controlled by SB.

ASIA 2019

blackhat OSF example for LBC Chipset



Buses

Primary

Side Band

DFx

External

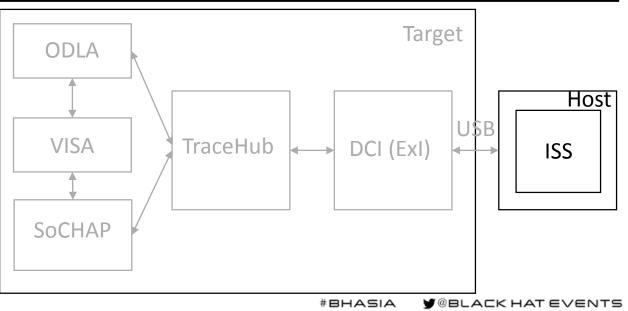
*IE Gasket has implied bridges



FUSEs



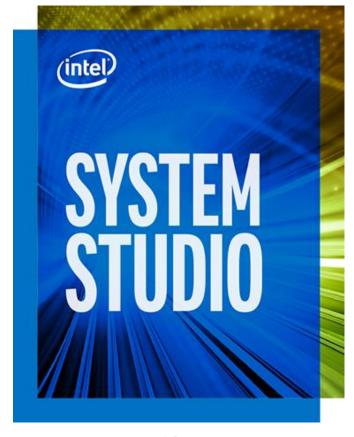
The Secrets of the Public Version of Intel System Studio





Intel System Studio Ultimate Trial Version (not available now???)

- Intel DAL
- Intel Debugging Tools
- Intel OpenIPC
- Intel System Trace

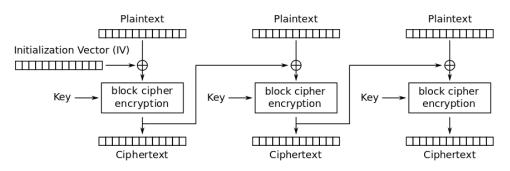


black hat

TDef Binary Files (DAL)

```
37 08 E5-5B 1B DA A4-17 30 7C 2C-C1 78 C1 90
                                                        /7<mark>•</mark>σ[←rñ⊉0], <sup>⊥</sup>x-
                                                                                          FB 37 08 E5-5B 1B DA A4-17 30 7C 2C-C1 78 C1 90
                                                       <sup>1</sup>Θο¼@-4∞Γ sífπ<sub>1</sub>ì
                BE 17 EC-E2 DD 73 A1-66 E3 BB 8D
                                                                                          36 A4 D5 1A-90 51 AD 9B-72 46 07 FA-7A BF 25 CB 6ñ → ÉQ; ¢rF•·
                                                       ¢<sup>⊥</sup>-ß∞↑zìäs♥R<sup>⊥</sup>vED
                                                                                          47 D5 51 1D-B4 C1 CF E4-ED 40 A0 CC-18 64 1F 33 G<sub>F</sub>Q+ ΔΣΦ@ά
       16 E1-EC 18 7A 8D-84 73 03 52-C1 79 45 44
                                                       ò₩z²$ùn↑ k1:4 kdé
                                                                               00000040: EC 22 65 DD-05 A9 A3 54-CE A9 B6 D0-F9 25 4B <mark>E6</mark>
                                                                               00000050: 47 71 96 07-52 1A 57 41-B2 4C 79 C7-DC D0 D7 22
                                                       ÷τ-δβFuI lo eQ- rm
F6 E7 A9 EB-0E 46 E6 49-BD F8 DF 65-0C B4 10 6D
7C 58 CC AD-B9 E5 AA 34-36 CB 8F 2B-DA F1 50 4A
                                                       |X|_1 + \sigma_46\pi A + r \pm PJ
76 33 1F 90-E3 7D 00 3D-19 82 62 88-85 9B 18
ED 32 29 CO-D5 A1 04 C9-4D F0 8C C9-11 21 64 D7
FB 37 08 E5-5B 1B DA A4-17 30 7C 2C-C1 78 C1 90
                                                        √7°σ[← rñ$0], <sup>⊥</sup>x<sup>⊥</sup>l
                                                                                              01 09 AC-40 BE 17 EC-E2 DD 73 A1-66 E3 BB 8D
                                                        90%@ $∞Γ sífπa
BC 01 09 AC-40 BE 17 EC-E2 DD 73 A1-66 E3 BB 8D
                                                                                          60 44 3D 30-55 F5 E9 42-2E A2 1C 91-96 87 E1 9F
9B CA 16 E1-EC 18 7A 8D-84 73 03 52-C1 79 45 44
                                                        ¢<sup>⊥</sup>-ß∞↑zìäs♥R<sup>⊥</sup>vED
                                                                                          52 03 47 D1-8B 22 F7 7F-2D CB 1A E7-10 6E FD BF
AD 28 3F 25-AE A7 E7 04-0E 8C 4F B4-25 19 7B 65
                                                                                          AF F6 8B D4-AF 38 38 A3-47 F2 95 8D-90 EE 28 16
E5 E0 82 F3-7B 70 28 03-0F 4D 83 34-D6 6C 3F F9
                                                                                          AC 12 AB BC-21 A9 79 D0-FD 2B 5B A3-BB 69 84 66
60 D9 5A B2-1A 4D 36 88-C1 1C DA 86-AC C8 4C 4E
E2 08 99 D3-CD A2 29 EB-F4 58 DF FB-9B DD C4 B6
5B D8 9E 19-0F A5 5F 21-C6 4F 0A B4-CC 57
```







