BUS E SPAZI DI INDIRIZZAMENTO

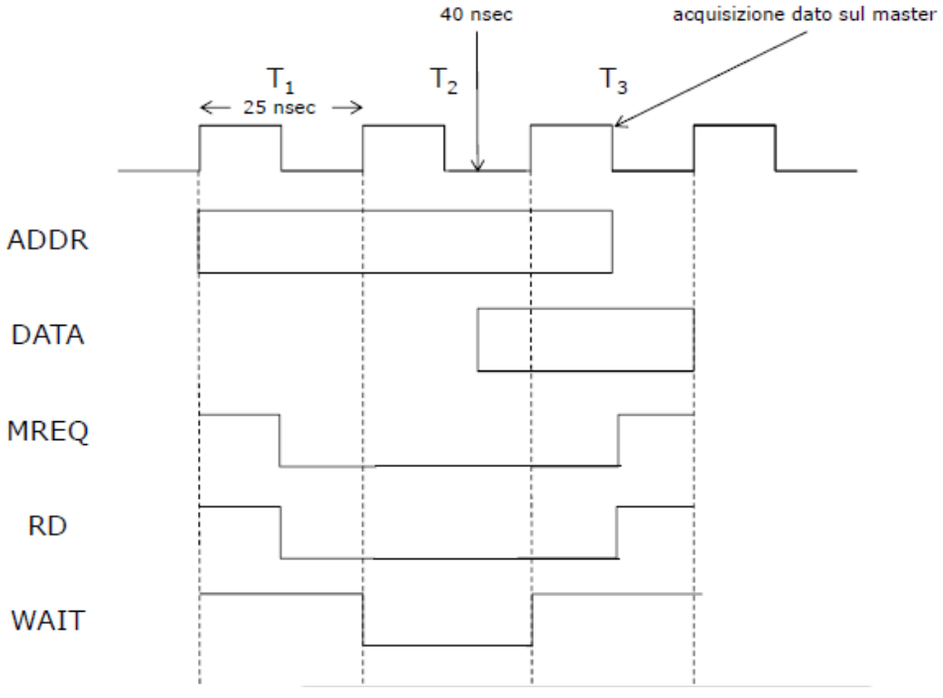
Il bus serve per la comunicazione tra i dispositivi nel calcolatore e per l'accesso ai dati e alle istruzioni. Un bus è composto da più linee, su ognuna delle quali viaggia un segnale. Il problema di aumentare la velocità di propagazione nel bus è che aumenta il bus skew (differenza di velocità tra due segnali che viaggiano su diverse linee).
Alcuni bus famosi: ISA, AGB (per schede grafiche), PCI (può trasferire 64 bit, arbitraggio centralizzato), PCI-express (connessione punto-a-punto, trasmissione seriale), USB (trasmissione seriale, non si può connetterne più di uno in una porta).
In ogni ciclo in un bus comunicano solo due dispositivi: il master e lo slave. Lo stesso dispositivo può avere ruoli diversi in diversi momenti.
In un bus sincrono i trasferimenti vengono gestiti sulla base del ciclo di clock.
In un bus asincrono i trasferimenti non avvengono sulla base del clock, ma tramite segnali MSYN e SSYN.
L'arbitro del bus gestisce i trasferimenti sul bus per evitare conflitti, assegnando a uno dei dispositivi il ruolo di bus master, permettendo così a quel dispositivo di utilizzare il bus. I dispositivi fanno richiesta di utilizzare il bus, e l'arbitro glielo concede se non c'è già un bus master.
Normalmente i dispositivi più vicini all'arbitro sono i più privilegiati, ma si può assegnare a ciascun dispositivo una priorità, in modo che un dispositivo può utilizzare il bus prima di quelli con priorità minore ma non prima di quelli con priorità maggiore.
Esiste anche un sistema senza arbitro, cioè l'arbitraggio decentralizzato, in cui in qualsiasi momento un dispositivo può utilizzare il bus se una linea definita busy (occupato) non è asserita da un altro dispositivo. Ovviamente un dispositivo che trova la linea busy non asserita, la asserisce e poi utilizza il bus.

