

**PROVA SCRITTA DEL MODULO DI  
CALCOLATORI ELETTRONICI  
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA  
CORSO DI LAUREA IN INGEGNERIA ELETTRICA, ELETTRONICA ED INFORMATICA  
28 giugno 2017**

**MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI**

**Cognome e Nome:** \_\_\_\_\_ - **Matricola:** \_\_\_\_\_ - **Crediti:** \_\_\_\_\_

**ESERCIZIO 1 (7 punti)**

1. (4 punti) Si scriva la tabella di verità di un Full Adder e si rappresenti il relativo circuito impiegando il numero minimo di porte logiche.
2. (3 punti) Si definisca una rete logica realizzabile con N Full Adder, con relative caratteristiche e schema logico.

**ESERCIZIO 2 (10 punti)**

Si scriva il codice Assembler MIPS di una funzione `contaPariDispari` che, ricevendo in ingresso l'indirizzo iniziale di un vettore di interi senza segno `v`, la sua dimensione `N`, un secondo vettore di interi `w`, scriva nel vettore `w[0]` il numero degli elementi pari in `v` e in `w[1]` il numero degli elementi dispari. Si assuma che il contenuto di `w` sia inizializzato a 0. Nell'implementare questa funzione si supponga di utilizzare la funzione `pariDispari` che, ricevendo intero su `$4`, scrive su `$5` il valore 0 se l'intero è pari, 1 se è dispari.

Allocazione dei parametri di ingresso: `&v[0] → $4`, `N → $5`, `&w[0] → $6`.

Il codice MIPS potrebbe implementare il seguente codice C:

```
void contaPariDispari(int v[], int N, int w[])
{
    int i;

    for (i=0; i<N; i++)
        w[pariDispari(v[i])]++;
}
```

**ESERCIZIO 3 (10 punti)**

Si consideri una memoria primaria costituita da 32 parole e una memoria cache costituita da 8 parole, con blocchi da 2 parole. Si consideri il metodo di indirizzamento associativo su insiemi a 2 vie.

1. (6 punti) Indicare lo stato finale della cache, inizialmente vuota, nel caso il processore acceda in sequenza alle parole di indirizzo da 0 a 15 in questo ordine. Gli indirizzi dati sono espressi nel formato decimale.
2. (4 punti) Nell'ipotesi che il tempo di accesso alla memoria primaria sia 5 volte il tempo di accesso alla memoria cache  $T_c$ , calcolare la curva relativa al rapporto tempo medio di accesso alla gerarchia/tempo di accesso in cache ( $T/T_c$ ), in funzione dell'hit ratio di cache  $H$ , spiegando cosa accade per i casi  $H=0$  e  $H=1$  (motivare la risposta).

**ESERCIZIO 4 (6 punti)**

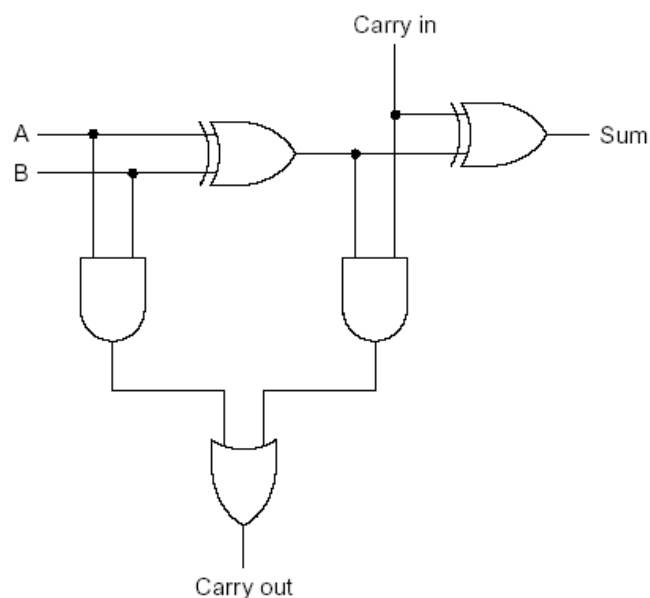
Si spieghi in modo dettagliato, ma chiaro e sintetico, cosa si intende per "periferica" e com'è strutturato un modulo di I/O.

## ESERCIZIO 1

### Soluzione.

1.

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



2.

Con N Full Adder è possibile realizzare un Parallel Adder. Se infatti si hanno due parole di N bit da sommare, i bit i-esimi di ciascuna parola sono gli ingressi del Full Adder i-esimo ( $i=0, \dots, N-1$ , col bit in posizione 0 nella funzione di bit meno significativo). Il riporto in uscita del Full Adder i-esimo va collegato al riporto in ingresso del Full Adder (i+1)-esimo. Il bit di riporto in ingresso del Parallel Adder corrisponde così al bit in ingresso del Full Adder relativo ai bit meno significativi. Lo schema progettuale è il seguente:

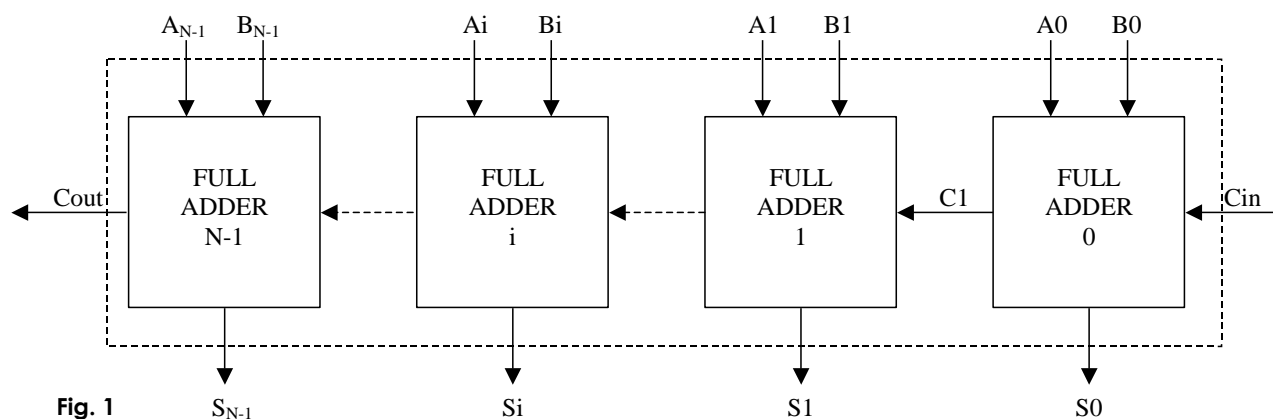


Fig. 1

Un'altra rete implementabile è il Carry Save-Adder, costituito da N Full Adder in parallelo:

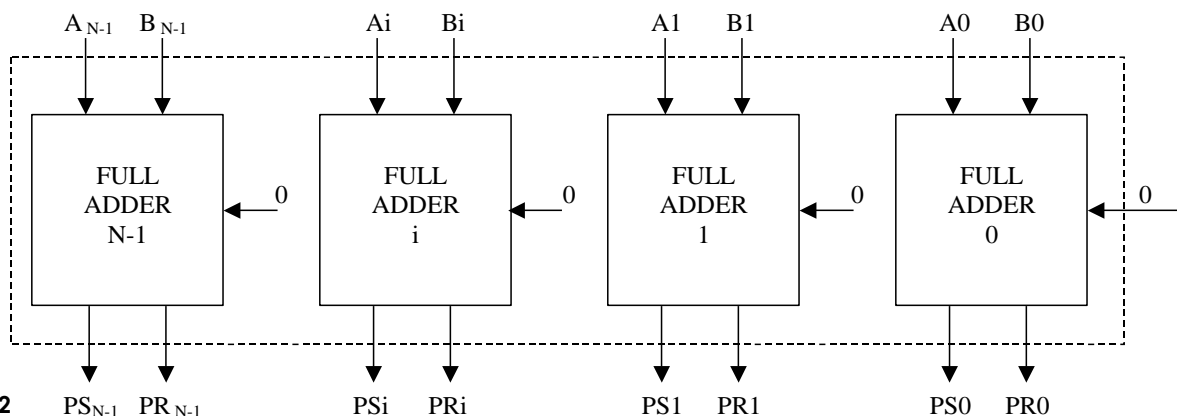


Fig. 2

PS e PR non sono somma e riporto dei due bit, bensì soltanto Pseudo-Somma e Pseudo-Riporto. Per ottenere la somma complessiva, bisogna moltiplicare per 2 lo Pseudo-Riporto (shift) e poi effettuare la somma  $PS + 2PR$  con un Parallel Adder.

## ESERCIZIO 2

### Soluzione.

$\$8 \leftarrow i;$

$\$12 \leftarrow \text{copia di } \&v[0]; \$13 \leftarrow \text{copia di } N$

```

occorrenze:  addi $29, $29, -16      #salvataggio contesto
              sw $8, 0($29)
              sw $12, 4($29)
              sw $13, 8($29)
              sw $31, 12($29)
              move $12, $4
              move $13, $5
              move $8, $0            #i=0
for:         beq $8, $13, exit
              muli $4, $8, 4
              add $4, $4, $12
              lw $4, 0($4)           #carico v[i] e lo passo a $4
              jal pariDispari
              muli $4, $5, 4          #in $5 c'è 1 se v[i] è dispari
              add $4, $4, $6
              lw $5, 0($4)
              addi $5, $5, 1
              sw $5, 0($4)
              addi $8, $8, 1
              j for
exit:        move $4, $12
              move $5, $13
              lw $8, 0($29)           #ripristino contesto
              lw $12, 4($29)
              lw $13, 8($29)
              lw $31, 12($29)
              addi $29, $29, 16
              jr $31                 #ritorno a chiamante

```

### **ESERCIZIO 3 (8 punti)**

#### **Soluzione.**

1.

Metodo set-associativo: < TAG 3 bit > < Index 1 bit > < Offset 1 bit >

Stato finale della cache nel caso in esame:

Insieme 0 → Linea 0 : Parole 8-9 – Linea 1 : Parole 12-13

Insieme 1 → Linea 0 : Parole 10-11 – Linea 1 : Parole 14-15

2.

Il tempo medio di accesso alla gerarchia cache-primaria è pari a:

$$T = T_c + (1 - H) * T_p$$

Dove  $T_p = 5T_c$ . Per cui:

$$T = T_c + (1 - H) * 5 * T_c = (6 - 5H) * T_c$$

Quindi :

$$T/T_c = 6 - 5H$$

Si osservi, come da aspettativa, che per  $H=0$ , il tempo medio di accesso diventa sempre sei volte quello di accesso in cache (in quanto si sommano sempre il tempo di accesso alla primaria e quello alla cache); per  $H=1$ , invece, il rapporto è 1, in quanto il tempo medio di accesso si riduce a quello di accesso alla cache.

### **ESERCIZIO 4**

#### **Soluzione.**

Vedi dispense del corso (capitolo modulo I/O).