Use FMAN VSP Mechanism to Implement Dedicated Buffer Pool Allocation Based on the Frame Header

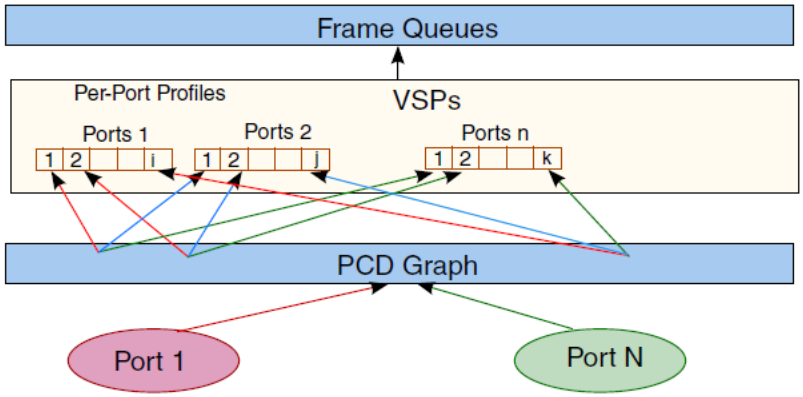
If a user is running a multiple software entities system where a single MAC may be used by several software entities simultaneously. Users may use a different VSP(Virtual Storage Profiles) for each SW entity; that way, the buffer may be private, It allows the virtualization of the buffer pool selection for frame storage (and other parameters related to storage in external memory) from the physical hardware ports.

Please refer to the following Rx packets flow in FMAN(DPAA 1.0).
a. Packets arriving in FMAN internal memory.
b. Buffer Acquisition Request
c. Get Buffer Reference from BMAN
d. Packet Data written to main memory subsystem(DDR SDRAM) managed by BMAN.
e. Parse, classify/hash, select queue

Using VSP mechanism, different packets received on the same physical port may be stored in different BM pools based on the frame header, in a similar way to FQID selection. For example, in the case allocating a dedicated buffer pool for VLAN 2 packets, there are buffers always available for VLAN 2 traffic, if too many traffic coming in, other packets can be dropped.
Each port has a default VSP. On each PCD classification, a VSP may be selected. Received packets will be written into the destination buffer according to the VSP parameters, while the VSP is selected according to the frame headers and the PCD configuration.

Usage of VSP by FMAN Port

An FMan Port may use the legacy Physical Storage Profile or the Virtual Storage Profiles (VSP). This section will discuss the usage of VSP by an FMan port.
When a user wants to set an Rx or OP port to work in virtualization mode using VSP's rather than the physical SP, user should call the function which allocates a storage profile window (range of VSPs allocated in continuously manner) to a port.
The user should also define which profile in this range should be used as default SP; note that the default profile should be a relative index within the allocated window. Upon calling the window allocation routine, the driver enables virtual mode (i.e.using VSPs) for this port, allocates its profiles and defines default SP.
In order to redirect a packet into a certain VSP, user may set the 'relative-VSP-id' within the PCD graph nodes (e.g. in the match-table entries). The value in the PCD graph nodes is port relative so if two ports are sharing the same PCD graph node (e.g. a match-table), the actual VSP will be selected by the 'relative-VSP-id' plus the port's base VSP as shown in the figure below.



Add Chosen Device node in the DTS file to support Virtual storage Profile

VSP can be enabled by adding the chosen nodes in the device tree. Please refer to the following T4240-40g-vsp.dtsi is created with VSP chosen nodes, in order to enable VSP on T4240 target board, T4240-40g-vsp.dtsi should be included in the dts file.

```
/ {
        chosen {
                name = "chosen";
                dpaa-extended-args {
                        fman0-extd-args {
                                cell-index = <0>;
                                compatible = "fsl,fman-extended-args";
                                dma-aid-mode = "port";

                                fman0_rx0-extd-args {
                                        cell-index = <0>;
                                        compatible = "fsl,fman-port-10g-rx-extended-args";
                                        vsp-window = <8 0>;
                                };
                                fman0_tx0-extd-args {
                                        cell-index = <0>;
                                        compatible = "fsl,fman-port-10g-tx-extended-args";
                                };

                                fman0_rx1-extd-args {
                                        cell-index = <1>;
                                        compatible = "fsl,fman-port-10g-rx-extended-args";
                                        vsp-window = <8 0>;
                                };
                                fman0_tx1-extd-args {
                                        cell-index = <1>;
                                        compatible = "fsl,fman-port-10g-tx-extended-args";
                                };
                        };
                        fman1-extd-args {
                                cell-index = <1>;
                                compatible = "fsl,fman-extended-args";
                                dma-aid-mode = "port";

                                fman1_rx0-extd-args {
                                        cell-index = <0>;
                                        compatible = "fsl,fman-port-10g-rx-extended-args";
                                        vsp-window = <8 0>;
                                };
                                fman1_tx0-extd-args {
                                        cell-index = <0>;
                                        compatible = "fsl,fman-port-10g-tx-extended-args";
                                };

                                fman1_rx1-extd-args {
                                        cell-index = <1>;
                                        compatible = "fsl,fman-port-10g-rx-extended-args";
                                        vsp-window = <8 0>;
                                };
                                fman1_tx1-extd-args {
                                        cell-index = <1>;
                                        compatible = "fsl,fman-port-10g-tx-extended-args";
```

```
                                        };
                                };
                        };
                };
};
```