blackhat DALT Encrypted XML Files

DAL configuration is included in encrypted XML files. Encryption is performed using PBKDF2 and AES. The key and salt are hardcoded in DAL.



William Wordsworth

```
using System.IO;
namespace Intel.DAL.Common.TargetDefinitions
 public static class TargetDefDecryptor
  internal const string Salt = "I wandered lonely as a cloud,\r\n
                                                                   That floats on high o'er vales and
hills,\r\n
               When all at once I saw a crowd,\r\n
                                                        A host of golden daffodils";
  public static string Decrypt(string fileName)
   return Intel.DAL.Common.Decryption.Decryptor.DecryptText(File.ReadAllBytes(fileName),
Intel.DAL.Common.IState.Lib.Decryptor.Password, Salt);
  public static Stream DecryptStreamed(string fileName)
  return Intel.DAL.Common.Decryption.Decryptor.DecryptStream((Stream) File.OpenRead(fileName),
Intel.DAL.Common.IState.Lib.Decryptor.Password, Salt);
Intel.DAL.Common.IState.Lib.Decryptor.Password = "ITP"
```

blackhat Our collection (DAL)

DAL Version	TDef size after decryption (B)	System Studio Version	Platforms		
1.9.3906	17,883,723	2014.0.026	AMT_MODULE, AVN, CLN, HSW, LMT, SLM_MODULE, SNB, VLV, VLV2		
1.9.4772	7,375,914	2014.2.035	+TNG		
1.9.5204	8,240,541	2015.0.027	+BDW, +IVB		
1.9.5642	8,749,574	2015.1.039	+CHT		
1.9.6202	49,999,719	2015.2.044, 2015.3.047, 2015.4.051	+ANN, CCK, HSX,IVT,JCK, JKT -CHT		
1.9.7076	75,253,885	2016.0.024	+SKL, SPT		
1.9.7302	75,254,918	2016.0.012	-		
1.9.7438	75,273,050	2016.0.028, 2016.1.028			
1.9.7622	76,037,659	2016.0.045	+BDX, BDXDE		
1.9.7806	76,044,679	2016.2.040	-		
1.9.7870	76,044,472	2016.0.050, 2016.3.043	<u>-</u>		
1.9.8116	76,053,788	2016.0.053, 2016.0.057, 2016.4.046	-		
1.9.8490	80,276,568	2017.0.039, 2017.1.045	+BDXDE_NS		
1.9.g_8968	81,445,662	2017.2.050	+KBL, KBP		
1.9.g_9114	81,026,115	2017.3.057	-		
1.9.g_9490	82,952,952	2018b.0.0.028	+LBG, SKX		
1.9.g_9588	86,433,780	2018.0.040, 2018.1.051, 2018.2.059	+CFL,CHT		
1.1819.222.110	259,507,590	2019b.0.014	+CNP,+MCIVR		
1.1831.340.110	297,853,487	2019.0.033	+CNL, WHL		
1.1839.428.110	309,947,000	2019.1.050, 2019.2.057	+CIV		



Interesting Finds in XML

blackhat CPU FUSEs: With Sauce or Not?

\dal standalone v1.9.3906\DAL\TDef\Devices\CLN\CLN CLTAP\STAP FUS\Fuse.xml

```
<FuseDescriptor Bits="446:446" Name="fb dun ooo dis fuse" description="Out of order Disabled" />
<FuseDescriptor Bits="679:673" Name="fus thm icaldatslope" description="required for calibration value store - calibration fuses used for temperature slope" />
<FuseDescriptor Bits="698:680" Name="fus dpll spare fuses" description="dpll spare fuse bits" />
<FuseDescriptor Bits="699:699" Name="fus_bkend_secretsource" description="Backend Secret Sauce bit: 0 : disable Secret Sauce 1 : enable Secret Sauce" />
<FuseDescriptor Bits="700:700" Name="fus_dfx_scan_disable" description="0 - enable DFX MBIST and scan mode 1 - disable DFX MBIST and scan mode" />
```

