Semnalizarea handshake pentru transferuri de date

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

18 noiembrie 2024

Introducere

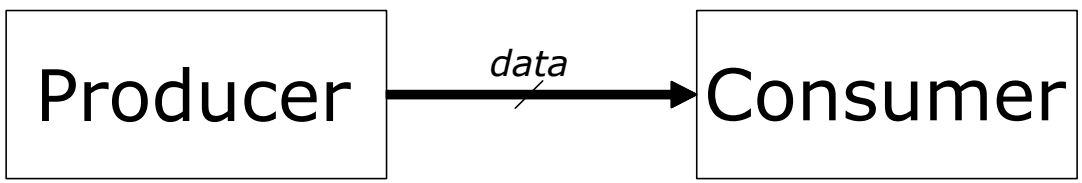
Obiective:

- Utilizarea semnalizării handshake pentru transfer fiabil între componente digitale

De citit:

- Chris Fletcher: "EECS150: Interfaces: "FIFO" (a.k.a. Ready/Valid)", [Flet09c]

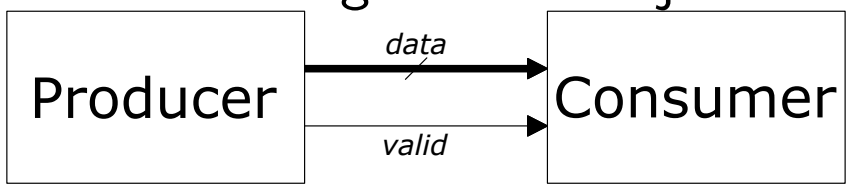
Semnalizarea handshake permite adaptarea ratei de transfer a datelor între componente digitale. Fără a pierde din generalitate, se vor considera două componente secvențiale sincrone, una care generează date (producător) și una care consumă date (consumator). Componentele schimbă pachete de date folosind o magistrală comună.



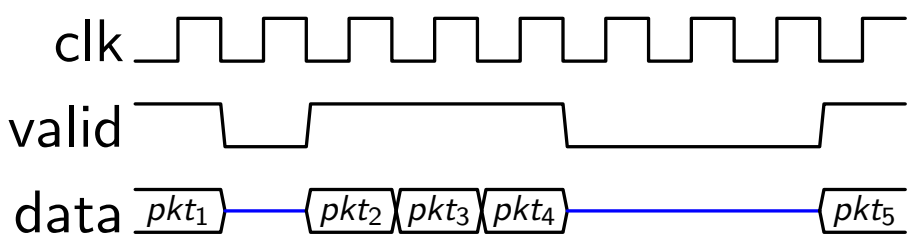
Ajustarea ratei de transfer

Dacă producătorul poate genera un pachet în fiecare ciclu de tact iar consumatorul îl poate procesa în același ciclu, semnalizarea între cele două componente nu este necesară.

Dacă producătorul nu poate genera pachete la viteza cu care consumatorul le procesează, se adaugă ieșirea *valid* producătorului, activată când un pachet a fost plasat pe liniile de date partajate. Configurația este descrisă în figura de mai jos:

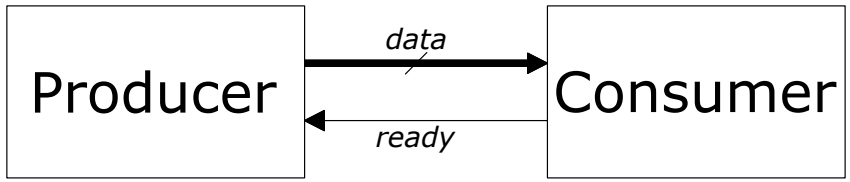


Consumatorul inspectează, în fiecare ciclu de tact, ieșirea *valid* a producătorului și, când aceasta este activă, preia pachetul de date, altfel, așteaptă activarea sa.

