SAMSUNG

Debugpci: Making PCIe Common Error Debugging Easier

Shradha Todi and Padmanabhan Rajanbabu

Samsung Semiconductor India R&D Center

Agenda

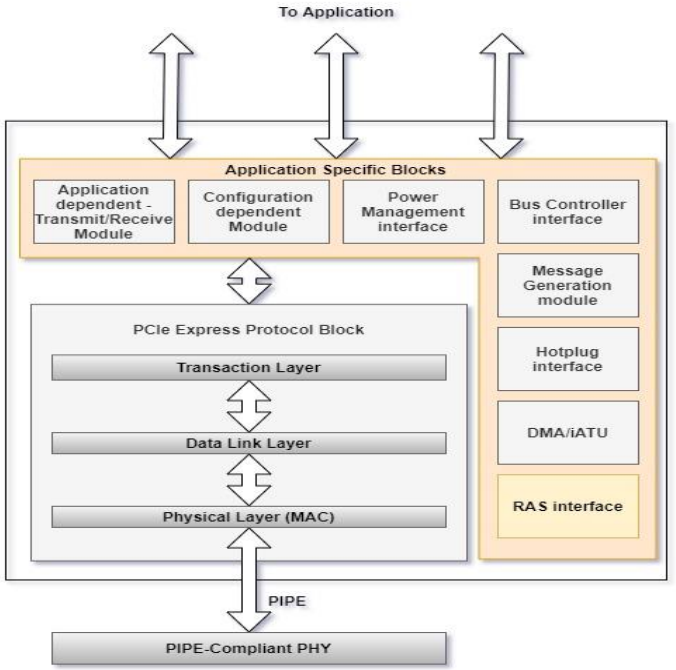
- Introduction
- Debug features in DesignWare PCIe controller
- Need for a streamlined PCIe debugging procedure
- pciutils – Open-Source Tool for diagnosis
- debugpci – command line based diagnostic tool
- Architecture and software stack of debugpci
- Software implementation of debugpci
- Real Use-Cases
- Conclusion and Future scope

Introduction

- PCIe - most prevalent interface standard for connecting high-speed components
- Critical applications utilizing PCIe shall be bulletproof and validated thoroughly for all conditions
- Debugging PCIe IP is cumbersome in the absence of IP Internal debug registers for status check
- DesignWare PCIe IP has a versatile debug and RAS-DES register set to ease debugging effort
- This presentation proposes a simple command line based diagnostic tool compatible with any Linux system
- This tool utilizes debug registers to simplify PCIe error solving methodology
- The tool also follow a streamlined debugging process, improving the debugging efficiency

Debug Features in DesignWare PCIe Controller

- Vital debug hooks provided by the controller
  - LTSSM state and equalization status
  - Error Injection - ECRC/LCRC/Unsupported Request
  - Error detection, error logging and error handling mechanisms
  - Time and event-based counters to measure the % of time the controller spends in each low power state
  - TX/RX data throughput in the system
  - Statistical event counters



Existing PCIe Debugging Procedure

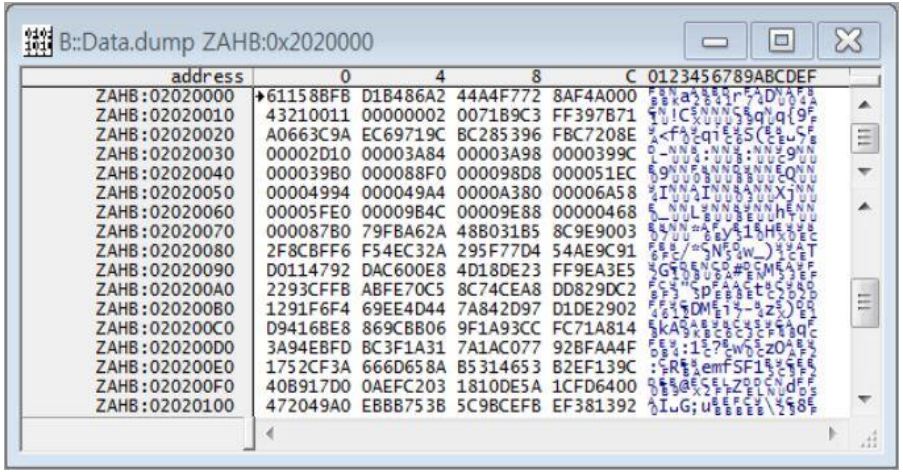
- Mainstream debugging procedure involves capturing trace using PCIe analyzers

Source: https://www.storagenewsletter.com/wp-content/uploads/2018/06/TELEDYNE_LECROY_1806_summit_m5x_2.jpg

But why does it need an alternative?