What is the "secret sauce bit" or "secret source bit"???





blackhat Deterministic Random Generator

\dal standalone v1.9.3906\DAL\TDef\Devices\HSW\HSW UC\Fuse.A0.xml

<FuseDescriptor Bits="208:208" Name="RNG_DV_EN" __description="RNG_Design Verification(DV) Enable. The usage model for this fuse is to blow it for units sent to testing</p> organizations such as FIPS. Special testing specific microcode patches will be written and provided to FIPS so that they can dump out the output of "/> < Fuse Descriptor Bits="209:209" Name="RNG DISABLE" description="RNG Disable: Blowing this fuse will disable RNG functionality. Microcode will read the local copy of the fuse to enumerate the feature in CPUID (please see ucode in section 5.2, CPUID on page 20). When SW asks for a RDRAND when the feature is not avail "/>

<FuseDescriptor Bits="669:669" Name="RNG DETERMINISTIC" description="RNG Deterministic Mode - The usage model for this fuse is to blow it for post-silicon SV testing for Intel</p> internal use only. This would be provided to validation groups that need to test IVB units with the same microcode as production units or groups th" /> <FuseDescriptor Bits="670:670" Name="RNG FIPS MODE" description="FIPS mode. It is assumed that this will be blown for production parts. It is defined to mean that intervention</p> to the functionality of the DRNG from external sources is disabled. For example, if it is blown, the test port will reset internal DRNG state" />

\dal standalone v1.9.3906\DAL\TDef\Devices\HSW\HSW UC\Fuse.X0.xml

<FuseDescriptor Bits="464:464" Name="RNG_DV_EN" description="RNG_Design Verification(DV) Enable. The usage model for this fuse is to blow it for units sent to testing</p> organizations such as FIPS. Special testing specific microcode patches will be written and provided to FIPS so that they can dump out the output of "/> <FuseDescriptor Bits="465:465" Name="RNG DISABLE" description="RNG Disable: Blowing this fuse will disable RNG functionality. Microcode will read the local copy of the fuse to</p> enumerate the feature in CPUID (please see ucode in section 5.2, CPUID on page 20). When SW asks for a RDRAND when the feature is not avail "/>

<FuseDescriptor Bits="925:925" Name="RNG DETERMINISTIC" description="RNG Deterministic Mode - The usage model for this fuse is to blow it for post-silicon SV testing for Intel</p> internal use only. This would be provided to validation groups that need to test IVB units with the same microcode as production units or groups th" /> <FuseDescriptor Bits="926:926" Name="RNG FIPS MODE" description="FIPS mode. It is assumed that this will be blown for production parts. It is defined to mean that intervention</pre> to the functionality of the DRNG from external sources is disabled. For example, if it is blown, the test port will reset internal DRNG state" />

Deterministic Mode fuse for Hardware Random Generator



What about security?

How can you unlock arbitrary your platform?



blackhat Hardware Security Levels

Intel provides the following access levels: GREEN, ORANGE, RED, and WHITE(?)

Level	JTAG CPU-core (SoC, CPU)	JTAG ME-core (SoC, PCH)	JTAG ISH-core (SoC, PCH)	SB (Full) (SoC, PCH)	JTAG UNCORE (SoC, CPU)	VISA (SoC, PCH,CPU)	Microcode (SoC, CPU)	Purpose
RED	+	+	+	+	+	Full	+	Intel
WHITE(?)	-	-	-	-	-	Full	-	???
ORANGE	+	-/*	+	-	-	Partial	-	Vendors
GREEN	+	-	-	-	-	-/Partial	-	Customer



