S6. Vectori de instanțe și blocuri generate

Construirea instanțelor multiple în Verilog

1. Scopul laboratorului

Construirea vectorilor de instante si configurarea blocurilor generate

2. Desfășurarea laboratorului

2.1 Vectori de instanțe și structura generate

Vectorii de instanțe permit scrierea rapidă a mai multor instanțe ale aceluiași modul.

Blocurile generate oferă o variantă flexibilă de creare a unui număr mare de instanțe, cu interconexiuni complexe.

2.2.1 Vectorii de instanțe

Facilitează crearea rapidă a mai multor instanțe ale aceluiași modul, atunci când toate instanțele sunt conectate la aceleași semnale. Utilizarea lor pentru proiecte cu interconexiuni complexe poate deveni dificilă.

Implementarea Verilog de pe pagina următoare construiește un convertor BCD8421 la E3 pentru numere cu ***k*** cifre, folosind ***k*** instanțe ale unui sumator pe 4 biți, numit ***adder_4b***.

În general, formatul vectorilor de instanțe adoptă următoarea regulă:

```
module-name instance_name [top-index: bottom index]

        ( .p(s), ... )
```

Codul Verilog pentru convertorul BCD8421 la E3 este prezentat mai jos:

```
module bcde3_conv #(
parameter k = 4 ; // numarul de cifre
)(
 input  [4*k-1:0] bcd, // numarul de intrare bcd
 output [4*k-1:0] e3 // numarul de iesire e3
);
// Aplicam regula enuntata pentru vectori de instante si avem:
   adder_4b cnv[k-1:0] ( // sunt construite k instante adder_4b
        .x(bcd),
        .y(4'd3),
        .z(e3)
);
endmodule
```

În ceea ce privește formatul general al vectorilor de instanțe, dacă lățimea semnalului ***s*** este egală cu numărul de instanțe înmulțit cu cu lățimea portului p , atunci s este partiționat egal în numărul de biți ai portului p, fiecare partiție fiind legată la una din instanțele create. Pentru exemplul ***bcde3_conv***, intrarea ***bcd*** va fi partiționată în grupe de 4 biți, fiecare grup fiind legat la una din cele ***k*** instanțe (similar pentru ieșirea ***e3***).

Dacă lățimea semnalului ***s*** este egală cu lățimea portului ***p***, atunci întreg semnalul s este legat la toate instanțele. Pentru exemplul ***bcde3_conv*** , valoarea 3, reprezentată pe 4 biți, ce se va aduna la fiecare cifră BCD8421, are aceeași lățime cu portul ***y*** al sumatorului și, deci, va fi conectată la toate cele ***k*** instanțe.

2.2.2 Blocuri generate

Bucla for din interiorul *blocului* generate:

- Folosește o variabilă de tip genvar ca index
- Are conținutul cuprins într-un bloc ***begin ... end*** cu *nume*

Convertorul anterior din BCD8421 la E3 este rescris ca mai jos:

```
module bcde3_conv #(
parameter k = 4 ; // numarul de cifre
)(
 input  [4*k-1:0] bcd, // numarul de intrare bcd
 output [4*k-1:0] e3 // numarul de iesire e3
);
  generate
       genvar i;
              for (i = 0 ; i < k ; i = i + 1) begin: vect
               adder_4b uconv ( .x(bcd[i*4+3:i*4]),
                                .y(4'd3),
                                .z(e3[i*4+3:i*4])
);
              end
  endgenerate
endmodule
```

Blocul ***begin*** ... ***end*** începe cu linia unde a fost definit ciclul ***for***. Cuvântul cheie begin este urmat de un identificator (în acest caz vect, dar se poate folosi orice identificator Verilog valid).

În blocul cu nume este construită o singură instanță, numită uconv, în fiecare iterație.