Need for Streamlined PCIe Debugging Procedure

- Need to dump software debug registers to obtain information
- Decoding the dump in the absence of any user space tool is tedious



Source: http://trace32.com/wiki/index.php/Accessing_System_Data

pciutils – Open Source Tool for Diagnosis

- Collection of utilities to access PCI bus configuration registers
- Runs on Linux, Windows and many more platforms
- Easy to add support for platform/board specific PCIe configuration registers and commands
- The library has the following utilities:
  - lspci: displays information about all PCI buses and devices.
  - setpci: allows to read from and write to PCI device configuration registers
  - update-pciids: download the current version of the pci.ids file.
- Source - https://git.kernel.org/pub/scm/utils/pciutils/pciutils.git

lspci output

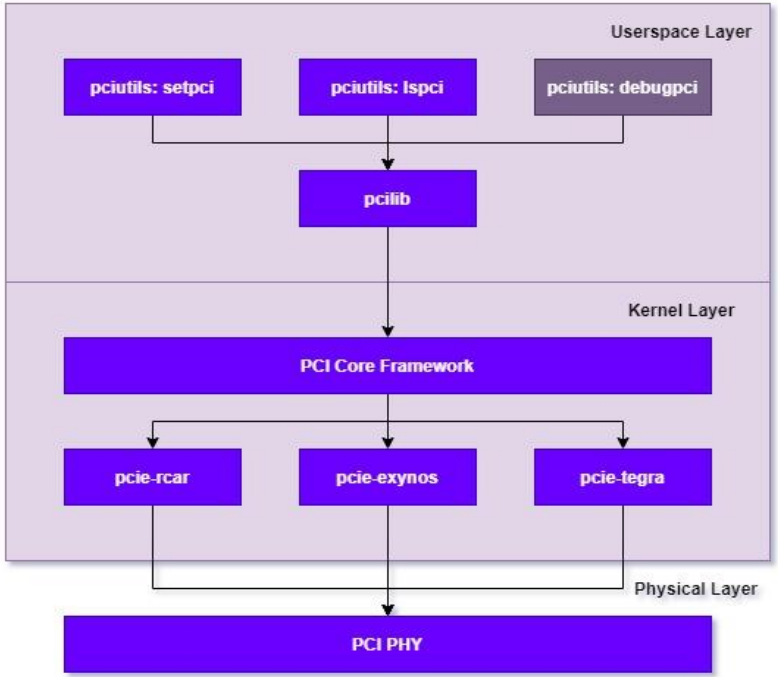

```
# lspci -s 0000:01:00.0 -vvv
0000:01:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd Device a804 (prog-if 02 [NVM Express])
        Subsystem: Samsung Electronics Co Ltd Device a801
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Interrupt: pin A routed to IRQ 193
        NUMA node: 0
        Region 0: Memory at 15a00000 (64-bit, non-prefetchable) [size=16K]
        Capabilities: [40] Power Management version 3
                Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0-,D1-,D2-,D3hot-,D3cold-)
                Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [50] MSI: Enable+ Count=8/32 Maskable- 64bit+
                Address: 00000002760f0000  Data: 0008
        Capabilities: [70] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 256 bytes, PhantFunc 0, Latency L0s unlimited, L1 unlimited
                        ExtTag- AttnBtn- AttnInd- PwrInd- RBE+ FLReset+ SlotPowerLimit 0.000W
                DevCtl: Report errors: Correctable+ Non-Fatal+ Fatal+ Unsupported+
                        RlxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop+ FLReset-
                        MaxPayload 128 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr- UncorrErr- FatalErr- UnsuppReq- AuxPwr+ TransPend-
                LnkCap: Port #0, Speed 8GT/s, Width x4, ASPM L1, Exit Latency L0s unlimited, L1 <64us
                        ClockPM+ Surprise- LLActRep- BwNot- ASPMOptComp+
                LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk+
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 8GT/s, Width x4, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
                DevCap2: Completion Timeout: Range ABCD, TimeoutDis+, LTR+, OBFF Not Supported
                DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disabled
                LnkCtl2: Target Link Speed: 8GT/s, EnterCompliance- SpeedDis-
                         Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
                         Compliance De-emphasis: -6dB
                LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete+, EqualizationPhase1+
                         EqualizationPhase2-, EqualizationPhase3-, LinkEqualizationRequest-
        Capabilities: [b0] MSI-X: Enable- Count=8 Masked-
                Vector table: BAR=0 offset=00003000
                PBA: BAR=0 offset=00002000
        Capabilities: [100 v2] Advanced Error Reporting
                UESta:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
                UEMsk:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
                UESvrt: DLP+ SDES+ TLP- FCP+ CmpltTO- CmpltAbrt- UnxCmplt- RxOF+ MalfTLP+ ECRC- UnsupReq- ACSViol-
                CESta:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr-
                CEMsk:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
                AERCap: First Error Pointer: 00, GenCap+ CGenEn- ChkCap+ ChkEn-
        Capabilities: [148 v1] Device Serial Number 00-00-00-00-00-00-00-00
        Capabilities: [158 v1] Power Budgeting <?>
        Capabilities: [168 v1] #19
        Capabilities: [188 v1] Latency Tolerance Reporting
                Max snoop latency: 0ns
                Max no snoop latency: 0ns
        Capabilities: [190 v1] L1 PM Substates
                L1SubCap: PCI-PM_L1.2+ PCI-PM_L1.1+ ASPM_L1.2+ ASPM_L1.1+ L1_PM_Substates+
                          PortCommonModeRestoreTime=10us PortTPowerOnTime=10us
        Kernel driver in use: nvme
```