\dal standalone v1.9.3906\DAL\TDef\Devices\CLN\CLN_CLTAP\STAP_FUS\Fuse.bin.xml

```
<FuseDescriptor Bits="664:664" Name="fuse visa customer disable" description="0 – All VISA signals will be propogated through the VISA tree 1 – Only</pre>
                                                                                   customer visible signals will be propogated through the VISA tree" />
<FuseDescriptor Bits="665:665" Name="fuse visa disable all" description="0 – VISA Signals denoted by the VISA CUSTOMER DISABLE fuse are</p>
                                            propogated through the VISA tree 1 – All VISA signals are blocked from propogating through the VISA tree" />
<FuseDescriptor Bits="700:700" Name="fus dfx scan disable" description="0 - enable DFX MBIST and scan mode 1 - disable DFX MBIST and scan mode"/>
<FuseDescriptor Bits="701:701" Name="fus_jtag_lockout" __description="jtag full lockout: 0 - enable JTAG 1 - lockout JTAG" />
<FuseDescriptor Bits="702:702" Name="fus_itag password disable" description="0 - Enable itag password 1 - Disable itag password" />
<FuseDescriptor Bits="706:703" Name="fus jtag id versioncode" description="version identifier" />
<FuseDescriptor Bits="722:707" Name="Reserved: fus jtag id partnumber" description="Part number #5282 [HEX]" />
<FuseDescriptor Bits="725:723" Name="fus dfx fuse dlcs" description="fuse dlcs" />
<FuseDescriptor Bits="727:726" Name="fus dfx fuse probemode dis" description="fuse probemode dis" />
<FuseDescriptor Bits="729:728" Name="fus fuse resetbrk dis" description="fuse reset brk dis" />
<FuseDescriptor Bits="731:730" Name="fus fuse tap2iosfsb dis" description="fuse tap2iosfsb dis" />
<FuseDescriptor Bits="801:732" Name="dfx spare fuse" description="Spare fuse for dfx unit" />
```

VISA disable bit



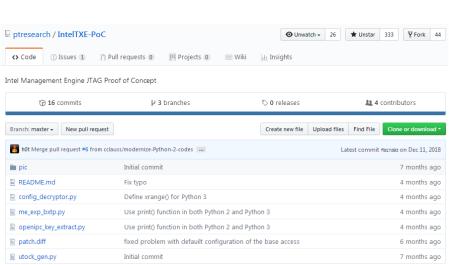
blackhat Unlock Methods

1. Vulnerability



blackhat Unlock: Intel-SA-00086





https://github.com/ptresearch/IntelTXE-PoC





- 1. Activate Manufacture Mode for target.
- 2. Set DCI strap in flash descriptor.
- 3. Using the vulnerability, load value 3 to DFx personality register

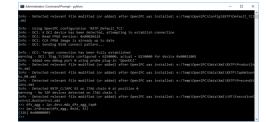
Activation Without Intel's Crypto Keys

4. Enjoy;)









Red Unlock



Memory Tracing



INTEL® CSME, SERVER PLATFORM SERVICES, TRUSTED EXECUTION ENGINE AND INTEL® ACTIVE MANAGEMENT TECHNOLOGY 2018.4 QSR ADVISORY

Intel ID:	INTEL-SA-00185
Advisory Category:	Firmware, Software
Impact of vulnerability:	Escalation of Privilege, Denial of Service, Information Disclosure
Severity rating:	HIGH
Original release:	03/12/2019
Last revised:	03/12/2019

CVEID: CVE-2018-12190

Description: Insufficient input validation in Intel® CSME subsystem before versions 11.8.60, 11.11.60, 11.22.60 or 12.0.20 or Intel® TXE before 3.1.60 or 4.0.10 may allow privileged user to potentially execute arbitrary code via local access.

CVSS Base Score: 8.2 High

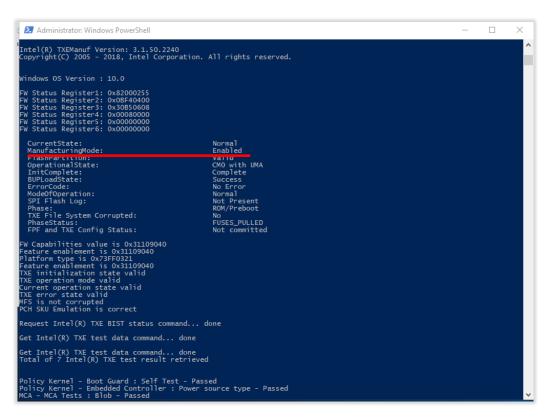
CVSS Vector: CVSS:3.0/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H



blackhat Unlock Methods

2. Orange Mystery (TXE only)

USE: ASIA 2019 the state of the s





Administrator: Windows PowerShell

ypto HW Support eplay Protection Not Supported eplay Protection Bind Counter orage Device Type eplay Protection Bind Status Pre-bind eplay Protection Rebind Not Supported eplay Protection Max Rebind Intel(R) PTT Supported Intel(R) PTT initial power-up state Enabled. AVP Supported integrated Sensor Hub Initial Power State nd of Manufacturing Enable ost Manufacturing NVAR Config Enabled rotect BIOS Environment U Debugging Enabled ISP Initialization Enabled leasured Boot Disabled EM Public Key Hash FPF OEM Public Key Hash TXE FW oot Guard Profile 0 - Legacy ley Manifest ID Not set 0x1 Enabled Not set JFS Boot Source Not set Disabled Disabled MC Boot Source Disabled Disabled Not set SPI Boot Source Enabled | ED Indication Enabled. Disabled Disabled Enabled. Not set Enabled. Not set 0x0 OEM Platform ID Not set SOC Config Lock Not set Not set Not set RPMB Bind Counter 0x0 PMB Migration Done Not set Not set Not set ersistent PRTC Backup Power Exists Exists Exists low OEM Signing of DAL Applets Revoke State Not Revoked Not Revoked PMIC2 to I2C Function Interrupt Community

