```
aglx_hps_bridges: bridge@80000000 {
        compatible = "simple-bus";                          <physical_base_adress lenght>
        reg = <0x80000000 0x20200000>,                      <- axi_h2f physical base address and length
              <0xf9000000 0x00100000>;                      <- axi_h2f_lw physical base address and lenght
        reg-names = "axi_h2f", "axi_h2f_lw";
        #address-cells = <0x2>;
        #size-cells = <0x1>;
        ranges = <0x00000000 0x00000000 0x80000000 0x00040000>,   <- 256k OCRAM physical base address and length
                 <0x00000000 0x10000000 0x90000000 0x10000000>,   <- PCIe TXs physical base address and length
                 <0x00000000 0x20000000 0xa0000000 0x00200000>,   <- PCIe HIP physical base address and length
                 <0x00000001 0x00010000 0xf9010000 0x00008000>,   <- PCIe Cra physical base address and length
                 <0x00000001 0x00018000 0xf9018000 0x00000080>,   <- MSI vector_slave physical base address and length
                 <0x00000001 0x00018080 0xf9018080 0x00000010>;   <- MSI csr physical base address and length
```

Format of ranges
<**bridge address_offset** **physical_base_address** **length**>
*bridge 0x00000000 refers to h2f, 0x00000001 refers to h2f_lw
*bridge physical_base_address + address_offset (1st parameter + 2nd parameter) should be equal to physical_base_address (3rd parameter)

```
pcie_0_pcie_aglx: pcie@200000000 {
        compatible = "altr,pcie-root-port-3.0";             <bridge address_offset length>
        reg = <0x00000000 0x20000000 0x00200000>,           <- PCIe HIP bridge, address offset and length
              <0x00000000 0x10000000 0x10000000>,           <- PCIe TXs bridge, address offset and length
              <0x00000001 0x00010000 0x00008000>;           <- PCIe Cra bridge, address offset and length
         reg-names = "Hip", "Txs", "Cs";
         /* p-tile
         port_conf_stat = <0x104000>;*/      <- Interrupt number mapping for PCIe
         /* f-tile */
         port_conf_stat = <0x14000>;
         interrupt-parent = <&intc>;
         interrupts = <0x0 0x14 0x4>;
         interrupt-controller;
         #interrupt-cells = <0x1>;
         device_type = "pci";
         bus-range = <0x0000000 0x000000ff>;
         ranges = <0x82000000 0x00000000 0x00000000 0x00000000 0x10000000 0x00000000 0x10000000>;
         msi-parent = <&pcie_0_msi_irq>;
         #address-cells = <0x3>;
         #size-cells = <0x2>;
         dma-coherent;
         interrupt-map-mask = <0x0 0x0 0x0 0x7>;
         interrupt-map = <0x0 0x0 0x0 0x1 &pcie_0_pcie_aglx 0x1>,
                         <0x0 0x0 0x0 0x2 &pcie_0_pcie_aglx 0x2>,
                         <0x0 0x0 0x0 0x3 &pcie_0_pcie_aglx 0x3>,
                         <0x0 0x0 0x0 0x4 &pcie_0_pcie_aglx 0x4>;
        }; //end pcie@0x010000000 (pcie_0_pcie_aglx)
```

1st cell: bus number assigned to this node
2nd cell: maximum bus number of any of the subordinate PCI busses

<- This is for PCIe legacy interrupt (not used in RP MSI design)

Formato o franges
<phys.hi phys.mid phys.low bridge address_offset length_hi length_low>
*please refer to https://elinux.org/Device_Tree_Usage#PCI_Host_Bridge, PCI Address Translation for details

```
            pcie_0_msi_irq: msi@10008080 {
                    compatible = "altr,msi-1.0";                    <bridge address_offset length>
                    reg = <0x00000001 0x00018080 0x00000010>,       <- MSI CSR bridge, address offset and length
                        <0x00000001 0x00018000 0x00000080>;         <- MSI vector_slave bridge, address offset and length
                    reg-names = "csr", "vector_slave";
                    interrupt-parent = <&intc>;
                    interrupts = <0x0 0x13 0x4>;                    <- Interrupt number mapping for MSI
                    msi-controller = <0x1>;
                    num-vectors = <0x20>;
                    }; //end msi@0x100008000 (pcie_0_msi_irq)
                };
            };
};
```