debugpci – Command Line Based Diagnostic Tool

- *debugpci* - command line based diagnostic tool to help users dump all the required debug data
- The commands that will be used for capturing and dumping data:
  - Begin Capture - `debugpci [[[[<domain>]:]<bus>]:][<slot>][.[<func>]] c`
  - Ananlyse capture - `debugpci [[[[<domain>]:]<bus>]:][<slot>][.[<func>]] d`
- Internally captures counters for different errors, LTSSM states and silicon debug registers provided by DesignWare PCIe IP
- Presents the captured debug data in a human readable format
- Analyses the discrepancy (If any) seen in the collected data and presents the same to user with all possible root causes
- For example, if the tool checks that there was a receiver detection related timeout when reading SD_CONTROL2_REG. It will alert the user and provide input by printing following message:
  - "A Receiver detection timeout was seen. If the PHY requires more time for receiver detection, the application software can hold LTSSM in Detect.Active by setting the HOLD_LTSSM field of SD_CONTROL2_REG[0] register."

Architecture And Software Stack of debugpci

- The PCI library is a portable library for accessing PCI devices and their configuration space present in the user space layer.
- Linux PCI subsystem enumerates and populates the PCI devices. The vendors have underlying low level device drivers in the kernel layer.
- PCI PHY is the physical layer in the PCI architecture stack