Asocierea porturilor utilizează indexul ***i*** pentru gruparea a 4 biți consecutivi din intrarea ***bcd*** și a 4 biți din ieșirea ***e3***. Grupele de 4 biți sunt realizate folosind expresia ***part-select*** [*i* * 4+3 : *i* * 4].

La portul ***y*** al sumatoarelor este conectată valoarea 3, reprezentată pe 4 biți.

2.2 Problemă rezvolată

Unitatea de preprocesare a intrării a unei aplicații criptografice

Exercițiu: Să se construiască calea de date pentru unitatea de preprocesare a intrării a unui design Secure Hash Algorithm 2 (SHA-2) pe 256 de biți.

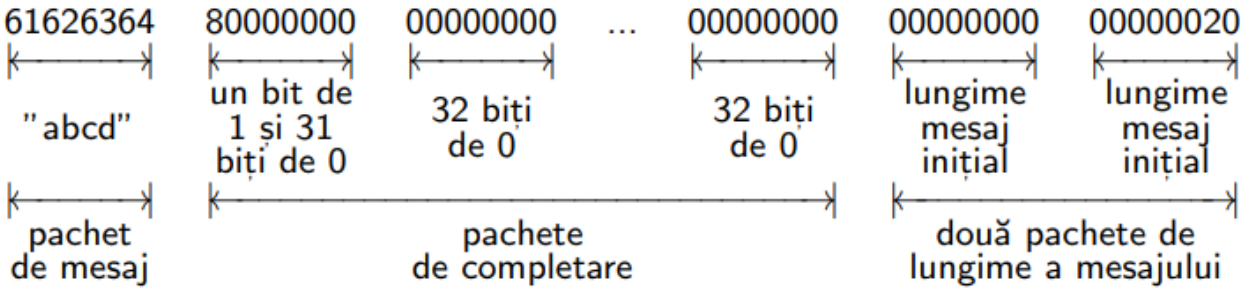
Soluţie: Unitatea de preprocesare a intrării SHA-2 pe 256 de biţi implementează faza de preprocesare, descrisă în [FIPS15] (secţiunea 5.1.1) constând în completarea şi împărţirea mesajului.

Completarea mesajului: Pentru un mesaj iniţial de lungime ***l***, completarea adaugă un bit de 1 şi ***k*** biţi de 0, astfel încât ***l***+1+***k***=448 (mod 512). La final este ataşată lungimea mesajului iniţial ca număr fără semn pe 64 de biţi.

Împărţirea mesajului: Mesajul iniţial cu biţii completaţi mai sus este divizat în blocuri de 512 biţi, care sunt livraţi la ieşirea unităţii. Pentru concizie, unitatea recepţionează mesajul iniţial în pachete de 32 de biţi iar mesajul iniţial are o lungime ***l***, multiplu de 32.

