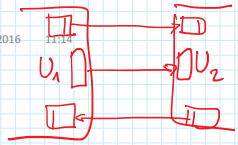
martedì 18 ottobre 2016 11:14

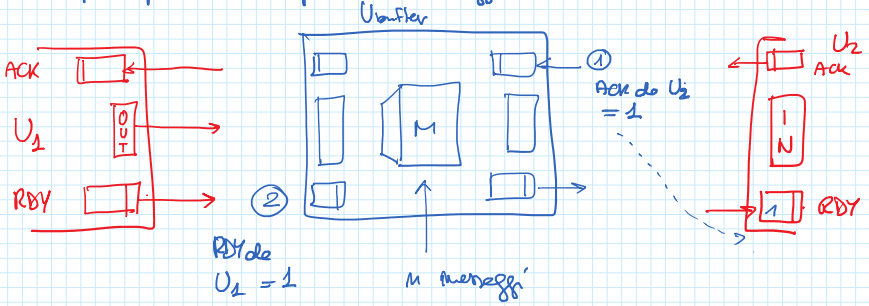


asincrona (1 posizione) simmetrica

asincrone (n-posizioni) asimmetriche

ASINCRONA (con un buffer da: n-posizioni)

mittente può spedire al più n messaggi senza che il destinatario effettui ricezioni.



① se msg in M ⇒ rendere uno col U_2

② se c'è posto in M ⇒ ricevi messaggio da U_1 e memorizzalo in M

①② M è vuota ⇒ il messaggio in arrivo da U1 passato a U2

μ-codice di Ubuffer avrà come var di condizionamento RDY da U_1 ACK da U_2

Memoria piena Memoria vuota

M 2^k posizioni CONT k+1 bit

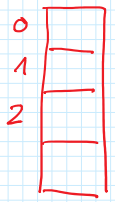
Memoria vuota OR(CONT)

Memoria piena $CONT_\emptyset$

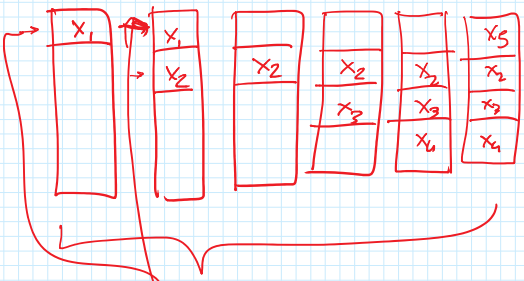
$\emptyset$. $(RDY_1, ACK_2, OR(CONT), CONT_\emptyset = 00--)$ map, $\emptyset$

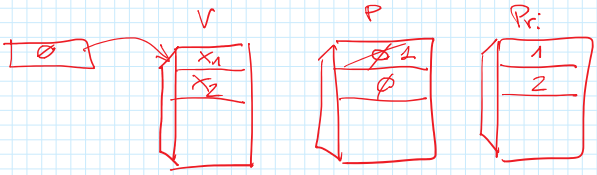
(= ...

BUFFER implementato con politica FIFO



U_1 spedisce x_1
U_1 spedisce x_2
U_2 riceve
U_1 spedisce x_3
U_1 spedisce x_4
U_1 " x_5





primo cella vuota

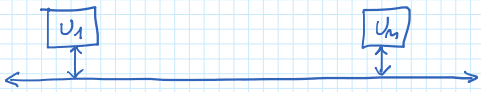
ultima cella piena

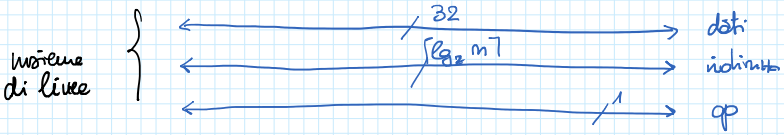
(++) % 2^k

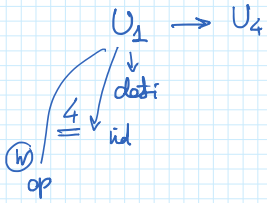
ASIMMETRICHE (COMUNICAZIONI)

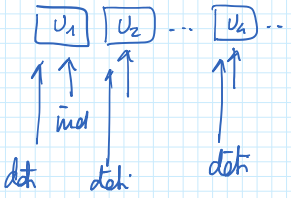
martedì 18 ottobre 2016 11:42

BUS









Serve un meccanismo di arbitraggio (politica)

- centralizzati
- distribuiti

Protocollo di interazione U/Bus

1) Richiesta uso della risorsa (del Bus) (all'arbitro)
2) comunicazione (utilizzo il bus)
3) Rilascio della risorsa

Problema della sincronizzazione degli accessi

⇒ Servono degli indicatori { a livello / a transizione di livello ← NON si possono utilizzare (per costruzione)

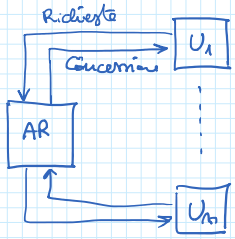
Bus supporto { punto a punto (simmetrici) / uno a molti (asimmetrici (Broadcast, Multicast)) }

1 a tutti — 1 a un sottoinsieme di tutti

ARBITRI (centralizzati)

martedì 18 ottobre 2016 12:13

Arbitro a RICHIESTE INDIPENDENTI



AR: accorda le richieste
fa le richieste, con una politica P
individua l'unità "richiedente"
assegna la richiesta all'unità
attende il rilascio della risorsa assegnata

	U_1	U_2	U_3	U_4
Rich	1	1	0	0
Concess	1	0	0	0
Rich	0	1	0	0
Conc	0	0	0	0
	0	1	0	0

U_1 utilizza (il bus)
allo fine rette la rich. a $\emptyset$

$\emptyset.\ (OR(R_1 R_2 R_3 R_4) = 0)$ nop, $\emptyset$

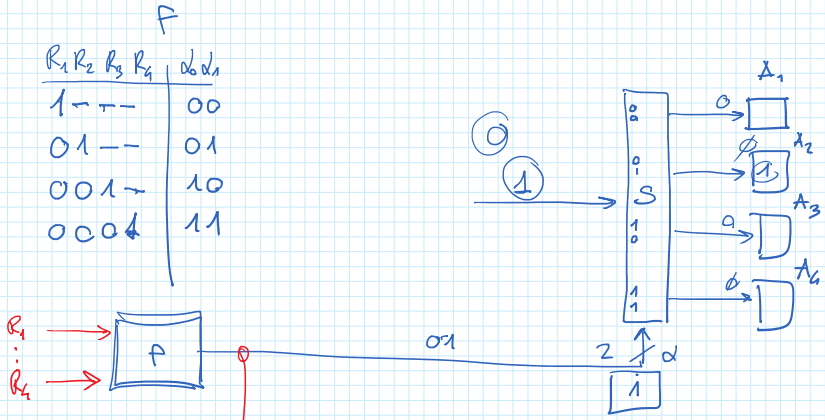
$OR(R_1 \dots R_4) = 0$

$\emptyset.\ (R_1 R_2 R_3 R_4 = 0000)$ nop, $\emptyset$

$(= 1000)$ set A_1, 1
$(= 0100)$ set A_2, 2
$(= 0010)$ . . .
$(= 0001)$ . . .
$(= 1---)$ set A_1, 1
$(= 01--)$ set A_2, 2
$(= 001-)$
$(= 0001)$

{ politica: priorità all'U_i con i + bassi

1. $(R_1 = 1)$ nop, 1
   $(= 0)$ reset A_1, $\emptyset$
2. $(R_2 = 1)$ nop, 2
   $(= 0)$ reset A_2, $\emptyset$

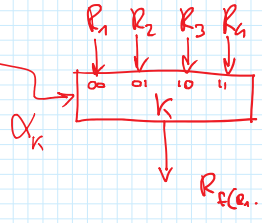


Set A_i

reset A_i

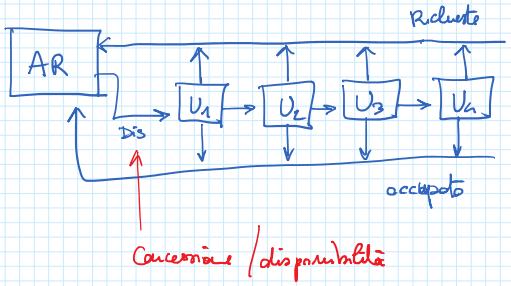
$\emptyset.\ (or(R_1 R_2 R_3 R_4) = \emptyset)$ nop, $\emptyset$
$(= 1)$ $1 \to A_{f(R_1 R_2 R_3 R_4)}$, 1

1. $(R_{f(R_1 R_2 R_3 R_4)} = 1)$ nop, 1
   $(= \emptyset)$ $0 \to A_{f(R_1 \dots R_4)}$, $\emptyset$



Centralizzato : Daisy chaining

martedì 18 ottobre 2016 12:37



wired OR → Stessa politica del caso precedente (controllo rendezvous)

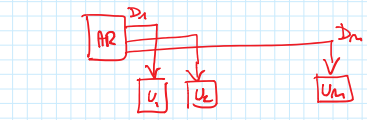
→

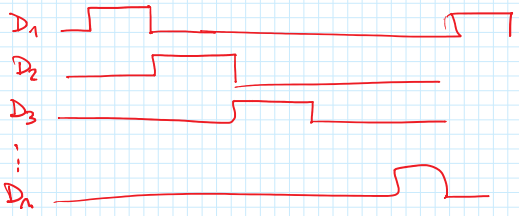
AR: Se c'è una richiesta
manda il segnale di disponibilità
attende che il bit di occupato = 1
attende che " " " " = 0

Polling

(Divisione di tempo)

AR: aspetta una richiesta (linea wired OR)
↳ interroga le unità
politica
↳ scelta unità ricettrice

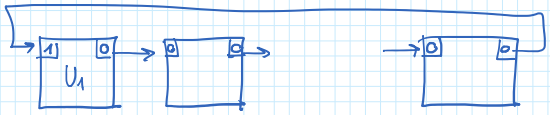




arbitri decentralizzati

martedì 18 ottobre 2016 12:50

disciplina circolare (token ring)



Se U_i riceve un 1 in ingresso
e deve utilizzare la risorsa
→ { utilizza }
passa l'"1" nell'uscita

non deterministici

modo + rilevare "collisioni"