Software Implementation : Main Function

- The main function gets the pci_access structure, initializes the PCI library and gets the device that is requested by user.
- Depending on whether the user has selected option “c” or “d”, it calls the capture or dump function respectively.

```
int main(int argc, char **argv)
{
        struct pci_access *pacc;
        struct pci_dev *dev;

        if (argc < 3) {
                printf("Usage: debugpci <slot> c - Start to capture debug data\n");
                printf("Usage: debugpci <slot> d - Dump the debug data\n");
                return 0;
        }

        /* Get the pci_access structure */
        pacc = pci_alloc();
        pci_filter_init(pacc, &filter);
        pci_filter_parse_slot(&filter, argv[1]);

        /* Initialize the PCI library */
        pci_init(pacc);
        /* We want to get the list of devices */
        pci_scan_bus(pacc);
        /* Iterate over all devices */
        for (dev=pacc->devices; dev; dev=dev->next)
        {
                if (pci_filter_match(&filter, dev))
                        break;
        }

        /* Present the captured debug data in a human readable format */
        if (strcmp(argv[2], "c") == 0)
                debugpci_capture(dev);
        /*
         * Analyze the captured debug data and present
         * and highlight possible root-cause
         */
        if (strcmp(argv[2], "d") == 0)
                debugpci_dump(dev);

        /* Close everything */
        pci_cleanup(pacc);
        return 0;
}
```

Software Implementation : Capture Function


```
static void debugpci_capture(struct pci_dev *dev)
{
        ras_des_base = find_ras_base(dev);
        if (ras_des_base == 0) {
                printf("Device does not support debug registers\n");
                return;
        }

        printf("Enabling all debug counters...\n");

        for (i = 0; i < sizeof(events) / sizeof(events[0]); i++) {
                val = pci_read_long(dev, ras_des_base + EVENT_OFFSET);
                val &= ~(EVENT_MASK);
                val |= events[i].event_id << EVENT_SHIFT;
                val &= ~(GROUP_MASK);
                val |= events[i].group_id << GROUP_SHIFT;
                val &= ~(EVENT_ENABLE_MASK);
                val |= EVENT_ENABLE << EVENT_ENABLE_SHIFT;
                if (events[i].group_id == 0) {
                        for (j = 0; j < 4; j++) {
                                val &= ~(LANE_SEL_MASK);
                                val |= j;
                                pci_write_long(dev, ras_des_base + EVENT_OFFSET, val);
                        }
                } else {
                        pci_write_long(dev, ras_des_base + EVENT_OFFSET, val);
                }
        }
}
```

```
static struct event_counters events[] = {
        {0x0, 0x00, "EBUF Overflow"},
        {0x0, 0x01, "EBUF Underrun"},
        {0x0, 0x02, "Decode Error"},
        {0x0, 0x03, "Running Disparity Error"},
        {0x0, 0x04, "SKP OS Parity Error"},
        {0x0, 0x05, "SYNC Header Error"},
        {0x0, 0x06, "Rx Valid de-assertion"},
        {0x0, 0x07, "CTL SKP OS Parity Error"},
        {0x0, 0x08, "1st Retimer Parity Error"},
        {0x0, 0x09, "2nd Retimer Parity Error"},
        {0x0, 0x0A, "Margin CRC and Parity Error"},
        {0x1, 0x05, "Detect EI infer"},
        {0x1, 0x06, "Receiver Error"},
        {0x1, 0x07, "Rx Recovery Request"},
        {0x1, 0x08, "N_FT3 Timeout"},
        {0x1, 0x09, "Framing Error"},
        {0x1, 0x0A, "Deskew Error"},
        {0x2, 0x00, "BAD TLP"},
        {0x2, 0x01, "LCRC Error"},
        {0x2, 0x02, "BAD DLLP"},
        {0x2, 0x03, "Replay Number Rollover"},
        {0x2, 0x04, "Replay Timeout"},
        {0x2, 0x05, "Rx Nak DLLP"},
        {0x2, 0x06, "Tx Nak DLLP"},
        {0x2, 0x07, "Retry TLP"},
        {0x3, 0x00, "FC Timeout"},
        {0x3, 0x01, "Poisoned TLP"},
        {0x3, 0x02, "ECRC Error"},
        {0x3, 0x03, "Unsupported Request"},
        {0x3, 0x04, "Completer Abort"},
        {0x3, 0x05, "Completion Timeout"},
        {0x4, 0x00, "EBUF SKP Add"},
        {0x4, 0x01, "EBUF SKP Divide"},

};
```

- The capture function enables all debug and error counters present in DWC controller for all the lanes
- Some error counters added in a static structure is shown in the figure above

