1. Mailboxes

1.1 Comunicarea prin mailbox-uri intre PPE si SPE-uri

PPE-ul comunica cu SPE-urile prin registrii MMIO(memory mapped IO) privileged-state si problem-state controlati de MFC pe fiecare SPE. Acesti registrii sunt accesati de SPE-ul asociat prin canalele proprii (registri unidirectionali si cozi) si suport logic. Cele trei mecanisme importante de comunicare intre PPE si SPE sunt mailbox-urile, registrii de notificare prin semnale si DMA.

Mailbox-urile sunt cozi care permit trimiterea de mesaje scurte, de 32 de biti. Fiecare mailbox are asociat cate un canal SPE si cate un registru MMIO corespunzator. Exista trei tipuri de mailbox-uri. Doua din ele SPU Write Outbound Mailbox si the SPU Write Outbound Interrupt Mailbox) sunt pentru mesaje de la SPE spre PPE, una (SPU Read Inbound Mailbox) e pentru mesaje de la PPE ls SPE.

Nota: Mailbox-urile pot fi folosite si ca mecanism de sincronizare intre SPE-uri. Unul din SPE-uri trimite prin prin DMA date in mailbox-ul celuilalt SPE, folosind adresa efectiva a maparii memoriei PS(problem state).

In figura de mai jos avem un overview cu functiile pentru mailbox-uri la PPU si SPU. Detalii despre aceste functii gasiti pentru PPU aici (Attach:IBM SPE Runtime Management Library.pdf) iar pentru PPU & SPU aici (Attach:IBM C/C++ Language Extensions for Cell Broadband Engine Architecture v.2.5.pdf)

1. **Outbound Mailboxes** - SPE scrie, PPE citeste.
    1. **SPU scrie in Outbound Mailbox**
        - Capacitate de un singur mesaj pe 32 biti
        - SPE se blocheaza (apare stall) daca mailbox-ul e plin, pana cand PPE sau altcineva citeste mesajul respectiv
        - Daca mailbox-ul accepta mesajul, contorul asociat canalului e decrementat cu 1. Cand mailbox-ul e plin, contorul va fi 0.
        - Dupa ce mesajul din mailbox e citit (de PPE), contorul creste cu 1 (pentru canal gol e 1, avand in vedere ca poate tine maxim un mesaj)
    2. **SPU scrie in Outbound Interrupt Mailbox**
        - Capacitate de un singur mesaj pe 32 biti
        - SPE se blocheaza (apare stall) daca mailbox-ul e plin, pana cand PPE sau altcineva citeste mesajul respectiv

- Daca mailbox-ul accepta mesajul (initial e gol), contorul asociat canalului e decrementat cu 1 si se va actiona o intrerupere in PPE. Nu se garanteaza o ordine a intreruperilor si a comenzilor MFC anterioare.
- Cand mailbox-ul e plin, contorul va fi 0.
- Dupa ce mesajul din mailbox e citit (de PPE), contorul creste cu 1 (devine 1)

3. **PPU citeste din SPU Outbound (si Interrupt) Mailbox**
   - Mai intai se verifica Mailbox Status Register (mai exact campul SPU_Out_Mbox_Count), pentru a vedea daca e vreun mesaj de citit din SPU Outbound Mailbox sau SPU Outbound Interrupt Mailbox
   - Daca SPU_Out_Mbox_Count e 0, PPE nu trebuie sa citeasca, ci sa faca poll pe Mailbox Status register. Daca mailbox-ul e gol, PPE se blocheaza incercand sa citeasca sau intoarce o valoare nedefinita
   - Daca SPU_Out_Mbox_Count e 1, un mesaj e prezent in coada. (invers decat contorul de la SPE)