Esercizio 1
Con riferimento al funzionamento dei bus di un calcolatore:

- **tracciare e illustrare il diagramma di temporizzazione di un bus <u>sincrono</u> a 40 Mhz <u>con linee separate per dati e indirizzi</u> e segnali di MREQ, RD e WAIT, per una lettura da una memoria con un tempo di risposta di 40 nsec *dal momento in cui gli indirizzi sono disponibili.***

Si assuma di lavorare in condizioni ideali (nessun ritardo nell'asserimento dei segnali)

Prima di tutto determiniamo il periodo di clock. Basta fare il reciproco della frequenza (40 Mhz), e moltiplicare per 1000 per avere i nanosecondi.
Quindi T = 1000/40 = 25 nsec.
Disegniamo il segnale di clock, sapendo che a tra due fronti di salita ci sono 25 nsec di differenza.



La linea ADDR è per gli indirizzi, parte fin dall'inizio in condizioni ideali e termina al fronte in cui i dati sono disponibili presso il master (metà di T3, cioè 62,5 nsec).
La linea DATA è per i dati, inizia al tempo di risposta (40 nsec) e termina al fronte successivo a quello in cui i dati sono disponibili presso il master (fine di T3, cioè 75 nsec).
Le linee MREQ e RD vanno di pari passo, e sono asserite a metà del primo ciclo e negate al fronte in cui i dati sono disponibili presso il master (sempre metà di T3, cioè 62,5 nsec).
La linea WAIT è asserita per tutti i cicli (escluso il primo) in cui i dati non sono ancora disponibili (sono stati effettivamente presi al fronte di salita di T3, cioè a 50 nsec). Attenzione: disponibili non vuol dire che sono disponibili presso il master, ma che sono pronti per essere acquisiti.

Esercizio 2
Con riferimento al funzionamento dei bus di un calcolatore:

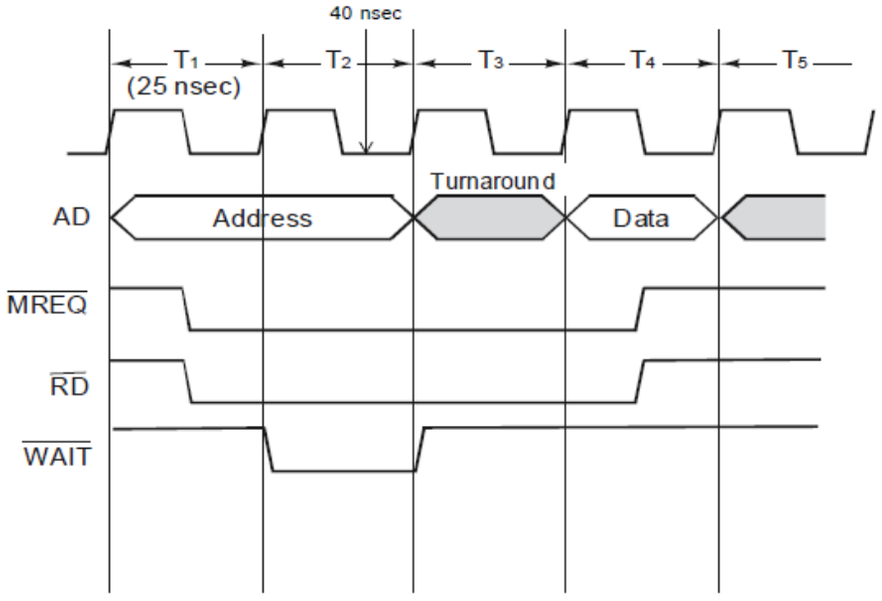
- **tracciare e illustrare il diagramma di temporizzazione di un bus <u>sincrono</u> a 40 Mhz <u>con linee condivise per dati e indirizzi</u> e segnali di MREQ, RD e WAIT, per una lettura da una memoria con un tempo di risposta di 40 nsec *dal momento in cui gli indirizzi sono disponibili***

- **tracciare e illustrare il diagramma di temporizzazione di un di bus <u>asincrono</u> con <u>linee separate per dati e indirizzi</u> e segnali di MREQ, RD, MSYN e SSYN, per una lettura da una memoria con un tempo di risposta di 50 nsec *dal momento in cui gli indirizzi sono disponibili.***