Software Implementation : Dump Function


```
static void debugpci_dump(struct pci_dev *dev)
{
        for (i = 0; i < sizeof(events) / sizeof(events[0]); i++) {
                val = pci_read_long(dev, ras_des_base + EVENT_OFFSET);
                val &= ~(EVENT_MASK);
                val |= events[i].event_id << EVENT_SHIFT;
                val &= ~(GROUP_MASK);
                val |= events[i].group_id << GROUP_SHIFT;
                val &= ~(EVENT_ENABLE_MASK);
                val |= 0x0 << EVENT_SHIFT;
                if (events[i].group_id == 0) {
                        for (j = 0; j < 4; j++) {
                                val &= ~(LANE_SEL_MASK);
                                val |= j;
                                pci_write_long(dev, ras_des_base + EVENT_OFFSET, val);
                                printf("%s- Lane %d: %d\n", events[i].name, j,
                                                pci_read_long(dev, ras_des_base + EVENT_DATA_OFFSET));
                        }
                } else {
                        printf("%s: %d\n", events[i].name,
                                        pci_read_long(dev, ras_des_base + EVENT_DATA_OFFSET));
                }
        }
        for (i = 0; i < sizeof(debug) / sizeof(debug[0]); i++) {
                if (debug[i].lane_debug) {
                        for (j = 0; j < 4; j++) {
                                val = pci_read_long(dev, ras_des_base + debug[i].offset);
                                val &= ~(LANE_SELECT_MASK);
                                val |= j << LANE_SELECT_SHIFT;
                                pci_write_long(dev, ras_des_base + debug[i].offset, val);
                                debug[i].val = SHOW_BITS(pci_read_long(dev, ras_des_base + debug[i].offset), debug[i].mask);
                                printf("%s- Lane %d: %d\n", debug[i].name, j, debug[i].val);
                        }
                }
                else {
                        debug[i].val = SHOW_BITS(pci_read_long(dev, ras_des_base + debug[i].offset), debug[i].mask);
                        printf("%s: %d\n", debug[i].name, debug[i].val);
                }
        }
        /* analyze the captured data and do root-cause analysis */
        root_cause_issue(dev);
}
```

- The dump function dumps all debug and error counters present in DWC controller for all the lanes by reading respective error/debug register
- Function root_cause_issue compares the read value with expected value and in case of mismatch, prints the root-cause analysis for various observed issues

Real Use-Cases

- **Receiver Detection -** Receiver detection is used to determine if a remote PCI e link partner is available to establish a linkup
- **Broken Lanes -** After receiver detection, if some lanes are broken, the link m ay not reach L0 at the desired link width
- **Speed Change -** Sometimes link does not come up / unstable at the new spe ed or link falls back to a lower speed after speed change
- **Link Equalization -** Sometimes link fails to come up at Gen3 / Gen4 data rate or is unstable after speed change to Gen3 / Gen4 data rate

Real Use-Cases

- **Receiver Error -** PCIe link does not remain stable in L0 and goes down to Recovery
- **Correctable Errors -** Correctable errors are the most common consequence of poor link quality and can lead to link instability.
- **Uncorrectable Errors -** Uncorrectable errors are further classified as Fatal Errors and Non-Fatal Errors based on the Severity
- **ASPM issues -** Sometimes link is unable to properly enter low power states like L0S, L1 and it's substates

Real Use-Case: Healthy PCIe link


```
EBUF Overflow
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
EBUF Underrun
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
Decode Error
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
Running Disparity Error
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
SKP OS Parity Error
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
SYNC Header Error
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
CTL SKP OS Parity Error
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
1st Retimer Parity Error
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
```

```
Margin CRC and Parity Error
        Lane 0:                 0
        Lane 1:                 0
        Lane 2:                 0
        Lane 3:                 0
Detect EI infer:                0
Receiver Error:                 0
Rx Recovery Request:            0
N_FT3 Timeout:          0
Framing Error:          0
Deskew Error:           0
BAD TLP:                0
LCRC Error:             0
BAD DLLP:               0
Replay Number Rollover:                 0
Replay Timeout:                 0
Rx Nak DLLP:            0
Tx Nak DLLP:            0
Retry TLP:              0
FC Timeout:             0
Poisoned TLP:           0
ECRC Error:             0
Unsupported Request:            0
Completer Abort:                0
Completion Timeout:             0
FC Protocol Error:              0
DL Protocol Error:              0
Malformed TLP:          0
Surprise Down:          0
Internal Error:                 0
RX Valid
        Lane 0:                 1
        Lane 1:                 1
        Lane 2:                 1
        Lane 3:                 1
Lane detected
        Lane 0:                 1
        Lane 1:                 1
        Lane 2:                 1
        Lane 3:                 1
```