Unfused OEM Public key

Manufacturing Mode

16 a ade en leg

blackhat Uniock Token

Manufacturing Mode allows setting up one's own public key



Self-signed partition gives Orange Unlock



Self-signed partition gives Orange Unlock for VISA



Orange Unlock gives JTAG access to TXE(ME)-core

Intel said that this is not a bug but a feature!

```
typedef struct {
          MANIFEST_HEADER manifest_header;
          CRYPTO_BLOCK crypto_block;
          SECURE_TOKEN_EXT secure_token;
} SECURE_TOKEN_MANIFEST;
```

```
typedef struct {
    int type;
    int length;
    int ext_version;
    int payload_version;
    int number_ids;
    int token_id;
    int flags;
    int expiration_seconds;
    int manuf_lot;
    char reserved[16];
    SECURE_TOKEN_PARTID part_id_entries[2];
} SECURE_TOKEN_EXT;
```

```
typedef struct {
      char public_key[256];
      int exponent;
      char signature[256];
} CRYPTO_BLOCK;
```



blackhat Unlock Methods

3. Intel JTAG Password

blackhat Unlock: Traffic Light Passwords

\DAL 1.1839.428.110\TDef\Devices\SKL\SKL M UC\TapCmdReg.Tap.SA.M0.xml

```
<_tdef deviceType="SKL M UC" tables="TapDefs">
< tdefDevice steppings="M0">
  <TapDefs>
   <TapCommands>
             <TapCommand Ir="0x042" TapRegister="SA TAP LR GLOBALUNLOCK" name="SA TAP LR GLOBALUNLOCK" Register="READWRITE" />
             <TapCommand Ir="0x048" TapRegister="SA TAP LR REDUNLOCK" name="SA TAP LR REDUNLOCK" Register="READWRITE" PreScan="PackageAndC10Wakeup.Awake" />
             <TapCommand Ir="0x049" TapRegister="SA TAP LR ORANGEUNLOCK" name="SA TAP LR ORANGEUNLOCK" Register="READWRITE" PreScan="PackageAndC10Wakeup.Awake" />
             <TapCommand Ir="0x04A" TapRegister="SA TAP LR UNIQUEID" name="SA TAP LR UNIQUEID" Register="READ1" PreScan="PackageAndC10Wakeup.Awake" />
             <TapRegister indices="71:0" map="" name="SA TAP LR REDUNLOCK">
                          < tag key="Visible" value="false" />
                          <Field _map="71:0" _name="DR_PASSWORD" />
             </TapRegister>
```

\TCI\2013.0.013\itag.ini

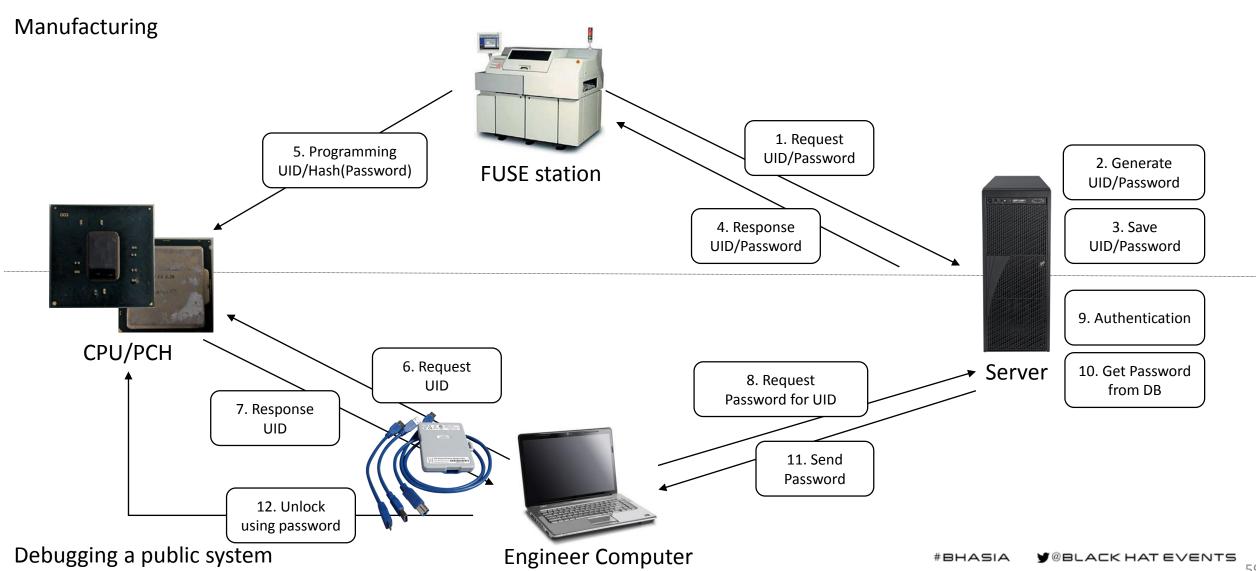
```
[CE4200]
xdp3jtagtclk=10000000
mcrgitagtclk=4000000
cpu#0=1,11,tci-jtag-gen-ia.dll,CE4200,HT0,Master
cpu#1=2,11,tci-jtag-gen-ia.dll,CE4200,HT1,Slave
STUB=jtag.ini
passwd01=13,64,31
passwd02=64,101,77,107,67,111,76,110,85;65 4D 6B 43 6F 4C 6E 55
passwd03=13,2,0
passwd04=16,2,2; make this 7,3 to enable the 8051 on secondary chain
```

