POKAZNA VEŽBA 9
Priprema za ispit

ZADACI

1. Automat – sistem za simulaciju igre

Pred vama je apstraktna igra čija pravila nisu u potpunosti poznata. Ono što je poznato jeste da igrači A i B odigravaju poteze koji im donose po 1 poen. Ukoliko neki od igrača u bilo kom trenutku ostvari prednost od 2 poena, igra se završava. Kada igrač gubi sa 1 poenom razlike dobija mogućnost specijalnog poteza koji ukoliko je uspešan donosi 2 poena i samim tim ga stavlja u prednost u odnosu na drugog igrača. Dok je rezultat nerešen, igrači ne mogu da koriste specijalne poteze.

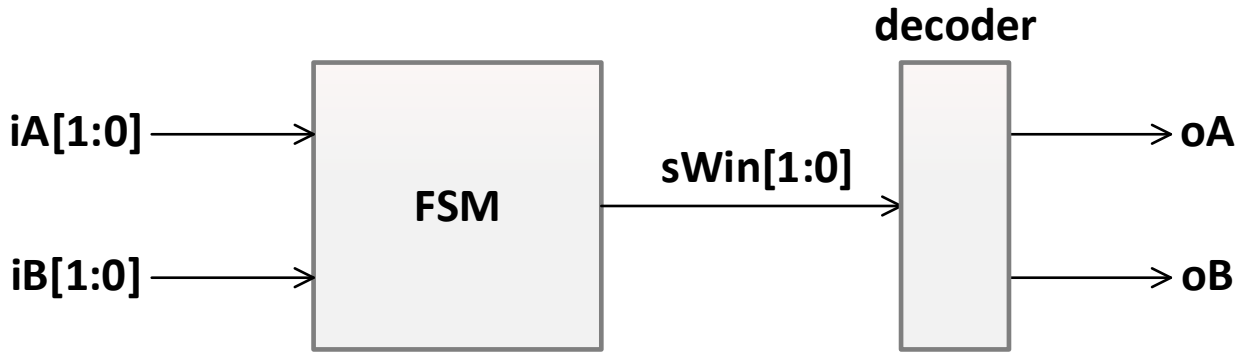
Vaš zadatak je da u VHDL-u projektujete sistem za simulaciju rezultata ove igre, prikazan na Slici 1, koji je baziran na automatu čija funkcija prelaza je data na Slici 2.

Ulazi u sistem:

- **iCLK** : signal takta
- **iRST** : signal asinhronog reseta, aktivan na visokom nivou
- **iA** : dvobitni ulaz čija je vrednost “01” kada igrač A odigra običan potez ili “10” kada odigra specijalni potez
- **iB** : dvobitni ulaz čija je vrednost “01” kada igrač B odigra običan potez ili “10” kada odigra specijalni potez

Izlazi iz sistema:

- **oA** : aktivan ukoliko je igrač A pobedio
- **oB** : aktivan ukoliko je igrač B pobedio

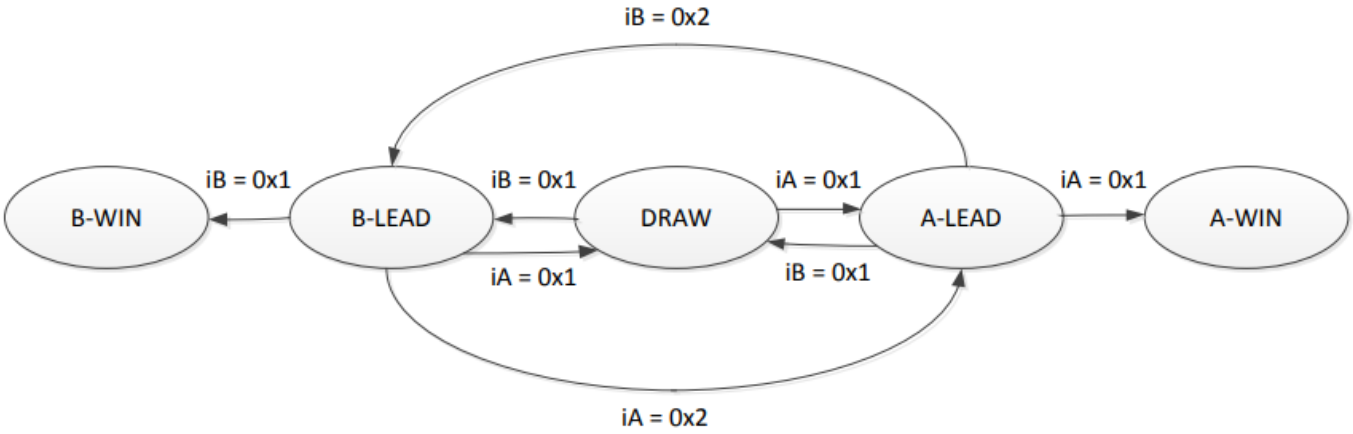


Slika 1 – Blok šema sistema
(zbog jednostavnijeg prikaza, ulazi iCLK i iRST su izostavljeni ali su podrazumevani kod svih sekvencijalnih komponenti)

Izlaz iz automata je dvobitni signal **sWIN** koji ima sledeće vrednosti:

- “01” kada igrač A pobedi
- “10” kada igrač B pobedi
- “00” dok igra traje

Početno stanje automata je **DRAW**, a konačna stanja su **A-WIN** i **B-WIN**.



Slika 2 – Graf prelaza stanja
(0x1 predstavlja heksadecimalno 1, kao što 0x2 predstavlja heksadecimalno 2 ali su vrednosti dvobitne)

Simulirati sistem na sledeći način:

- U prvoj partiji neka pobedi igrač A bez da je iko iskoristio specijalni potez i da su oba igrača barem jednom postigla poen
- U drugoj partiji neka pobedi igrač B ali tako da su oba igrača iskoristila specijalni potez barem jednom
- Između partija resetovati sistem na 20 perioda takta

2. Računanje vrednosti izraza

Koristeći LPRS1 asembler zajedno sa procesorom koji je konstruisan tokom semestra realizovati asemblerski program koji **izračunava vrednost izraza**:

$$\frac{2^N + \sum_{i=0}^{N-1} array[i]}{4}$$

Elementi niza se nalaze u RAM memoriji (***data_ram.vhd***). Program računa **N-ti stepen broja 2** i sabira ga sa **sumom svih elemenata niza**, a zatim rezultat **deli sa 4**.

- **N** je broj elemenata koji su smešteni u memoriji za podatke počevši od nulte lokacije (0x00).
- ***array*** je samo oznaka za niz elemenata u memoriji za podatake.
- U asemblerskom kodu (u .data sekciji) zadati su elementi niza: -1, -2, -3, -4, 6.
- Rezultat izraza smestiti u registar **R0**, kao i na memorijsku lokaciju za rezultat (***p_result***) koji se čuva u memoriji za podatke na adresi 0x1A.
- Kada se program završi, potrebno je upisati **vrednost 1** na memorijsku lokaciju za kraj (***p_done***) koji se čuva u memoriji za podatke na adresi 0x20.
- Testirati napisan asemblerski program u Quartus i ModelSim alatu.

Pseudo kod algoritma, kao i podaci se već nalaze u datoteci **kod.asm.txt** i prikazani su u nastavku:

```
/*
// .data
short* p_result   = 0x1A;
short* p_done     = 0x20;
short N = 5;
short a[5] = {-1, -2, -3, -4, 6};
*/

.data
0x1A
0x20
5
-1, -2, -3, -4, 6

/*
// .text
short res = 0;
short pow = 0;

    pow = 2^N;
    for (short i = 0; i<N; i++)
    {
            res += a[i];
    }
    res += pow;
    res /= 4;
    *p_result = res;
    *p_done = 1;
*/
```

U nastavku sledi tabela sa podržanim instrukcijama za LPRS1 procesor:

Operacija	C funkcionalnost	Kod instrukcije	Komentar
mov Rz, Rx	Rz = Rx	00 0000	Prebacuje sadržaj registra X u registar Z
add Rz, Rx, Ry	Rz = Rx + Ry	00 0001	Zbir registara X i Y upisuje u odredišni registar Z
sub Rz, Rx, Ry	Rz = Rx – Ry	00 0010	Razliku registara X i Y upisuje u odredišni registar Z
and Rz, Rx, Ry	Rz = Rx & Ry	00 0011	Rezultat logičkog i registara X i Y upisuje u registar Z
or Rz, Rx, Ry	Rz = Rx | Ry	00 0100	Rezultat logičkog ili registara X i Y upisuje u registar Z
not Rz, Rx	Rz = ~ Rx	00 0101	Negiranu vrednost registra X upisuje u odredišni registar Z
inc Rz, Rx	Rz = Rx + 1	00 0110	Inkrementira registar X i upisuje u odredišni registar Z
dec Rz, Rx	Rz = Rx – 1	00 0111	Dekrementira registar X i upisuje u odredišni registar Z
shl Rz, Rx	Rz = (unsigned short) Rx << 1	00 1000	Logički pomeren registar X ulevo upisuje u odredišni registar Z
shr Rz, Rx	Rz = (unsigned short) Rx >> 1	00 1001	Logički pomeren registar X udesno upisuje u registar Z
ashl Rz, Rx	Rz = (signed short) Rx << 1	00 1010	Aritmetički pomeren registar X ulevo upisuje u odredišni registar Z
ashr Rz, Rx	Rz = (signed short) Rx >> 1	00 1011	Aritmetički pomeren registar X udesno upisuje u odredišni registar Z
ld Rz,Ry	Rz = *Ry	10 0000	Upisuje u Rz sadržaj koji se nalazi u memoriji na adresi u Ry
st Rx, Ry	*Ry = Rx	11 0000	Upisuje u memoriju na adresu koja piše u Ry sadržaj iz Rx
jmp ADDR	jmp	01 0000	Bezuslovni skok na labelu ADDR
jmpz ADDR	if Z = 1 jmp	01 0001	Uslovni skok na labelu ADDR ukoliko je rezultat 0
jmps ADDR	if S = 1 jmp	01 0010	Uslovni skok na labelu ADDR ukoliko je rezultat negativan
jmpc ADDR	if C = 1 jmp	01 0011	Uslovni skok na labelu ADDR ukoliko je došlo do prenosa
jmpnz ADDR	if Z != 1 jmp	01 0101	Uslovni skok na labelu ADDR ukoliko rezultat nije 0
jmpns ADDR	if S != 1 jmp	01 0110	Uslovni skok na labelu ADDR ukoliko rezultat nije negativan
jmpnc ADDR	if C != 1 jmp	01 0111	Uslovni skok na labelu ADDR ukoliko nije došlo do prenosa