- Tool prints all possible debug and error register values for all possible lanes
- Against each error, it prints the number of times the error occurred
- Against debug values like RX valid, or lane detected value 1 indicates true and value 0 indicates false

Real Use-Case: Uncorrectable Errors


```
Detect EI infer:                        0
Receiver Error:                         0
Rx Recovery Request:                    0
N_FT3 Timeout:                  0
Framing Error:                  0
Deskew Error:                   0
BAD TLP:                        0
LCRC Error:                     0
BAD DLLP:                       0
Replay Number Rollover:                         0
Replay Timeout:                         0
Rx Nak DLLP:                    0
Tx Nak DLLP:                    0
Retry TLP:                      0
FC Timeout:                     0
Poisoned TLP:                   0
ECRC Error:                     0
Unsupported Request:                    0
Completer Abort:                        0
Completion Timeout:                     0
FC Protocol Error:                      1
DL Protocol Error:                      0
Malformed TLP:                  0
Surprise Down:                  0
Internal Error:                         0
RX Valid
        Lane 0:                         1
        Lane 1:                         1
        Lane 2:                         1
        Lane 3:                         1
Lane detected
        Lane 0:                         1
        Lane 1:                         1
        Lane 2:                         1
        Lane 3:                         1
```

```
Flow Control Protocol Error: Occurs if no DLLP is received within a 200us window
This indicates that the link quality is severely deteriorated
```

- These are errors that render the link and related hardware unreliable. These are only seen when any of the link components is severely broken, and forces link to go to detect state
- As we can see, Protocol error status is flagged in the tool
- The root cause is given in the last line stating the reason why protocol error can occur

Real Use-Case: Framing Error


```
Receiver Error:                         0
Rx Recovery Request:                    0
N_FT3 Timeout:          0
Framing Error:          1
Deskew Error:           0
BAD TLP:                0
LCRC Error:             0
BAD DLLP:               0
Replay Number Rollover:                         0
Replay Timeout:                         0
Rx Nak DLLP:            0
Tx Nak DLLP:            0
Retry TLP:              0
FC Timeout:             0
Poisoned TLP:           0
ECRC Error:             0
Unsupported Request:                    0
Completer Abort:                        0
Completion Timeout:                     0
FC Protocol Error:                      0
DL Protocol Error:                      0
Malformed TLP:          0
Surprise Down:          0
Internal Error:                         0
RX Valid
        Lane 0:                         1
        Lane 1:                         1
        Lane 2:                         1
        Lane 3:                         1
Lane detected
        Lane 0:                         1
        Lane 1:                         1
        Lane 2:                         1
        Lane 3:                         1
```

- Framing Token
  - When EDS token was expected but not received or whenever an EDS token was received but not expected.
  - When a framing error was detected in the deskew block while a packet has been in progress in token_finder.
- Unexpected STP Token
  - When Framing CRC in STP token did not match
  - When Framing Parity in STP token did not match.
- Unexpected Block
  - When RxStatus Error was detected in Datastream state
  - When Not full 16 EIEOS symbols are received in EIEOS state

```
Framing Error: Indicates poor link quality. Investigate PHY and System level factors affecting link quality
The type of framing error recieved is "When PHY status error was detected in SKPOS state"
For debug purpose, set bit[16] of SD_CONTROL2_REG to disable transition to Recovery due to Framing error.
```

