AC_TEMA1: Transform the number - README

Mărgăritescu Vlad - 333AC

In rezolvarea temei 1 am urmarit indicatiile din cerinta si am implementat cele 2 module.

Modulul **div_algo**:

Am inceput mai intai cu modulul div_algo folosit pentru a calcula catul si restul a doua numere. Astfel, am consultat pagina Wikipedia pusa in resurse. Dupa ce m-am uitat pe toti algoritmii prezentati acolo mi-am dat seama ca am de ales dintre 3 variante care lucreaza in binar: cel de la Long Division si cele 2 de la Slow division methods.

Dar dupa ce le-am analizat in detaliu pe toate 3 am observat ca cele 2 de la Slow division (Restoring division si Non-restoring division) au ca necesitate ca R si D sa aiba numar dublu de biti fata de N si Q. Cum modulul din tema are toate valorile pe 16 biti singura optiune a ramas „Integer division (unsigned) with remainder", care nu avea nimic de genul acesta comentat.

Am stat putin sa inteleg exact cum functioneaza acel algoritm, am incercat cateva exemple pe foaie, iar apoi am incercat sa il implementez in Verilog. Am intampinat o problema la loop-ul for, deoarece nu accepta „>=0" asa cum este in algoritm (primeam failed to synthesize in checker). Am probat apoi si cu „>-1", dar tot nu era ok (rula, dar pe checker 99% din rezultate erau gresite). Dupa multe incercari esuate de a schimba sintaxa, mi-a venit ideea sa las acel for cu i de la 15 la 1, iar pentru cazul in care i este 0 sa fac un if special. Am testat din nou cu checker-ul offline si am primit ok pe toate testele, astfel rezolvand acest modul. Mai mult, trebuie precizat ca algoritmul este inclus intr-un bloc always combinational, deoarece se executa repetitiv.

Modulul **base2_to_base3**:

Acesta este modulul principal al temei. Pentru inceput am declarat iesirile Q,R de tip wire pentru a le putea folosi in implementare. Mai departe am declarat de tip reg starile state si next_state ale FSM-ului, precum si width care se ocupa cu memorarea resturilor pe pozitiile din numarul in baza 3. In plus, am avut nevoie si de o variabila auxiliara pe care an numit-o „noul_deimpartit" si care va lua mereu valoarea catului dupa executarea lui div_algo.

Mai departe am folosit schema automatului din cerinta temei si am declarat cele 4 stari ale acestuia. Astfel, folosind si notiunile din laboratoare, am realizat partea secventiala, iar apoi pe cea combinationala in care au loc efectiv tranzitiile de la o stare la alta. Mai mult, aici am instantiat modulul div_algo cu iesirile Q, R, dar in loc de deimpartitul N avem numarul in baza 2, dar acesta trebuie mereu actualizat dupa fiecare rulare a lui div_algo, tocmai de aici necesitatea de a avea si variabila „noul_deimpartit" pt numarul in baza 2. Evident, in loc de D este valoarea 3, doarece algoritmul mereu v-a imparti numarul la 3.

In blocul always combinational este FSM-ul propriuzis.

Starea 0 (READ): semnalul done_reg este initializat cu 0, la fel si numarul in baza 3 si width. Daca en==1 numarul in baza 2 este citit si se merge mai departe.

Starea 1 (EXEC): In aceasta stare valoarea „noul_deimpartit" ia valoarea catului rezultat din apelarea lui div_algo. Apoi se merge mai departe.

Starea 2 (EXEC2): Aici se formeaza efectiv numarul in baza 3. Restul poate fi 0,1 sau 2, astfel este reprezentat pe 2 biti. Variabila width este cea care ia bitii din base3_no_r doi cate doi si retine pe acestia valoarea celor 2 biti din rest. Acesta e si motivul pt care numarul in baza 2 are 16 biti, iar cel in baza 3 e pe 32. Dupa aceea numarul numarul in baza 2 ia valoarea catului obtinut dupa rularea div_algo. Daca acest cat este 0, rezulta ca am terminat de impartit numarul si se merge in starea urmatoare. Altfel, acest cat este trimis din nou in starea EXEC, unde noul_deimpartit primeste noua valoare si div_algo se executa iar. Evident, restul se reseteaza si valoarea lui width creste cu 2 pt a putea fi stocate in base3_no_r urmatorii 2 biti corespunzatori noului rest. Si tot asa pana cand base2_no_r va ajunge la un moment dat sa fie nul.

Starea 3 (DONE): Starea finala. Daca s-a ajuns aici inseamna ca numarul dat a fost impartit complet si nu se mai poate aplica div_algo. Astfel, conversia se incheie si valoarea done devine 1. Mai departe se revine in starea READ si se citeste urmatorul numar. Asadar, algoritmul se repeta pana cand nu mai are ce sa testeze.