Si assuma in entrambi i casi di lavorare in condizioni ideali (nessun ritardo nell'asserimento dei segnali)

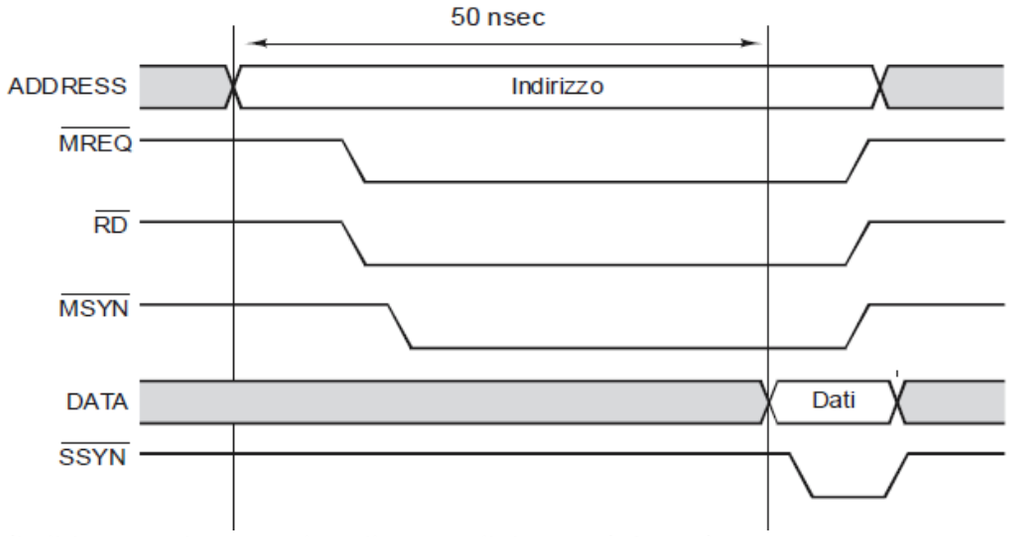
- T=1000/40 = 25 nsec.

Stavolta le linee di ADDR e DATA sono condivise. Quindi ci sarà una sola linea ADDR/DATA.



Address termina al fronte subito dopo il tempo di risposta (40 nsec).
Si attende un ciclo per il turnaround, cioè per permettere ai dati di sostituire l'indirizzo sulla linea.
MREQ e RD sempre asseriti a metà del primo ciclo.
WAIT asserito per il ciclo T2 (non si considera il ciclo del turnaround).
Dopo il turnaround, i dati sono presi all'inizio di T4 e disponibili al master a metà di T4.

• Nel caso di bus asincrono non c'è il segnale di clock. Per gestire i trasferimenti ci sono i segnali MSYN (master synchronization) e SSYN (slave synchronization).