Real Use-Case: Broken Lanes


```
Detect EI infer:                        0
Receiver Error:                         0
Rx Recovery Request:                    0
N_FT3 Timeout:                  0
Framing Error:                  0
Deskew Error:                   0
BAD TLP:                        0
LCRC Error:                     0
BAD DLLP:                       0
Replay Number Rollover:                                 0
Replay Timeout:                         0
Rx Nak DLLP:                    0
Tx Nak DLLP:                    0
Retry TLP:                      0
FC Timeout:                     0
Poisoned TLP:                   0
ECRC Error:                     0
Unsupported Request:                    0
Completer Abort:                        0
Completion Timeout:                     0
FC Protocol Error:                      0
DL Protocol Error:                      0
Malformed TLP:                  0
Surprise Down:                  0
Internal Error:                         0
RX Valid
        Lane 0:                         1
        Lane 1:                         1
        Lane 2:                         0
        Lane 3:                         0
Lane detected
        Lane 0:                         1
        Lane 1:                         1
        Lane 2:                         1
        Lane 3:                         1
```

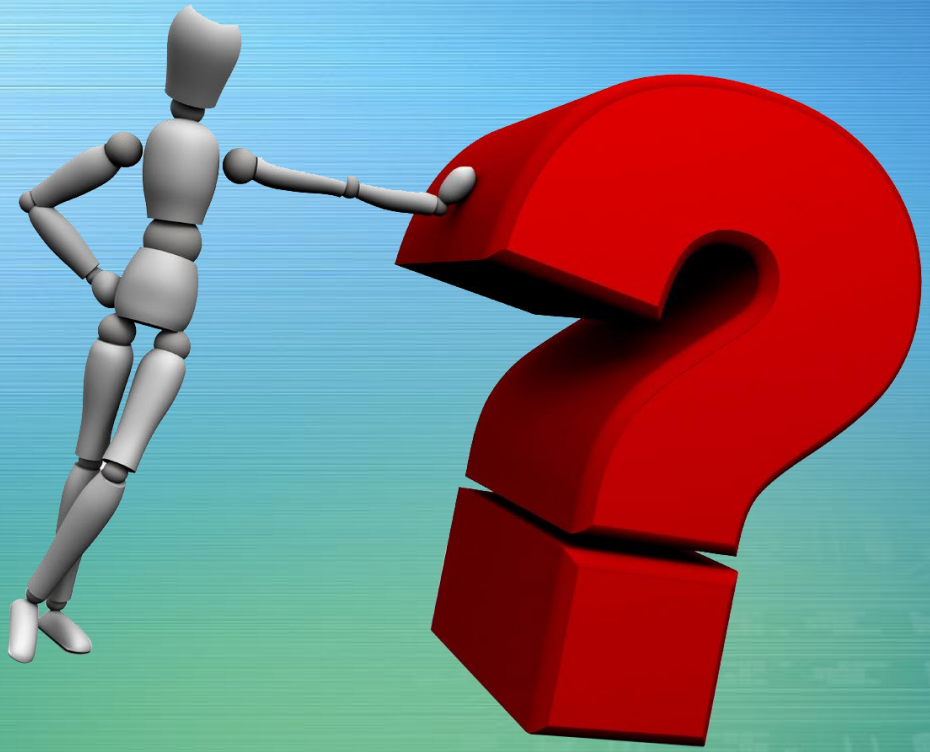
Rx Valid: Since Lane 2 and Lane 3 shows RX not valid, it might have broken lanes

- After receiver detection is completed, the LTSSM goes through following states, Polling -> Configuration -> Recovery before reaching L0 state at Gen1 data rate.
- If some lanes are broken after receiver detection, the link may not reach L0 at the desired link width
- The RX valid not being set indicates there might be broken lanes as mentioned in the last line

Conclusion and Future Scope

- We were able to use this diagnostic tool even in pre-silicon emulation with ZeBu PCIe transactor for PCIe link-partner and root-caused several issues being faced during pre-silicon validation
- This tool helps in avoiding any human error while manipulating the debug data
- The users of DesignWare PCIe IP can simply use this tool for self diagnosis and share the report to Synopsys or customer for further analysis
- For every issue detected, a possible error condition is always highlighted
- Future scope involves upstreaming the source code as part of GPL license in Linux or as open source tool

Any Questions ?



THANK YOU