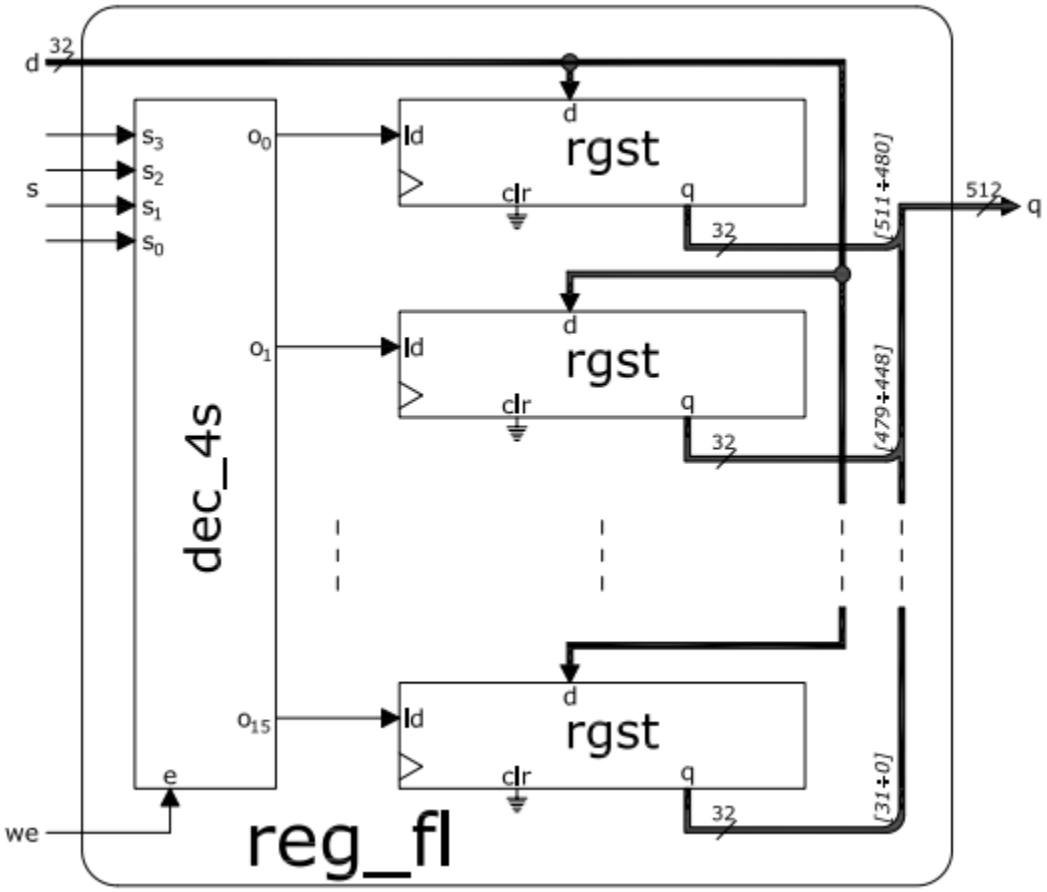
Se consideră mesajul “abcd”, reprezentat în cod ASCII pe 8 biţi, de procesat. Lungimea mesajului este ***l*** = 32 biţi şi acesta va fi completat cu 1 bit de 1 urmat de k = 415 biţi de 0 (k este obţinut din ecuaţia *l* + 1 + k = 448 (mod 512)). Ulterior se ataşează lungimea iniţială, în biţi, a mesajului, reprezentată ca număr fără semn pe 64 de biţi.

Blocul de 512 biţi generat de modulul de intrare pentru acest exemplu este detaliat mai jos (câmpurile sunt reprezentate în hexazecimal):



Unitatea de prelucrare operează cu 5 tipuri de pachete:

- ***mesaj***: pachetele în care este divizat mesajul inițial şi care sunt livrate la intrarea unității, se pot primi unul sau mai multe astfel de pachete
- ***padding***: pachet cu cel mai mai semnificativ bit de 1 şi 31 de biţi de 0, grupează primii 32 de biţi ataşaţi în faza de completare, un singur pachet de acest tip se generează
- ***zero***: pachet cu 32 de biţi de 0, grupează biţii succesivi, ataşaţi în faza de completare, se pot genera 0, 1 sau mai multe astfel de pachete
- ***prima jumătate lungime mesaj***: pachet conţinând cei mai semnificativi 32 de biţi ai lungimii iniţiale a mesajului, un singur pachet de acest tip se generează
- ***a doua jumătate lungime mesaj***: pachet conţinând cei mai puţin semnificativi 32 de biţi ai lungimii iniţiale a mesajului, un singur pachet de acest tip se generează.



Unitatea de prelucrare va diviza întreaga secvenţă binară (mesaj iniţial + biţii de completare + lungimea mesajului iniţial) în blocuri de 512 biţi. În fiecare ciclu de ceas unitatea de prelucrare a intrării primeşte un nou pachet din mesajul iniţial iar după primirea a 16 pachete consecutive un nou bloc de 512 biţi va fi livrat la ieşire.