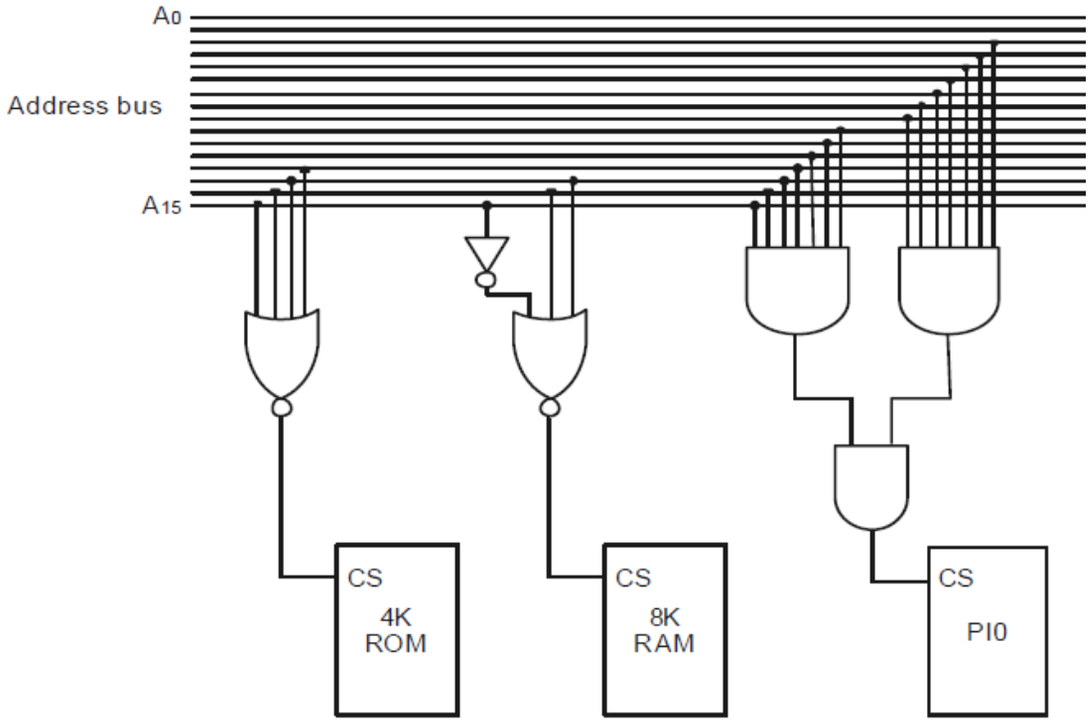
L'indirizzo termina poco dopo il tempo di risposta (50 nsec).
Poco dopo l'inizio dell'Indirizzo asseriamo MREQ e RD, che terminano insieme all'indirizzo.
MSYN inizia immediatamente dopo MREQ e RD e termina con loro, dopo che sono stati letti i dati.
SSYN inizia immediatamente dopo che i dati vengono presi, e termina insieme ai Dati.

Esercizio Spazi di indirizzamento
Si vuole realizzare un piccolo calcolatore embedded a 16 bit dotato di una ROM di 4KB, una RAM di 8KB e un dispositivo di I/O memory mapped a cui interfacciarsi tramite una PIO

- **Definire gli spazi di indirizzamento dei vari dispositivi a disposizione supponendo di poter utilizzare 16 bit per specificare gli indirizzi;**
- **Indicare come è possibile ottenere con porte logiche i corretti segnali di chip select a partire dalle linee del bus degli indirizzi;**
- **Indicare come è possibile semplificare i circuiti determinati al punto B con una decodifica parziale degli indirizzi.**

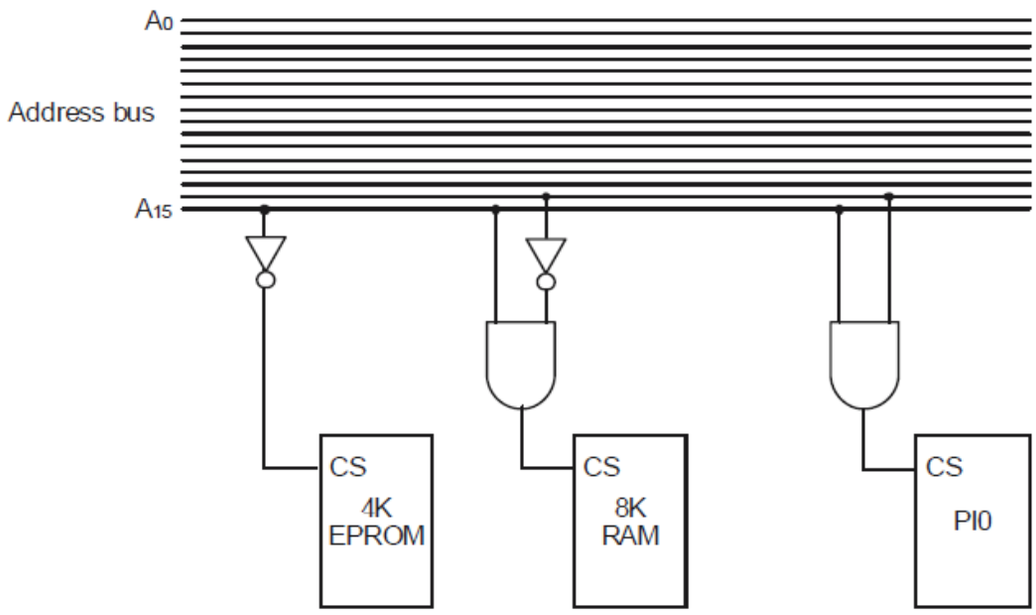
• Avendo degli indirizzi a 16 bit, bisogna assegnare gli spazi di indirizzamento alla ROM, alla RAM e alla PIO senza che si sovrappongano.
Iniziamo dalla ROM, che è di 4KB = 2^{12}. Vuol dire che servono 12 bit per indirizzare questo spazio.
Per definire gli spazi basta definire un indirizzo qualsiasi e sommarvi una sequenza di 16 bit con gli ultimi dodici bit uguali a 1 (0000 1111 1111 1111)
Per cui la ROM ha uno spazio che parte dall'indirizzo **0000 0000 0000 0000** e termina a **0000 1111 1111 1111** che viene dalla somma di 0000 0000 0000 0000 e 0000 1111 1111 1111.
Per la RAM di 8KB=2^{13}, ricordiamo che servono 13 bit. Bisogna stare attenti a non sovrapporre lo spazio di indirizzamento della RAM a quello della ROM.
Prendiamo un indirizzo non compreso nell'intervallo della ROM: per semplicità partiremo da **1000 0000 0000 0000** e termineremo a **1001 1111 1111 1111** che viene dalla somma di 1000 0000 0000 0000 e una sequenza di 16 bit con gli ultimi 13 bit uguali a 1 (0001 1111 1111 1111).
Ora definiamo lo spazio per la PIO. La PIO ha <u>sempre</u> 4=2^2 indirizzi (quindi aggiungiamo all' indirizzo scelto 0000 0000 0000 0011).
Per semplicità partiamo da **1111 1111 1111 1100** e quindi terminiamo a **1111 1111 1111 1111**.
• Dobbiamo disegnare tante linee quanti sono i bit di un indirizzo, in questo caso 16.
A questo punto dobbiamo collegare queste linee a delle porte logiche in modo che quando la sequenza di bit corrisponde a un indirizzo presente nell'intervallo di ROM, RAM o PIO, il segnale che entra nel chip sia uguale a 1.

Vediamo che lo spazio di indirizzamento della ROM, 0000 0000 0000 0000 – 0000 1111 1111 1111, è distinguibile per i primi 4 bit = 0000.
Lo spazio della RAM, 1000 0000 0000 0000 – 1001 1111 1111 1111 è distinguibile per i primi 3 bit = 100.
Mentre la PIO ha uno spazio, 1111 1111 1111 1100 – 1111 1111 1111 1111 distinguibile per i primi 14 bit = 1111 1111 11.
Nella decodifica totale basta collegare le linee che corrispondono ai bit della parte distinguibile a delle porte logiche: con i bit della parte distinguibile come input, queste porte logiche devono restituire 1.
Ad esempio, per la ROM che ha la parte distinguibile 0000 serve una porta che con questi input dia come output 1, e che entri nella ROM. Nell'immagine sotto viene usata una porta NOR, ma poteva benissimo essere usata al suo posto una porta AND (con le 4 linee di input negate).



Per la RAM viene usato una porta NOR in cui l'input 100 (con 1 negato dal NOT) dà 1, e per la ROM due porte AND (ne sono state usate due solo per evitare di fare una sola porta AND gigantesca) che restituiscono entrambe 1 se entra l'input 1111 1111 1111 11.

• Per la decodifica parziale si fa la stessa cosa di quella totale, ma invece della parte distinguibile si utilizzano i primi bit che bastano per distinguere lo spazio di indirizzamento da un altro.
Ad esempio, lo spazio della ROM è l'unico che inizia per 0, quindi basta collegare la linea del bit più significativo a una porta NOT che restituisce 1 al chip select della ROM.
Lo spazio della RAM è l'unico che inizia per 10, quindi basta collegare le due linee dei bit più significativi a una porta AND, con l'input della seconda linea negato da NOT.
Lo spazio della PIO è l'unico che inizia per 11, quindi basta una porta AND collegata alle due linee dei bit più significativi.


A0
Address bus
A15
CS
4K
EPROM
CS
8K
RAM
CS
PI0