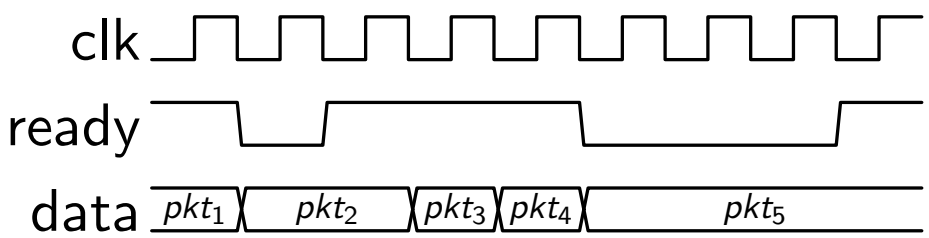


Ajustarea ratei de transfer (contin.)

Dacă consumatorul nu poate procesa pachetele la viteza de generare a acestora de către producător, se adaugă ieșirea *ready* consumatorului, activată când acesta poate primi un pachet nou. Configurația este descrisă mai jos:

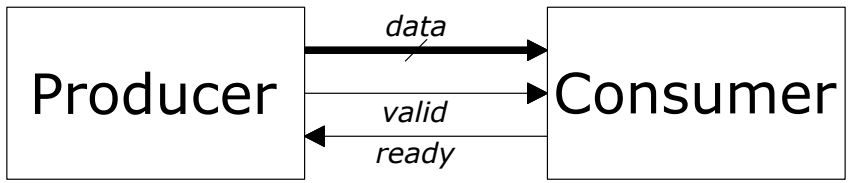


Producătorul încarcă pachetul curent pe liniile de date după care inspectează linia *ready*. Dacă este activă, acesta continuă cu generarea următorului pachet, altfel păstrează datele curente. Diagrama de timp de mai jos exemplifică transferul:

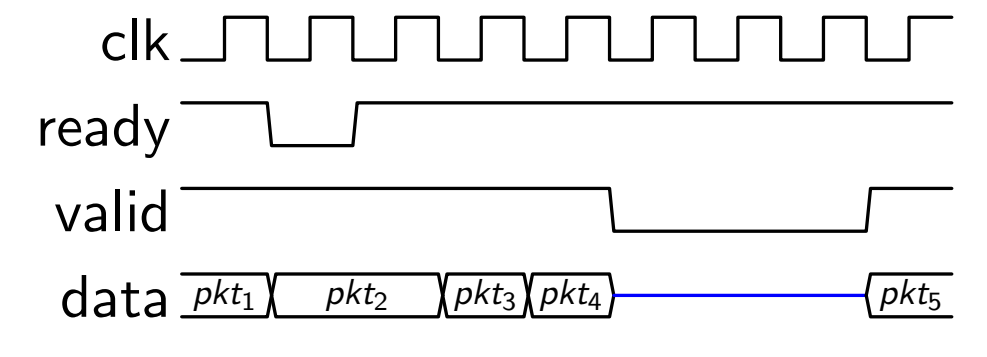


Semnalizare handshake

Dacă atât producătorul cât și consumatorul au nevoie de timp pentru generarea/procesarea datelor, poate fi utilizat un protocol valid/ready. Semnalele *valid* și *ready* vor fi utilizate conform considerațiilor anterioarele. Configurația este ilustrată mai jos:



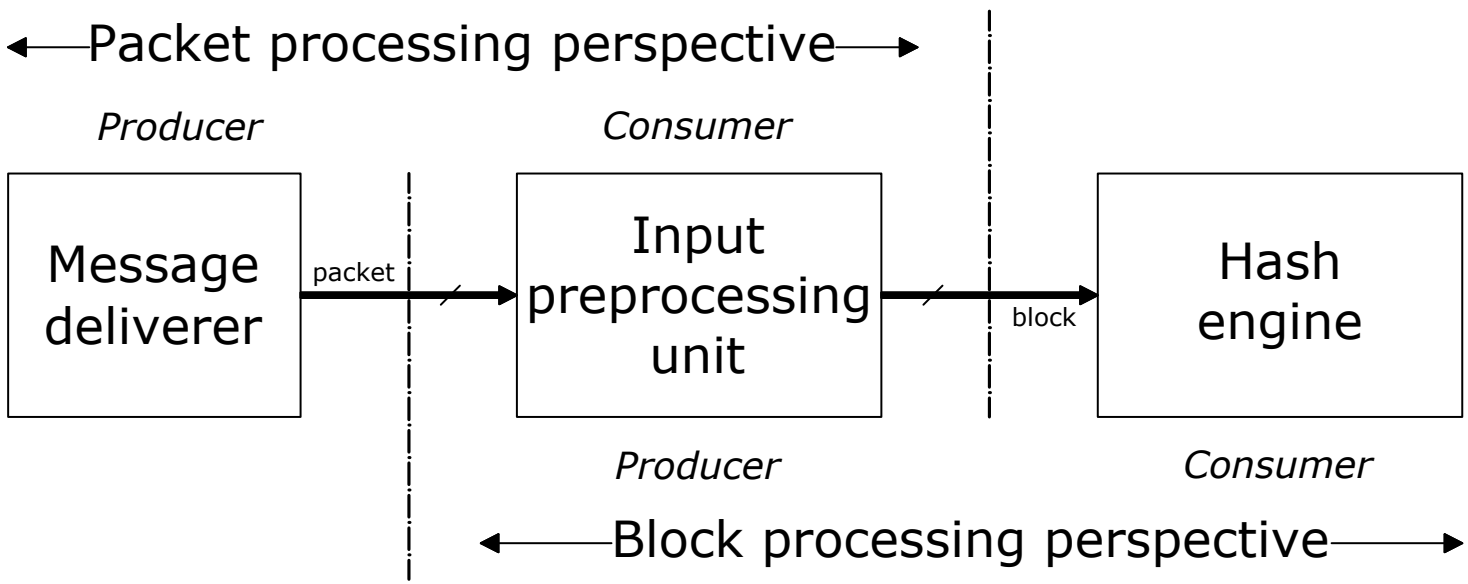
Acest protocol este cunoscut ca interfață FIFO sau "Ready/Valid". Transferul de date are loc pe frontul crescător al tactului, când ambele semnale *valid* și *ready* sunt active.



Studiu de caz

Unitatea de control a modulului de preprocesare a intrării pentru o aplicație criptografică

Unitatea de preprocesare a intrării (IPU sau *unitatea*) a Secure Hash Algorithm 2 (SHA-2) primește mesajul în pachete de 64-biți, le asamblează și le livrează în blocuri de 512-biți, nucleului *hash* [FIPS15]. După primirea completă a mesajului, unitatea îl extinde și atașază lungimea mesajului. Fluxul de date al unității este ilustrat mai jos. În fiecare perspectivă o componentă generează date și alta le consumă.



Studiu de caz (contin.)

Perspectiva procesării pachetelor

Pentru concizia prezentării, comunicarea cu *furnizorul de mesaj* nu necesită semnalizare handshake. După activarea semnalului de reset, *rst_b*, în fiecare ciclu de ceas, un nou pachet este livrat unității. Ultimul pachet al mesajului este marcat de furnizor prin activarea ieșirii *lst_pkt*.

Unitatea primește și stochează un nou pachet în fiecare ciclu de tact, furnizând *nucleului hash* un bloc nou la fiecare 16 cicluri de ceas. După recepționarea ultimului pachet (*lst_pkt* activ), unitatea adaugă un pachet extensie și, dependent de lungimea mesajului, atașază 0, 1 sau mai multe pachete zero. În ultimul bloc livrat la ieșire, unitate atașază în cei mai puțin semnificativi 64 de biți lungimea mesajului în biți.

Studiu de caz (contin.)

Perspectiva procesării blocurilor

Pentru comunicarea cu nucleul hash se folosește semnalul *valid* numit *blk_val*, activat de unitate când un bloc nou este disponibil la ieșirea sa. La livrarea ultimului bloc, odată cu activarea lui *blk_val* unitatea activează și ieșirea *msg_end* marcând terminarea transmiterii mesajului.

Unitatea de control a modulului de preprocesare a intrării:

- descrie faza de preprocesare SHA-2
- gestionează calea de date a unității
- evaluează semnalele *furnizorului de mesaj*
- semnalează condițiile specifice *nucleului hash*

Studiu de caz (contd.)

Algoritmul de preprocesare a intrării pentru SHA-2

```
1:  procedure DeliverMessage
2:      LungimeMesaj ← 0                                  ▷ setarea lungimea mesajului la 0
3:      index ← 0                                  ▷ setarea la 0 a indexului curent în RegisterFile
4:      loop
5:          RegisterFile[index] = pachet          ▷ stochează pachetul curent în RegisterFile
6:          index ← (index + 1) mod 2^3                    ▷ incrementare index RegisterFile
7:          LungimeMesaj ← LungimeMesaj + 64            ▷ incrementare lungime mesaj
8:          if index == 0 then semnalează bloc nou
9:          if lst_pkt == 1 then                                  ▷ începere extindere mesaj
10:             RegisterFile[index] = 64'h8000000000000000       ▷ 1 urmat de 63 de 0
11:             index ← (index + 1) mod 2^3
12:             while index ≠ 7 do          ▷ dacă index este 7, atașază lungime mesaj
13:                 if index == 0 then semnalează bloc nou
14:                 RegisterFile[index] = 64'h0000000000000000          ▷ pachet zero
15:                 index ← (index + 1) mod 2^3
16:             end while
17:             RegisterFile[index] = LungimeMesaj                ▷ pachet lungime mesaj
18:             semnalează bloc nou
19:             semnalează final mesaj          ▷ mesajul a fost transmis în întregime
20:             break                                  ▷ părăsește bucla de transmitere
21:     end loop
22: end procedure
```

Studiu de caz (contd.)

Calea de control a unității

Semnalele utilizate de calea de control pentru gestionarea căii de date:

- *c_up*: incrementarea contorului care stochează indexul în register file
- *clr*: ștergerea registrului care stochează lungimea mesajului și contorului care stochează indexul în register file
- *mgln_pkt*: atașarea pachetului lungime de mesaj
- *pad_pkt*: atașare pachet extensie
- *st_pkt*: stocare pachet curent în register file
- *zero_pkt*: atașază pachet zero

Unitatea de control are o intrare dedicată indexului curent în register file, numită *idx*.

Studiu de caz (contd.)

Interfațarea unității de control cu furnizorul de mesaj și cu nucleul hash

Semnale folosite pentru interfațarea cu furnizorul de mesaj:

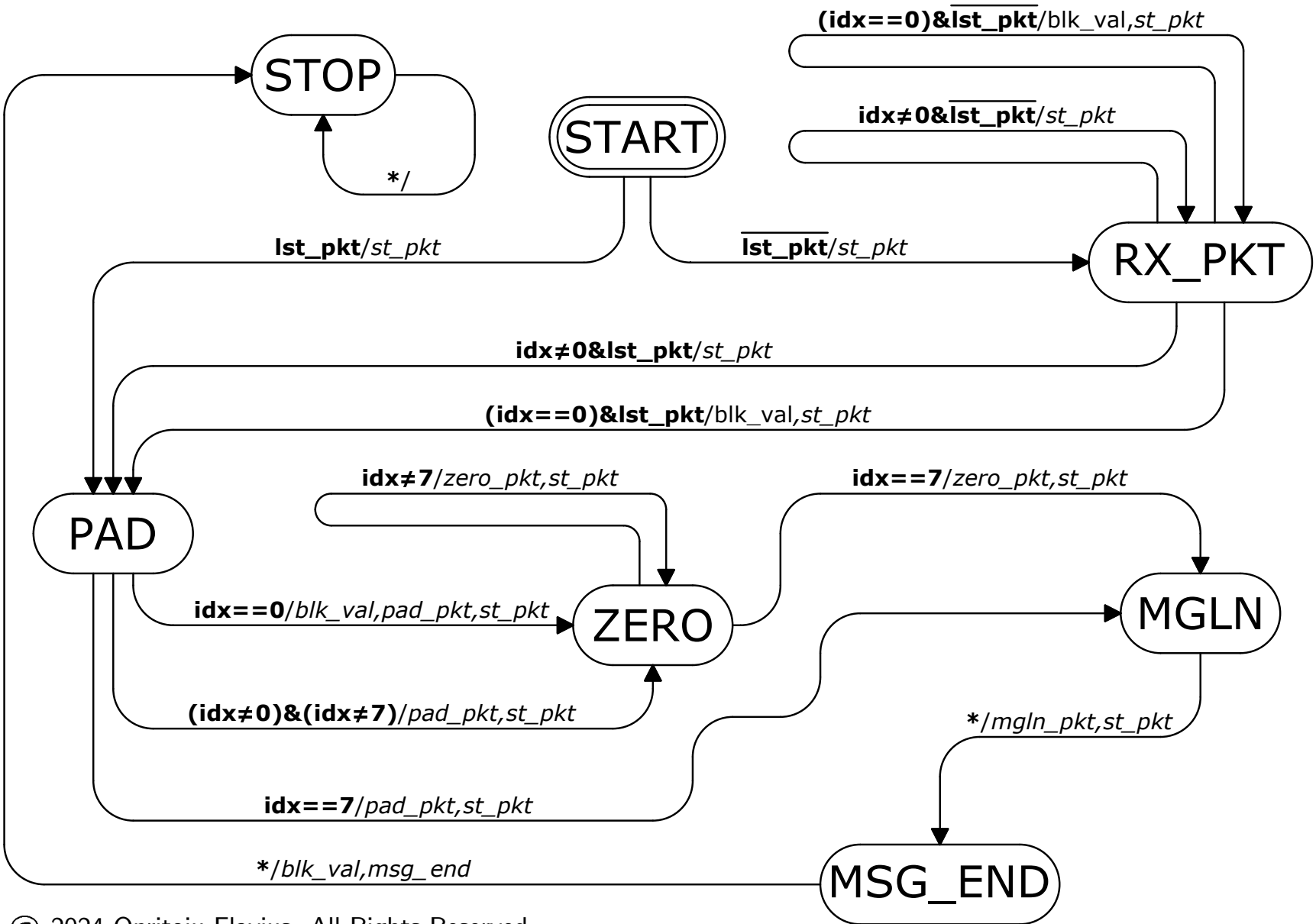
- *lst_pkt* (intrare): asertat la primirea ultimului pachet
- *pkt* (intrare, 64-biți): pachetul mesaj curent

Semnale folosite pentru interfațarea cu nucleul hash:

- *blk* (ieșire, 512-biți): blocul curent
- *blk_val* (ieșire): asertat când un bloc nou este disponibil
- *msg_end* (ieșire): asertat la livrarea ultimului bloc

Studiu de caz (contd.)

Diagrama tranzițiilor de stare ale unității de control



Studiu de caz (contd.)

Diagrama tranzițiilor de stare ale unității decontrol

Unitatea de control a modulului de preprocesare este o mașină cu stări finite Mealy.

Stările interne ale mașinii:

- **START**: starea inițială, după activarea lui *rst_b*
- **RX_PKT**: recepționează un pachet nou pînă când *lst_pkt* este activat
- **PAD**: atașază pachet extensie
- **ZERO**: atașază pachet zero
- **MGLN**: atașază pachet lungime mesaj
- **MSG_END**: anunță încheierea transmiterii mesajului
- **STOP**: stare finală, nu se activează nicio ieșire

Referințe bibliografice

[Flet09c] C. Fletcher. EECS150: Interfaces: "FIFO" (a.k.a. Ready/Valid). [Online]. Available: https://inst.eecs.berkeley.edu/~cs150/Documents/Interfaces.pdf (Last accessed 25/10/2017).

[FIPS15] National Institute of Standards and Technology, "FIPS PUB 180-4: Secure Hash Standard," Gaithersburg, MD 20899-8900, USA, Tech. Rep., Aug. 2015. [Online]. Available: http://dx.doi.org/10.6028/NIST.FIPS.180-4 (Last accessed 06/04/2016).