VSP Allocation and Initialization in USDPAA Application

It's required to implement VSP initialization function in USDPAA application code to allocate a specified buffer pool to every VSP with automatically and randomly way. Please refer to the following sample code for VSP allocation invoked in USDPAA application code, 3 VSPs are defined in the application, everyone of them could select one of the 3 available buffer pools(1,2 or 3) created by the application.

```
struct bpools bpool[] = {
        {-1, VSP_BP_SIZE, VSP_BP_NUM},
        {-1, VSP_BP_SIZE, VSP_BP_NUM},
        {-1, VSP_BP_SIZE, VSP_BP_NUM}
};

static t_Handle         vsp[NUM_VSP];
static t_Handle         fm_obj;

int vsp_init(int fman_id, int fm_port_number)
{
    t_FmBufferPrefixContent fmBufferPrefixContent;
    t_FmVspParams       fmVspParams;
    int             i, ret = E_OK;

    ret = bpool_init();
    if (ret < 0) {
            error(0, ret, "Buffer pools init failed\n");
            return -ENOMEM;
    }

    fm_obj = FM_Open(fman_id);
    if (!fm_obj) {
        ret = -EINVAL;
        error(0, ret, "FM_Open NULL handle.\n");
        return ret;
    }

    memset(&fmVspParams, 0, sizeof(fmVspParams));
      fmVspParams.h_Fm = fm_obj;
    fmVspParams.portParams.portId = fm_port_number;
    fmVspParams.portParams.portType = e_FM_PORT_TYPE_RX;
    fmVspParams.extBufPools.numOfPoolsUsed = 1;

    /*VSP  buffer prefix configuration*/
    memset(&fmBufferPrefixContent, 0, sizeof(fmBufferPrefixContent));
    fmBufferPrefixContent.privDataSize = 16;
    fmBufferPrefixContent.passPrsResult = TRUE;
    fmBufferPrefixContent.passTimeStamp = TRUE;
    fmBufferPrefixContent.passHashResult = FALSE;
    fmBufferPrefixContent.passAllOtherPCDInfo = FALSE;
    fmBufferPrefixContent.dataAlign = 64;

    memset(vsp, 0, NUM_VSP * sizeof(t_Handle));
```

```
	for (i = 0; i < NUM_VSP; i++) {
		fmVspParams.relativeProfileId = i;
		fmVspParams.extBufPools.extBufPool[0].id = bpool[i].bpid;
		fmVspParams.extBufPools.extBufPool[0].size = bpool[i].size;

		vsp[i] = FM_VSP_Config(&fmVspParams);
		if (!vsp[i]) {
			ret = -EINVAL;
			error(0, ret, "FM_VSP_Config NULL vsp handle for "
					"vspid %d.\n", i);
			return ret;
		}

		ret = FM_VSP_ConfigBufferPrefixContent(vsp[i],
				&fmBufferPrefixContent);
			ret = FM_VSP_ConfigBufferPrefixContent(vsp[i],
				&fmBufferPrefixContent);
		if (ret != E_OK) {
			error(0, ret, "FM_VSP_ConfigBufferPrefixContent error "
					"for vspid %d; err: %d\n", i, ret);
			return ret;
		}

		/* VSP final configuration */
		ret = FM_VSP_Init(vsp[i]);
		if (ret != E_OK) {
			error(0, ret, "FM_VSP_Init error: %d\n", ret);
			return ret;
		}

	}

	return ret;
}
```

Define VSP in FMC PCD Policy File

The virtual storage profile is only available on famer manager v3. The configuration is transparent to the user. Please refer to the following sample, three storage profiles are defined in the policy file, the first storage profile, Default_VSP is also the default storage profile of the port. The second and the third storage profiles are used for IPv4 and IPv6 flows. Each of the storage profiles described above has a set buffer pools associated with it.

The buffer pools are dynamically allocated and initialized by the USDPAA application.

When an IPv4 flow is reassembled / classified on inbound port, the IPv4_Reass_VSP is used and the buffers are selected from one of the buffer pools available for this VSP. When an IPv6 flow is reassembled/classified on inbound port, the IPv6_Reass_VSP with buffers from one of the buffer pools available for this VSP is used. For any other flows the Default_VSP is selected.
The VSP selection mechanism described in this paragraph is totally transparent to the user and doesn't affect the use case behavior in any way.

```
	<vsp name="Default_VSP" base="0"/>
	<vsp name="IPv4_Reass_VSP" base="1"/>
	<vsp name="IPv6_Reass_VSP" base="2"/>

	<classification name="ipv4_udp" statistics="frame">
```

```
        <key>
                <fieldref name="udp.dport" />
        </key>
        <entry>
                <data>0x868</data>
                <queue base="0x2e01" />
                <vsp name="IPv4_Reass_VSP" />
                <action statistics="enable" />
        </entry>
</classification>

<classification name="ipv4_udp_first_frag" masks="yes">
        <key>
                <fieldref name="ipv4.foffset" />
        </key>
        <entry>
                <data>0x0000</data>
                <mask>0x1FFF</mask>
                <action type="classification" name="ipv4_udp" />
        </entry>
</classification>

<!-- Coarse classification for IPv6 source IP-->
<classification name="reass_ipv6_classif" statistics="frame">
        <key>
                <fieldref name="ipv6.nexthdr" header_index="last" />
        </key>
        <entry>
                <data>0x11</data>
                <vsp name="IPv6_Reass_VSP"/>
                <queue base="0x1881"/>
                <action statistics="enable"/>
        </entry>
</classification>

<distribution name="default_dist">
        <vsp name="Default_VSP"/>
        <queue count="1" base="0x66"/>
</distribution>
```