Esempi e puntualizzazioni sull'uso del costrutto `function`

Si consideri una rete combinatoria, che ricevendo in ingresso una codifica ascii su 8 bit fornisca in uscita 1 se la codifica ascii è di una cifra decimale, 0 altrimenti.

Utilizziamo il costrutto `function` per descrivere la legge che la rete implementa, come illustrato di seguito:

```
function esempio;
 input [7:0] ascii;
  descrizione della legge
endfunction
```

Possibili formulazioni del costrutto `function` sono:

```
function esempio;
 input [7:0] ascii;
  casex(ascii)
   'B00110???: esempio=1;
   'B00111100?: esempio=1;
     default : esempio=0;
  endcase
endfunction
```

o anche

```
function esempio;
 input [7:0] ascii;
  esempio=(ascii>'H2F)&(ascii<'H3A)?1:0;
endfunction
```

o anche

```
function esempio;
 input [7:0] ascii;
  esempio=((ascii[7:4]=='B0011)&(ascii[3:0]<'B1010))?1:0;
endfunction
```

o anche, usando una espressione algebrica

```
function esempio;
 input [7:0] ascii;
  esempio=~ascii[7]&~ascii[6]&ascii[5]&ascii[4]&(~ascii[3]|(~ascii[2]&~ascii[1]));
endfunction
```

Poiché il costrutto **`function`** descrive una legge **equivalente ad una tabella di verità** è **ASSOLUTAMENTE SBAGLIATO** introdurre al suo interno variabili di appoggio tramite il costrutto **`wire`**.

E' pertanto errata la formulazione

```
function esempio;
 input [7:0] ascii;
 wire un_primo_pezzetto, un_secondo_pezzetto;
   un_primo_pezzetto  =~ascii[7]&~ascii[6]&ascii[5]&ascii[4];
   un_secondo_pezzetto=~ascii[3]|(~ascii[2]&~ascii[1]);
   esempio=un_primo_pezzetto & un_secondo_pezzetto;
endfunction
```

Poiché il costrutto `function` descrive la legge caratterizzante una rete combinatoria è ASSURDO dichiarare oggetti di tipo `reg` all'interno del costrutto.