1. Deklarisati registarsku promjenljivu po imenu *oscilator*. Inicijalizovati je na nulu i napraviti da svakih 30 vremenskih jedinica mijenja svoju vrijednost (sa 0 na 1 i obratno). Ne koristiti *always* blok.

2. Realizovati taktni signal sa periodom od 40ns i sa odnosom trajanja visoke i niske vrijednosti 1:4. U početnom trenutku vrijednost taktnog signala treba da bude 0.

3. Dat je *initial* blok sa blokirajućim proceduralnim dodjeljivanjem:

```
initial
        begin
                a = 1'b0;
                b = #10 1'b1;
                c = #5 1'b0;
                d = #20 {a, b, c};
        end
```

U kojim vremenskim trenucima će svaki od iskaza biti izvršen? Kolike su trenutne, a koje su konačne vrijednosti promjenljivih a, b, c i d?

4. Uraditi prethodni zadatak sa neblokirajućim dodjeljivanjima.

5. Koji je redoslijed izvršavanja iskaza u sljedećem Verilog kodu:

```
initial
        begin
                a = 1'b0;
                #0 c = b;
        end
initial
        begin
                b = 1'b1;
                #0 d = a;
        end
```

Ima li kakvih nejasnoća oko redoslijeda izvršavanja? Koje su krajnje vrijednosti za a, b, c i d?

6. Koja je krajnja vrijednost promjenljive d u sljedećem primjeru:

```
initial
        begin
                b = 1'b1; c = 1'b0;
                #10 b = 1'b0;
        end
initial
        begin
                d = #25 (b | c);
        end
```

Da li je u ovom primjeru došlo do *race condition*?

7. Realizovati D flip flop koji reaguje na silaznu ivicu *clock* signala, sa sinhronim resetom aktivnim na visokom logičkom nivou (flip flop se resetuje samo na silaznoj ivici *clock*-a kad je *reset* visok). Koristiti samo behavioral iskaze. Napraviti *stimulus* kojim će se testirati rad ovog flip flopa. Signal *clock* ima periodu od 10ns.

8. Realizovati D flip flop iz prethodnog zadatka sa asinhronim resetom (resetuje se kad god je *reset* signal visok – ne čeka silaznu ivicu takta).

9. Koristeći **wait** iskaz realizovati D *latch* koji je aktivan na visokom logičkom nivou upravljačkog signala: *q=d*, kad god je *clock*=1.

10. Koristeći **case** direktivu, realizovati ALU koja ima 8 funkcija (prikazanih u tabeli) nad 4-bitnim ulazima A i B. Izbor funkcije se vrši 3-bitnim selekcionim signalom SEL. Izlaz ALU-a je 5-bitni signal OUT. Ignorisati eventualnu pojavu prekoračenja prilikom računanja.

SEL	Funkcija
3'b000	OUT = A
3'b001	OUT = A + B
3'b010	OUT = A – B
3'b011	OUT = A / B
3'b100	OUT = A % B
3'b101	OUT = A << 1
3'b110	OUT = A >> 1
3'b111	OUT = (A > B)

11. Koristeći **while** petlju realizovati generator takta. Inicijalna vrijednost je 0, a perioda 10ns.

12. Koristeći **for** petlju postaviti na 0 lokacije 0 do 1023 4-bitnog registarskog polja *cache_mem*.

13. Koristeći **forever** komandu, realizovati taktni signal sa periodom od 10ns i sa odnosom visoke i niske vrijednosti 40:60. U početnom trenutku vrijednost takta treba da bude 0.

14. Koristeći **repeat** petlju, odložiti iskaz *a=a*+1 za 20 pozitivnih ivica signala *takt*.

15. Dat je dio koda sa ugniježdenim sekvencijalnim i paralelnim blokom. Odrediti vremenske trenutke u kojima se izvršava svaki od iskaza:

```
initial
begin
        x = 1'b0;
        #5 y = 1'b1;
        fork
                #20 a = x;
                #15 b = y;
        join
        #40 x = 1'b1;
        fork
                #10 p = x;
                begin
                        #10 a = y;
                        #30 b = 0;
                end
                #5 m = y;
        join
end
```