\dal_standalone_v1.9.3906\DAL\TDef\Devices\CLN\CLN_CLTAP\STAP_FUS\Fuse.bin.xml

\dal_standalone_v1.9.3906\DAL\TDef\Devices\HSW\HSW_UC\Fuse.A0.xml

```
<FuseDescriptor Bits="12081:12072" Name="SAPF COL RED" description="Display PF SRAM Column Redundancy" />
<FuseDescriptor Bits="12087:12082" Name="security spare" description="spare but Security locked" />
<FuseDescriptor Bits="12088:12088" Name="LR FUSE REG KEY ENABLE FUSE Orange En" description="" />
<FuseDescriptor Bits="12089:12089" Name="LR FUSE REG KEY ENABLE FUSE Red En" description="" />
<FuseDescriptor Bits="12090:12090" Name="LR FUSE REG KEY ENABLE FUSE Legacy Disable" description="" />
<FuseDescriptor Bits="12091:12091" Name="LR FUSE REG KEY ENABLE FUSE Tap Spare" description="" />
<FuseDescriptor Bits="12092:12092" Name="LOCKOUT FUSE HDCP key" description="Disable Programming of some fuses. This fuse is NOT override-able</p>
to prevent fuse programming after the lockout fuse is burnt." />
<FuseDescriptor Bits="12093:12093" Name="LOCKOUT FUSE PAVP2 key" description="Disable Programming of some fuses. This fuse is NOT override-</pre>
able to prevent fuse programming after the lockout fuse is burnt." />
<FuseDescriptor Bits="12094:12094" Name="LOCKOUT FUSE GLOB" description="Disable Programming of some fuses. This fuse is NOT override-able to</p>
prevent fuse programming after the lockout fuse is burnt." />
<FuseDescriptor Bits="12095:12095" Name="LOCKOUT FUSE spare" description="" />
<FuseDescriptor Bits="12159:12096" Name="LR FUSE REG UNIQUEID FUSE UNIQUEID" description="test" />
<FuseDescriptor Bits="12223:12160" Name="LR FUSE REG RED KEY FUSE PASSWORD" description="test" />
<FuseDescriptor Bits="12287:12224" Name="LR FUSE REG ORANGE KEY FUSE PASSWORD" description="test" />
```

blackhat JTAC Password Live Cycle





DAL Start-Up

Starting the DAL

Follow these steps to bring up the DAL software

- · Open the Python console
 - Programs > Intel > Intel DFx > Python Console
- Wait for a command prompt
- Type "itp.devicelist" to confirm communication
- Confirm that the uncore and cores are present
- Type "itp.unlock()" to unlock the CPU
- Enter your user name and password "AMR\username"
- Wait for a command prompt

Intel Architecture Group



45 INTEL CONFIDENTIA

```
1 int64 fastcall Authorization WriteKey(AuthorizationImpl *a1, const char *a2)
2 {
    AuthorizationImpl *v2; // rax
     int64 result: // rax
    if ( a2 )
      v2 = AuthorizationImpl::GetInstance(a1);
      result = AuthorizationImpl::WriteKey(v2, a1, a2);
11
    else
12
      OpenIPC_ErrorPostError(
        0x20000LL,
15
        "../../OpenIPC/Source/Components/Authorization/AuthorizationPublic.cpp",
      result = 0x20000LL:
    return result;
21 }
```

libAuthorization x64.so

https://web.archive.org/web/20190213084218/http://www.keenlit.com/wp-content/uploads/2018/03/IFDIM-BKM-1.pdf



We didn't manage to get the JTAG password for our platform, but we think it is possible this way.