1.2 Outbound Interrupt Mailbox - Lucrul cu eventuri

Ideea in folosirea Outbound Interrupt Mailbox e sa nu facem busy-waiting la PPE pentru a vedea cand vine un mesaj de la vreun SPE. Daca lucrul cu celelale doua mailboxuri este oarecum straitforward, aici trebuie sa fim putin atenti la detalii in codul din PPE; la SPE scrierea se face la fel ca in Outbound Mailbox. Detalii despre functiile folosite in lucrul cu eventuri gasiti in *pag 28-34*, aici (Attach:IBM SPE Runtime Management Library.pdf)

```
1.
2. spe_event_unit_t pevents[NO_SPU], events_received[NO_SPU];
3. spe_event_handler_ptr_t event_handler;
4. event_handler = spe_event_handler_create();
5.
6. /*unde spe_event_unit_t e (pre)definit astfel:
7. typedef struct spe_event_unit
8. {
9. unsigned int events;
10.     spe_context_ptr_t spe;
11.     spe_event_data_t data;
12.     } spe_event_unit_t;
13.     */
14.
15.     //In laboratorul 8 am vazut ca daca vrem sa lucram cu
   eventuri trebuie sa specificam acest lucru inca de la
   crearea contextelor:
16.
17.     ctx[i] = spe_context_create(SPE_EVENTS_ENABLE,NULL);
18.
19.     // precizam tipul de eventuri cu care vom lucra
20.     pevents[i].events = SPE_EVENT_OUT_INTR_MBOX;
```

```
21.     pevents[i].spe = ctx[i]; //asociem cate un context
   eventurilor
22.
23.     //In Outbound Interrupt Mailbox PPE poate primi mesaj
   de la orice SPE; vrem sa stim exact de la ce SPE vine
   mesajul, asa ca vom asocia un numar fiecarui spe, numar
   care va fi continut si in mesaj.
24.     //Acest numar ni se va intoarce nemodificat in
   spe_event_wait(), atunci cand se va primi un event de la
   speul asociat contextului
25.     pevents[i].data.u32 = i;
26.
27.     //inregistam un handler pentru eventuri
28.     spe_event_handler_register(event_handler, &pevents[i]
   );
29.     //Asteptarea unui event in PPE:
30.     spe_event_wait(handler, events_received, NO_SPU, 1);
31.
32.     printf("Am primit ceva de la speul %d:",
   events_received[0].data.u32);
33.     //PPE citeste date din mailboxul de intreruperi
   corespunzator speului de la care am primit eventul
34.     spe_out_intr_mbox_read
   (events_received[0].spe,(unsigned int*) &data, 1,
   SPE_MBOX_ANY_BLOCKING);
35.
36.     //PPE scrie date in mailboxul INBOUND corespunzator
   speului de la care a primit mesajul
37.     spe_in_mbox_write( events_received[0].spe, msg, 1,
   SPE_MBOX_ANY_BLOCKING);
```

[Get Code]

1.3 Inbound Mailbox Channel - PPE scrie, SPE citeste

1. **SPU citeste din Inbound Mailbox Channel**
   - Mailbox-ul este o coada FIFO, cu capacitatea de 4 mesaje de 32 biti
   - Daca mailbox-ul are cel putin un mesaj, valoarea citita de SPU este cea mai veche valoare scrisa in mailbox.
   - SPU vede contorul 0 pentru canal gol; SPU se blocheaza (stall) daca incearca sa citeasca din mailbox gol; ramane blocat pana PPE scrie
   - La fiecare scriere, contorul se incrementeaza cu 1; la fiecare citire, contorul se decrementeaza cu 1
2. **PPU scrie in Inbound Mailbox Channel**
   - Contorul din perspectiva PPU-ului (din nou, invers decat la SPU) este
     - 0 pentru mailbox gol
     - numarul de intrari libere de 32 biti (maxim posibile 4) din mailbox

- Spre deosebire de scrierea in celelalte mailbox-uri, PPU nu se blocheaza la scriere; daca nu mai e loc, se va suprascrie ultimul mesaj