PCI NTB Function

Author: Kishon Vijay Abraham I <kishon@ti.com>

PCI Non-Transparent Bridges (NTB) allow two host systems to communicate with each other by exposing each host as a device to the other host. NTBs typically support the ability to generate interrupts on the remote machine, expose memory ranges as BARs, and perform DMA. They also support scratchpads, which are areas of memory within the NTB that are accessible from both machines.

PCI NTB Function allows two different systems (or hosts) to communicate with each other by configuring the endpoint instances in such a way that transactions from one system are routed to the other system.

In the below diagram, PCI NTB function configures the SoC with multiple PCI Endpoint (EP) instances in such a way that transactions from one EP controller are routed to the other EP controller. Once PCI NTB function configures the SoC with multiple EP instances, HOST1 and HOST2 can communicate with each other using SoC as a bridge.

```
   +-------------+                                   +-------------+
   |             |                                   |             |
   |    HOST1    |                                   |    HOST2    |
   |             |                                   |             |
   +------^------+                                   +------^------+
          |                                                 |
          |                                                 |
+---------|-------------------------------------------------|---------+
|  +------v------+                                   +------v------+  |
|  |             |                                   |             |  |
|  |     EP      |                                   |     EP      |  |
|  | CONTROLLER1 |                                   | CONTROLLER2 |  |
|  |             <----------------------------------->             |  |
|  |             |                                   |             |  |
|  |             |                                   |             |  |
|  |             |  SoC With Multiple EP Instances   |             |  |
|  |             |  (Configured using NTB Function)  |             |  |
|  +-------------+                                   +-------------+  |
+---------------------------------------------------------------------+
```

Constructs used for Implementing NTB

1. Config Region
2. Self Scratchpad Registers
3. Peer Scratchpad Registers
4. Doorbell (DB) Registers
5. Memory Window (MW)

Config Region:

Config Region is a construct that is specific to NTB implemented using NTB Endpoint Function Driver. The host and endpoint side NTB function driver will exchange information with each other using this region. Config Region has Control/Status Registers for configuring the Endpoint Controller. Host can write into this region for configuring the outbound Address Translation Unit (ATU) and to indicate the link status. Endpoint can indicate the status of commands issued by host in this region. Endpoint can also indicate the scratchpad offset and number of memory windows to the host using this region.

The format of Config Region is given below. All the fields here are 32 bits.

```
      +------------------------+
      |         COMMAND        |
      +------------------------+
      |         ARGUMENT       |
      +------------------------+
      |         STATUS         |
      +------------------------+
      |         TOPOLOGY       |
      +------------------------+
      |    ADDRESS (LOWER 32)  |
      +------------------------+
      |    ADDRESS (UPPER 32)  |
      +------------------------+
      |           SIZE         |
      +------------------------+
      |   NO OF MEMORY WINDOW  |
      +------------------------+
      |  MEMORY WINDOW1 OFFSET |
      +------------------------+
      |       SPAD OFFSET      |
      +------------------------+
      |        SPAD COUNT      |
      +------------------------+
      |      DB ENTRY SIZE     |
      +------------------------+
      |         DB DATA        |
      +------------------------+
      |            :           |
      +------------------------+
      |            :           |
      +------------------------+
      |         DB DATA        |
      +------------------------+


COMMAND:

      NTB function supports three commands:

        CMD_CONFIGURE_DOORBELL (0x1): Command to configure doorbell. Before
      invoking this command, the host should allocate and initialize
      MSI/MSI-X vectors (i.e., initialize the MSI/MSI-X Capability in the
      Endpoint). The endpoint on receiving this command will configure
      the outbound ATU such that transactions to Doorbell BAR will be routed
      to the MSI/MSI-X address programmed by the host. The ARGUMENT
      register should be populated with number of DBs to configure (in the
      lower 16 bits) and if MSI or MSI-X should be configured (BIT 16).

        CMD_CONFIGURE_MW (0x2): Command to configure memory window (MW). The
      host invokes this command after allocating a buffer that can be
      accessed by remote host. The allocated address should be programmed
      in the ADDRESS register (64 bit), the size should be programmed in
      the SIZE register and the memory window index should be programmed
      in the ARGUMENT register. The endpoint on receiving this command
      will configure the outbound ATU such that transactions to MW BAR
      are routed to the address provided by the host.

        CMD_LINK_UP (0x3): Command to indicate an NTB application is
      bound to the EP device on the host side. Once the endpoint
      receives this command from both the hosts, the endpoint will
      raise a LINK_UP event to both the hosts to indicate the host
      NTB applications can start communicating with each other.

ARGUMENT:

      The value of this register is based on the commands issued in
      command register. See COMMAND section for more information.

TOPOLOGY:

      Set to NTB_TOPO_B2B_USD for Primary interface
      Set to NTB_TOPO_B2B_DSD for Secondary interface

ADDRESS/SIZE:

      Address and Size to be used while configuring the memory window.
      See "CMD_CONFIGURE_MW" for more info.

MEMORY WINDOW1 OFFSET:

      Memory Window 1 and Doorbell registers are packed together in the
      same BAR. The initial portion of the region will have doorbell
      registers and the latter portion of the region is for memory window 1.
      This register will specify the offset of the memory window 1.

NO OF MEMORY WINDOW:

      Specifies the number of memory windows supported by the NTB device.

SPAD OFFSET:

      Self scratchpad region and config region are packed together in the
      same BAR. The initial portion of the region will have config region
      and the latter portion of the region is for self scratchpad. This
      register will specify the offset of the self scratchpad registers.

SPAD COUNT:

      Specifies the number of scratchpad registers supported by the NTB
      device.

DB ENTRY SIZE:
      Used to determine the offset within the DB BAR that should be written
      in order to raise doorbell. EPF NTB can use either MSI or MSI-X to
      ring doorbell (MSI-X support will be added later). MSI uses same
      address for all the interrupts and MSI-X can provide different
      addresses for different interrupts. The MSI/MSI-X address is provided
      by the host and the address it gives is based on the MSI/MSI-X
      implementation supported by the host. For instance, ARM platform
      using GIC ITS will have the same MSI-X address for all the interrupts.
      In order to support all the combinations and use the same mechanism
      for both MSI and MSI-X, EPF NTB allocates a separate region in the
      Outbound Address Space for each of the interrupts. This region will
      be mapped to the MSI/MSI-X address provided by the host. If a host
      provides the same address for all the interrupts, all the regions
      will be translated to the same address. If a host provides different
      addresses, the regions will be translated to different addresses. This
      will ensure there is no difference while raising the doorbell.

DB DATA:

      EPF NTB supports 32 interrupts, so there are 32 DB DATA registers.
      This holds the MSI/MSI-X data that has to be written to MSI address
      for raising doorbell interrupt. This will be populated by EPF NTB
      while invoking CMD_CONFIGURE_DOORBELL.
```

Scratchpad Registers:

Each host has its own register space allocated in the memory of NTB endpoint controller. They are both readable and writable from both sides of the bridge. They are used by applications built over NTB and can be used to pass control and status information between both sides of a device.

Scratchpad registers has 2 parts

1. Self Scratchpad: Host's own register space
2. Peer Scratchpad: Remote host's register space.

Doorbell Registers:

Doorbell Registers are used by the hosts to interrupt each other.

Memory Window:

Actual transfer of data between the two hosts will happen using the memory window.

Modeling Constructs:

There are 5 or more distinct regions (config, self scratchpad, peer scratchpad, doorbell, one or more memory windows) to be modeled to achieve NTB functionality. At least one memory window is required while more than one is permitted. All these regions should be mapped to BARs for hosts to access these regions.

If one 32-bit BAR is allocated for each of these regions, the scheme would look like this:

BAR NO	CONSTRUCTS USED
BAR0	Config Region
BAR1	Self Scratchpad
BAR2	Peer Scratchpad
BAR3	Doorbell
BAR4	Memory Window 1
BAR5	Memory Window 2

However if we allocate a separate BAR for each of the regions, there would not be enough BARs for all the regions in a platform that supports only 64-bit BARs.

In order to be supported by most of the platforms, the regions should be packed and mapped to BARs in a way that provides NTB functionality and also makes sure the host doesn't access any region that it is not supposed to.

The following scheme is used in EPF NTB Function:

BAR NO	CONSTRUCTS USED
BAR0	Config Region + Self Scratchpad
BAR1	Peer Scratchpad
BAR2	Doorbell + Memory Window 1
BAR3	Memory Window 2
BAR4	Memory Window 3

BAR NO	CONSTRUCTS USED
BAR5	Memory Window 4

With this scheme, for the basic NTB functionality 3 BARs should be sufficient.

Modeling Config/Scratchpad Region:

```
+------------------+------->+------------------+        +------------------+
|       BAR0       |        |  CONFIG REGION   |        |       BAR0       |
+------------------+----+   +------------------+<-------+------------------+
|       BAR1       |    |   |SCRATCHPAD REGION |        |       BAR1       |
+------------------+    +-->+------------------+<-------+------------------+
|       BAR2       |           Local Memory             |       BAR2       |
+------------------+                                    +------------------+
|       BAR3       |                                    |       BAR3       |
+------------------+                                    +------------------+
|       BAR4       |                                    |       BAR4       |
+------------------+                                    +------------------+
|       BAR5       |                                    |       BAR5       |
+------------------+                                    +------------------+
  EP CONTROLLER 1                                         EP CONTROLLER 2
```

Above diagram shows Config region + Scratchpad region for HOST1 (connected to EP controller 1) allocated in local memory. The HOST1 can access the config region and scratchpad region (self scratchpad) using BAR0 of EP controller 1. The peer host (HOST2 connected to EP controller 2) can also access this scratchpad region (peer scratchpad) using BAR1 of EP controller 2. This diagram shows the case where Config region and Scratchpad regions are allocated for HOST1, however the same is applicable for HOST2.

Modeling Doorbell/Memory Window 1:

```
+------------------+    +----->+----------------+-----------+------------------+
|       BAR0       |    |      |   Doorbell 1   +-----------> MSI-X ADDRESS 1  |
+------------------+    |      +----------------+           +------------------+
|       BAR1       |    |      |   Doorbell 2   +---------+ |                  |
+------------------+----+      +----------------+         | |                  |
|       BAR2       |           |   Doorbell 3   +-------+ | +------------------+
+------------------+----+      +----------------+       | +-> MSI-X ADDRESS 2  |
|       BAR3       |    |      |   Doorbell 4   +-----+ |   +------------------+
+------------------+    |      |----------------+     | |   |                  |
|       BAR4       |    |      |                |     | |   +------------------+
+------------------+    |      |      MW1       +---+ | +-->+ MSI-X ADDRESS 3||
|       BAR5       |    |      |                |   | |     +------------------+
+------------------+    +----->-----------------+   | |     |                  |
  EP CONTROLLER 1              |                |   | |     +------------------+
                               |                |   | +---->+ MSI-X ADDRESS 4  |
                               +----------------+   |       +------------------+
                                EP CONTROLLER 2     |       |                  |
                                  (OB SPACE)        |       |                  |
                                                    +------->      MW1         |
                                                            |                  |
                                                            |                  |
                                                            +------------------+
                                                            |                  |
                                                            |                  |
                                                            |                  |
                                                            |                  |
                                                            |                  |
                                                            +------------------+
                                                             PCI Address Space
                                                             (Managed by HOST2)
```

Above diagram shows how the doorbell and memory window 1 is mapped so that HOST1 can raise doorbell interrupt on HOST2 and also how HOST1 can access buffers exposed by HOST2 using memory window1 (MW1). Here doorbell and memory window 1 regions are allocated in EP controller 2 outbound (OB) address space. Allocating and configuring BARs for doorbell and memory window1 is done during the initialization phase of NTB endpoint function driver. Mapping from EP controller 2 OB space to PCI address space is done when HOST2 sends CMD_CONFIGURE_MW/CMD_CONFIGURE_DOORBELL.

Modeling Optional Memory Windows:

This is modeled the same was as MW1 but each of the additional memory windows is mapped to separate BARs.