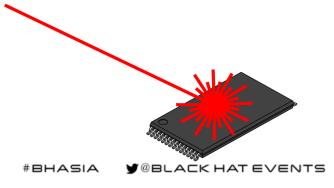
blackhat Unlock Methods ASIA 2019

4. Hardware Way



ME ROM

```
; CODE XREF: RomInit+21B j
ROM: 00020F2D
                                      loc 20F2D:
ROM: 00020F2D B8 50 10 0B F0
                                                               eax, offset gen mmio1 50
                                                       MOV
                                                               dword ptr [eax] 1
ROM: 00020F32 F7 00 01 00 00 00
                                                       test
                                                                                             It's needed to skip the instruction for red unlock
                                                               10c_1FC75
ROM: 00020F38 OF 84 37 ED FF FF
                                                       jz
ROM: 00020F3E
                                      1oc 20F3E:
                                                                                ; CODE XREF: RomInit+2461j
ROM: 00020F3E
ROM: 00020F3E 8B 18
                                                               ebx, [eax]
                                                       MOV
ROM: 00020F40 F7 C3 08 00 00 00
                                                       test
                                                               ebx, 8
                                                               short loc 20F3E
ROM: 00020F46 74 F6
                                                       jz
                                                               ebx, OFFFFFFFEh
ROM: 00020F48 83 E3 FE
                                                       and
ROM: 00020F4B 89 18
                                                               [eax], ebx
                                                       MOV
ROM: 00020F4D E9 AE EC FF FF
                                                               loc_1FC00
                                                       jmp
                                      RomInit
ROM: 00020F4D
                                                       endp
```





- Intel VISA is an **amazing technology for internal architecture research**: speculative execution, out-of-order, IOSF, SAI, etc.
- Intel VISA gives access to internal signals and allows capturing them at a high clock frequency.
- The DAL configuration file has a lot of information about the internal CPU/PCH structure.
- The public device has the special DFx /VISA disable bit, but on our platforms, it isn't used.
- Public CPU/PCH has a built-in UniqueID/Password pair for unlocking market devices.
- The JTAG password isn't a backdoor, it is for debugging the market platform!

blackhat Our Relative Papers

- [1] Where there's a JTAG, there's a Way: Obtaining Full System Access via USB;
- [2] How to Hack a Turned-Off Computer, or Running Unsigned Code in Intel Management Engine;
- [3] Inside Intel Management Engine, 34c3;
- [4] Intel ME: Security keys Genealogy, Obfuscation and other Magic;
- [5] Intel ME Manufacturing Mode: obscured dangers and their relationship to Apple MacBook vulnerability CVE-2018-4251;

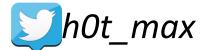


Thank you! Questions?

Mark Ermolov



Maxim Goryachy







POSITIVE TECHNOLOGIES



Bonus Slides



SB Port IDs for Louisburg (LBG) Chipset

BROADCAST1 = 0xFF,			
BROADCAST2 = 0xFE,			
DMI = 0xEF,			
,			
ESPISPI = 0xEE,			
ICLK = 0xED,			
MODPHY4 = 0xEB,			
MODPHY5 = 0x10,			
MODPHY1 = 0xE9,			
PMC = 0xE8,			
XHCI = 0xE6,			
OTG = 0xE5,			
SPE = 0xE4,			
SPD = 0xE3,			
SPC = 0xE2,			
SPB = 0xE1,			
SPA = 0xE0,			
UPSX8 = $0x06$,			
UPSX16 = $0x07$,			
TAP2IOSFSB1 = 0xDF,			
TRSB = 0xDD,			
ICC = 0xDC,			
GBE = 0xDB,			
SATA = 0xD9,			
SSATA = 0x0F,			
= = /			