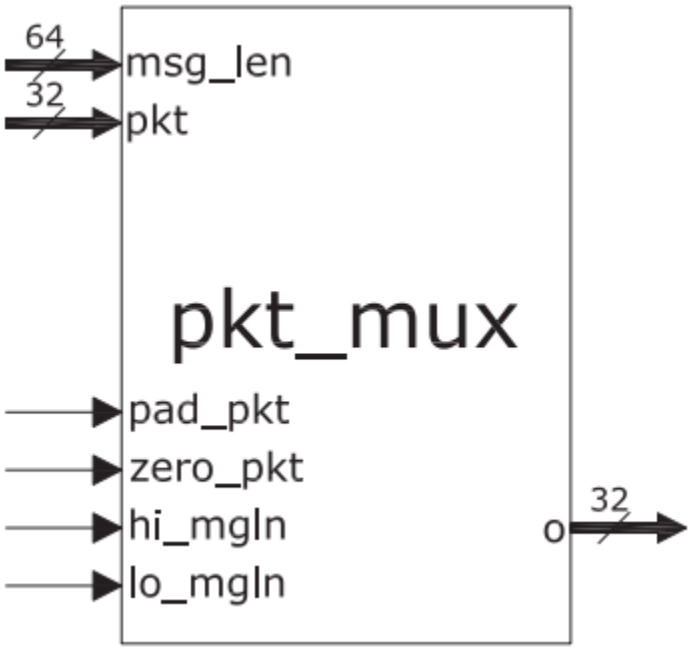
Pentru stocarea a 16 pachete de 32 de biţi se foloseşte un register file dedicat, ***reg_fl***, conţinutul său concatenat formând blocul. După asamblarea unui bloc acesta este livrat la ieşire şi este

reîncepută primirea următoarelor 16 pachete de mesaj. Când întreg mesajul a fost recepţionat, se generează celelalte 4 tipuri de pachete, stocate şi ele în ***reg_fl***, asamblate în blocuri de 512 biţi.

Un numărător pe 4 biţi păstrează evidenţa adresei de scriere în ***reg_fl***, pentru stocarea următorului pachet. Pentru contorizarea lungimii mesajului iniţial este folosit un registru pe 64 de biţi ce va fi incrementat de un sumator la fiecare pachet de mesaj recepţionat.

Register file-ul primeşte pachete la intrarea d, adresa de scriere este furnizată la intrarea ***s*** iar ***we*** activează scrierea. La ieşirea ***q*** este concatenat conţinutul tuturor registrelor interne, registrul de la adresa 0 având conţinutul cel mai semnificativ.

Modulul dec_4s din figura Register file-ului este o parametrizare a modulului disponibil **aici.**



Multiplexorul dedicat pkt_mux furnizează următorul pachet register file-ului. Intrările sale de date sunt ***pkt***, pentru pachetele mesajului şi ***msg_len***, pentru lungimea mesajului iniţial.

Ieşirea ***o*** este controlată de următoarele linii de selecţie:

- ***pad_pkt*** – furnizează un pachet padding
- ***zero_pkt*** – furnizează un pachet yero
- ***hi_msgln*** – furnizează jumătatea mai semnificativă a lungimii mesajului (primită la intrarea msg_len)
- ***lo_msgln*** – furnizează jumătatea mai puţin semnificativă a lungimii mesajului

Nu pot fi simultan active două sau mai multe intrări de selecţie. Dacă niciuna din cele 4 intrări de selecţie nu este activă, la ieşire se furnizează pachetul de mesaj curent, primit la intrarea ***pkt***.

3. Referințe bibliografice

[AMI]** Advanced Module Instantiation. [Online]. Available: http://www.eecs.umich.edu/courses/eecs470/OLD/w14/labs/lab6_ex/AMI.pdf

[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: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf