

OBJEKTNO ORIJENTISANO PROGRAMIRANJE

- domaći zadatak broj 2 -

Funkcionalna specifikacija

Na programskom jeziku C++ implementirati simulator elektronske digitalne mreže (*Simulator*).

Učitavanje digitalne mreže u simulator

Implementirati operaciju `void Simulator::loadCircuit(const string& filepath);` koja u simulator učitava informacije o digitalnoj mreži koje se nalaze u ulaznom fajlu. Putanja do fajla je zadata parametrom `filepath`. Ukoliko je prethodno već učitana digitalna mreža, sve informacije o staroj mreži se brišu i učitava se nova mreža. Format ulaznog fajla je dat u nastavku.

- Prvi red fajla sadrži informaciju o vremenu trajanja simulacije (μ s).
- Drugi red fajla sadrži informaciju o broju elemenata u mreži.
- Narednih nekoliko redova sadrži opis svakog od elemenata (po jedan opis elementa u jednom redu). Opis elementa započinje identifikatorom elementa (ceo broj) nakon čega se navodi tip elementa (ceo broj) i dodatni parametri koji zavise od tipa elementa.
- Ostatak fajla sadrži opise veza između elemenata (po jedan opis veze u jednom redu). Opis veze započinje identifikatorima elemenata koji su povezani, pri čemu se izlaz prvog navedenog elementa veže za jedan od ulaza drugog navedenog elementa. Ulaz koji učestvuje u vezi se navodi nakon identifikatora drugog elementa (ulazi elementa sa N ulaza su numerisani 0 .. N-1).

<i>Element</i>	Izvor signala takta	Ručno podesiv izvor signala	NE kolo	I kolo	ILI kolo	Sonda
<i>tip_elementa_u_fajlu</i>	1	2	3	4	5	0


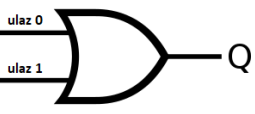
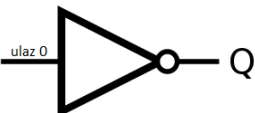
Elementi digitalne mreže

Podržati sledeće elemente digitalne mreže: logička kombinaciona kola, digitalne izvore signala i sonde. Moguće vrednosti signala u bilo kojoj tački mreže su 0 ili 1.

Logička kombinaciona kola koja treba podržati su NE, I i ILI kolo. Potrebno je maksimalno pojednostaviti dodavanje u sistem bilo koje druge vrste logičkih kombinacionih kola. Predvideti da svako logičko kolo može imati tačno jedan izlazni pin i proizvoljan broj ulaznih pinova (podrazumevano 2), osim NE kola koje uvek ima tačno jedan ulazni pin. Broj ulaznih pinova AND i OR kola je dat kao dodatan parametar u opisu elementa u ulaznom fajlu. Ne treba realizovati bidirekzione pinove. Smatrati da ne postoje kašnjenja kroz logička kola.

Digitalne izvore signala koje treba podržati su generator signala takta i ručno podesiv generator signala. Potrebno je maksimalno pojednostaviti dodavanje u sistem bilo koje druge vrste izvora signala. Generator signala takta je periodičan izvor čija je frekvencija (MHz) data kao dodatan parametar u opisu elementa u ulaznom fajlu. Prvu polovinu svoje periode generator signala takta generiše 0 na svom izlazu, a drugu polovinu svoje periode generiše 1. Izvor sa ručnim podešavanjem je neperiodičan izvor čiji su relativni vremenski trenuci (μ s) u kojima izvor menja vrednost svog izlaza dati kao dodatni parametri u opisu elementa u ulaznom fajlu. Početna vrednost na svim izvorima signala je 0.

Sonde su elementi mreže koji služe da prikupljaju informacije o vrednosti signala u tački za koju su priključeni. Sonde nemaju dodatne parametre u opisu elementa u ulaznom fajlu.

dvoulazno I kolo	Ulazi		Izlaz
	0	0	0
	0	1	0
	1	0	0
	1	1	1
dvoulazno ILI kolo	Ulazi		Izlaz
	0	0	0
	0	1	1
	1	0	1
	1	1	1
NE kolo	Ulaz	Izlaz	
	0	1	
	1	0	

Simulacija signala na izlazima

Implementirati operaciju `void Simulator::simulate(const string& filepath);` koja simulira digitalnu mrežu i ispisuje rezultate simulacije koje su prikupile sonde u izlazne fajlove. Putanja do fajlova je zadata parametrom `filepath`. Rezultati simulacije sadrže sve vremenske trenutke u kojima se dešavaju promene koje registruju sonde. Za svaku sondu se kreira zaseban fajl sa rezultatima tako što se na podrazumevano ime fajla doda identifikator sonde (pr. za fajl `output.txt` i sonde sa identifikatorima 2 i 4 treba da se kreiraju dva fajla `output_2.txt`, `output_4.txt`, itd.). Svaka promena se ispisuje u zasebnom redu u formatu `stara_vrednost -> nova_vrednost: vremenski_trenutak_promene`.

Test funkcija

Javni test sadrži funkciju `void test();` koja testira rad sa simulatorom digitalnih mreža. Studentima je javno dostupna implementacija funkcije `test` i mogu da je menjaju da bi dodatno testirali svoj kod kao što je opisano komentarima u kodu. Studentima su dati ulazni fajlovi koji predstavljaju test primere kao i izlazni fajlovi koji predstavljaju očekivane izlaze simulacije radi mogućnosti provere. Uz svaki ulazni test primer date su šema digitalne mreže koja odgovara ulaznom fajlu i izgled signala izlaza svakog od elemenata u mreži radi lakšeg snalaženja.

Tehnički zahtevi i smernice za izradu rešenja

Programski sistem realizovati tako da bude detaljno komentarisano, modularan i lako proširiv novim klasama i operacijama. Klasa *Simulator* i njene operacije moraju biti imenovane prema zahtevima iz domaćeg zadatka. Programski kod klasa rasporediti u odgovarajuće **.h** i **.cpp** fajlove. Iz kolekcije standardne biblioteke dozvoljeno je koristiti sledeće tipove: `<string>`, `<vector>`, `<list>`, `<stack>`, `<queue>`. Ukoliko u zadatku nešto nije dovoljno jasno definisano, treba usvojiti razumnu pretpostavku i na temeljima te pretpostavke nastaviti izgrađivanje svog rešenja.

VAŽNE NAPOMENE

Za uspešno odbranjen domaći zadatak potrebno je na odbrani pokazati kod podeljen na odgovarajuće projekte, **.h** i **.cpp** fajlove.

1. Klase kojima su implementirani osnovni koncepti treba da budu smeštene u poseban projekat rešenja koji se prevodi kao statička biblioteka (`simulator.lib`).
2. Glavni program treba da se nalazi u posebnom projektu koji se prevodi kao Win32 Console Application (`testdz2.exe`) fajl i koji treba povezati sa statičkom bibliotekom. Glavni program napraviti u fajlu `Main.cpp`. U glavni projekat je dodatno potrebno uključiti fajl `Test.cpp`.