LDO = 0x14,

```
LDO = 0x14,
DSP = 0xD7,
FUSE = 0xD5,
FSPROX0 = 0xD4,
DRNG = 0xD2,
FIA = 0xCF
FIAWM26 = 0x13,
USB2 = 0xCA,
LPC = 0xC7,
SMB = 0xC6,
P2S = 0xC5,
ITSS = 0xC4,
RTC = 0xC3,
PSF5 = 0x8F
PSF6 = 0x70,
PSF7 = 0x01,
PSF8 = 0x29,
PSF9 = 0x21,
PSF10 = 0x36,
PSF4 = 0xBD,
PSF3 = 0xBC
PSF2 = 0xBB,
PSF1 = 0xBA,
HOTHARM = 0xB9,
DCI = 0xB8,
DFXAGG = 0xB7,
```

```
NPK = 0xB6,
MMP0 = 0xB0,
GPIOCOM0 = 0xAF,
GPIOCOM1 = 0xAE,
GPIOCOM2 = 0xAD,
GPIOCOM3 = 0xAC,
GPIOCOM4 = 0xAB,
GPIOCOM5 = 0x11,
MODPHY2 = 0xA9,
MODPHY3 = 0xA8,
PNCRC = 0xA5.
PNCRB = 0xA4,
PNCRA = 0xA3,
PNCR0 = 0xA2,
CSME15 = 0x9F, //SMS2
CSME14 = 0x9E, //SMS1
CSME13 = 0x9D, //PMT
CSME12 = 0x9C, //PTIO
CSME11 = 0x9B, //PECI
CSME9 = 0x99, //SMT6
CSME8 = 0x98, //SMT5
CSME7 = 0x97, //SMT4
CSME6 = 0x96, //SMT3
CSME5 = 0x95, //SMT2
CSME4 = 0x94, //SMT1
```

```
CSME3 = 0x93, //FSC
CSME2 = 0x92, //USB-RSAI
CSME0 = 0x90, //CSE
CSME PSF = 0x8F, //MEPSF
CSMERTC = 0x8E
IEUART = 0x80,
IEHOTHAM = 0x7F,
IEPMT = 0x7E,
IESSTPECI = 0x7D,
IEFSC = 0x7C,
IESMT5 = 0x7B,
IESMT4 = 0x7A,
IESMT3 = 0x79,
IESMT2 = 0x78.
IESMT1 = 0x77,
IESMT0 = 0x76,
IEUSBR = 0x74,
IEPTIO = 0x73,
IEIOSFGASKET = 0x72,
IEPSF = 0x70,
FPK = 0x0A,
MPOKR = 0x3C,
MP1KR = 0x3E,
RUAUX = 0x0B,
```

RUMAIN = 0x3B, EC = 0x20, CPM2 = 0x38, CPM1 = 0x37, CPM0 = 0x0C, VSPTHERM = 0x25, VSPP2SB = 0x24, VSPFPK = 0x22, VSPCPM2 = 0x35, VSPCPM1 = 0x34, VSPCPM0 = 0x33, MSMROM = 0x08, PSTH = 0x89

blackhat Override-able Fuses

\dal standalone v1.9.3906\DAL\TDef\Devices\CLN\CLN CLTAP\STAP FUS\Fuse.bin.xml

```
<FuseDescriptor Bits="12071:12056" Name="security spare" description="spare but Security locked" />
<FuseDescriptor Bits="12075:12072" Name="LCP DEFAULTS core" description="default value of core LCP" />
<FuseDescriptor Bits="12079:12076" Name="LCP DEFAULTS IIc" description="default value of IIc LCP" />
<FuseDescriptor Bits="12083:12080" Name="LCP DEFAULTS ring" description="default value of ring LCP" />
<FuseDescriptor Bits="12087:12084" Name="LCP DEFAULTS sa" description="default value of sa LCP" />
<FuseDescriptor Bits="12088:12088" Name="LR FUSE REG KEY ENABLE FUSE Orange En" description="" />
<FuseDescriptor Bits="12089:12089" Name="LR FUSE REG KEY ENABLE FUSE Red En" description="" />
<FuseDescriptor Bits="12090:12090" Name="LR FUSE REG KEY ENABLE FUSE Legacy Disable" description="" />
<FuseDescriptor Bits="12091:12091" Name="LR FUSE REG KEY ENABLE FUSE Tap Spare" description="" />
<FuseDescriptor Bits="12092:12092" Name="LOCKOUT FUSE HDCP key" description="Disable Programming of some fuses. This fuse is NOT override-able</pre>
to prevent fuse programming after the lockout fuse is burnt." />
<FuseDescriptor Bits="12093:12093" Name="LOCKOUT_FUSE_PAVP2_key" description="Disable Programming of some fuses. This fuse is NOT override-</p>
able to prevent fuse programming after the lockout fuse is burnt." />
<FuseDescriptor Bits="12094:12094" Name="LOCKOUT FUSE GLOB" description="Disable Programming of some fuses. This fuse is NOT override-able to</p>
prevent fuse programming after the lockout fuse is burnt." />
<FuseDescriptor Bits="12095:12095" Name="LOCKOUT FUSE spare" description="" />
<FuseDescriptor Bits="12159:12096" Name="LR FUSE REG UNIQUEID FUSE UNIQUEID" description="test" />
<FuseDescriptor Bits="12223:12160" Name="LR FUSE REG RED KEY FUSE PASSWORD" description="test" />
<FuseDescriptor Bits="12287:12224" Name="LR FUSE REG ORANGE KEY FUSE PASSWORD" description="test" />
```

Does it mean that other fuses are override-able???