VTA-HLS

RTL Connection

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Physical Function | BAR | Module | Host Offset | RTL Offset |
| PF 0 | BAR 0 | NVME | TBD | B001\_0000 |
| BAR 1 | PS\_DDR 0 | TBD | 4500\_0000 |
| BAR 2 | Future Approach | | |
|  | | | | |
| PF 1 | BAR 0 | Sysmon | 0000\_0000 | B003\_0000 |
| GPIO 0 | 0020\_0000 | B005\_0000 |
| GPIO 1 | 0040\_0000 | B006\_0000 |
| VTA | 0080\_0000 | B000\_2000 |
| BAR 1 | PS\_DDR 1 | TBD | 4600\_0000 |
| BAR 2 | UART\_0 | 0080\_0000 | B002\_0000 |
| UART\_2 | 0000\_0000 | B004\_0000 |

Decode Logic:

As per above RTL connections PF1\_BAR0 is connected to many modules. So, host will select the module using decode logic as shown below.

Host address has 32 bits -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31:24 | 23 | 22 | 21 | 20:0 |

Masked bits access from host

Masters of PF1 BAR0 will decode from23rd, 22nd and 21st bits.

|  |  |  |  |
| --- | --- | --- | --- |
| 23rd bit |  | | |
| 1 | VTA | |  |
| 0 | Base Registers | |
|  | 22nd bit | 21st bit |
| 0 | 0 | Sysmon |
| 0 | 1 | GPIO0 |
| 1 | 0 | GPIO1 |

Example to understand decode logic as per above table.

1. If from host side address is 5580\_0000 then according to decode logic first 8 bits are masked so the address will be 0080\_0000, where 23rd bit is 1 which will encoded for VTA.
2. If from host side address is 5520\_0000, after masking the address will be 0020\_0000 after checking 23rd bit (which is 0) then after it will check 22nd bit (which is 0) & 21st bit (which is 1) hence it indicates GPIO\_0.

GPIO is used for user defined offset address management.

GPIO has 2 data channels, here one is used for offset address and other is for size.

To access 2 PS DDR’s addresses and size 2 GPIO is used.

For each data channel 3 signals are used,

1. Input
2. Output
3. Tri state

PS\_DDR fetch their addresses from host side signal, output address will depend on tri state as per below.

1. If tri state is 32’hffff\_ffff then output has default offset address for ps ddr(4500\_0000 or 4600\_0000).
2. If tri state is 32’h0 then output has offset address which is given by user.

**RTL logic:**

ddr\_offset\_0 = (addr\_tri\_offset\_0 == 32'h0) ? ps\_ddr0\_offset\_usr :`PS\_DDR0\_OFFSET;

ddr\_offset\_1 = (addr\_tri\_offset\_1 == 32'h0) ? ps\_ddr1\_offset\_usr :`PS\_DDR1\_OFFSET;

**Example:**

* User wants to access 4700\_0000 address of ps ddr, then used below logic:
  + PF1 BAR0- 5520\_0000 is set as 0x4700\_0000
  + PF1 BAR0- 5520\_0004 is set as 0x0
  + PF1 BAR1- 5600\_0000 is used for read write of ps ddr.
  + In RTL upper bit will masked (55), 20\_0000 will represent as GPIO0.   
    It will set 4700\_0000 address for ps ddr0 memory when tri\_state (5520\_0004) is cleared (0x0).
* From pf0-bar1 channel we can directly access user address (4700\_0010) using below command.  
  ./pcimem /sys/bus/pci/devices/0000\:01\:00.0/resource1 0x10 w 0x33333333
* To access all the addresses we have concatenated MSB 8 bits (31:24) from base address (i.e.4500\_0000 default ps\_ddr address or user address given in gpio) and lower bits (23:0) from axi\_awaddr

Virtual Comport Uart

1. If we connect PL-UART to PS UART-0 at that time "zynq\_mp" console is coming and read-write is working from script (Host) to minicom.
2. In build\_root console, while booting (“boot”) linux, not able to read-write properly from script (Host) to minicom.
3. Zynq\_mp console is not coming, when PL\_UART is connected to host and PS’s AXI interface.
4. So, for solving this issue merged both connections in single RTL and used 3 UART in design.
5. PF0-BAR2 channel connects with 2 UARTs. That means one uart connected with host and another uart connected with PS’s axi channel and both uart connected with each other via tx/rx channel.
6. PF1-BAR2 channel connects with single UART. That means uart connected with host and tx/rx channel connected with PS’s UART-0.

Using this approach we are able to get both console from alkali\_shell.py (host) script and able to read - write as well. Following is the image for all uart connection.

<img src="path to your image" width="128"/>