Elektrotehnički fakultet – Univerzitet u Beogradu

KATEDRA ZA ELEKTRONIKU

Vežbe

Praktikum iz Virtuelne instrumentacije

3. Čas

Školska 2019/20

Asistent: Haris Turkmanović (*haris@etf.rs*)

3. Zadatak

Potrebno je kreirati VI instrument koji realizuje iscrtavanje funkcije $y = a \cdot x^2 + b \cdot x \quad + c$. Korisniku se nudi mogućnost da proizvoljno podesi vrednosti parametara a, b i c kao i mogućnost da odredi krajnju granicu opsega za promenljivu *x* u kom će biti iscrtan grafik, tj. [0, KrajnjaGranica]. VI treba realizovati tako da se nakon modifikacije vrednosti parametara grafik funkcije iscrtava bez ponovnog pokretanja programa.

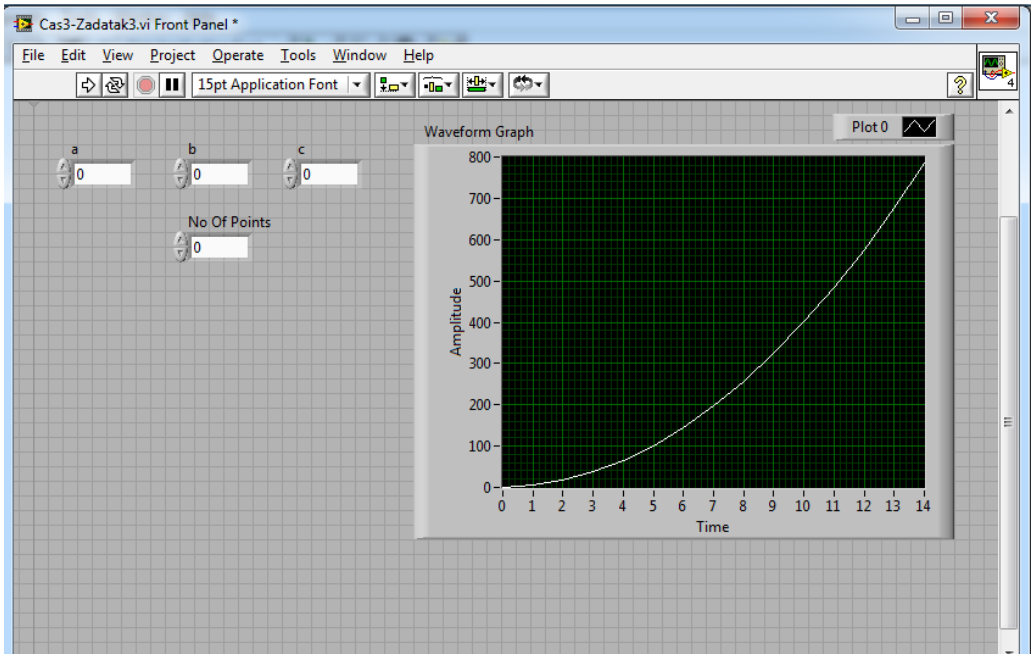
3.1. Rešenje:

U dosadašnjem kreiranju virtuelnih instrumenata koristili smo funkcije koje su dostupne u okviru *Simulate signal* bloka. U okviru ovog primera, biće objašnjen način za kreiranje proizvoljnih funkcija koje mogu biti od interesa tokom rada sa softverskim paketom LABView. Najpre ćemo kreirati prednji panel VI. Ovaj panel se kreira u okviru *Front Panel* prozora VI u okviru koga je, za realizaciju traženog VI, potrebno dodati kontrole navedene u tabeli 3-1

3-1 Podešavanja kontrola za zadatak 3

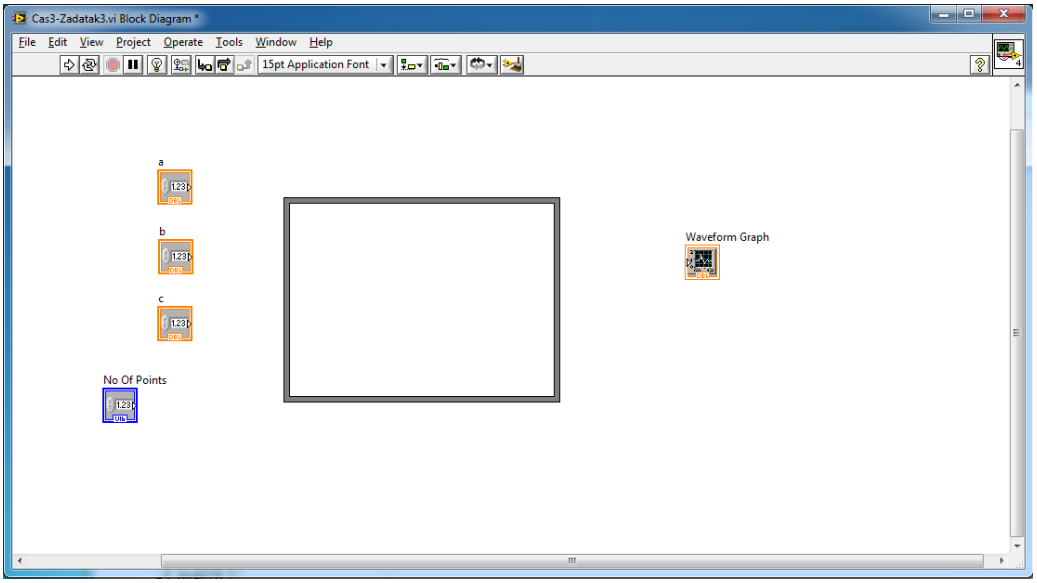
Generički naziv kontrole	Num Control	Num Control	Num Control	Num Control
Naziv	a	b	c	No. Of Points
Opseg	0-10	0-10	0-10	0-1000
Tip Podataka	Doble	Double	Double	U16

Pored kontrola navedenih u okviru tabele, potrebno je dodati i *Waveform Graph*. Nakon dodavanja svih navedenih kontrola, prednji panel VI izgleda kao na slici 3.1.



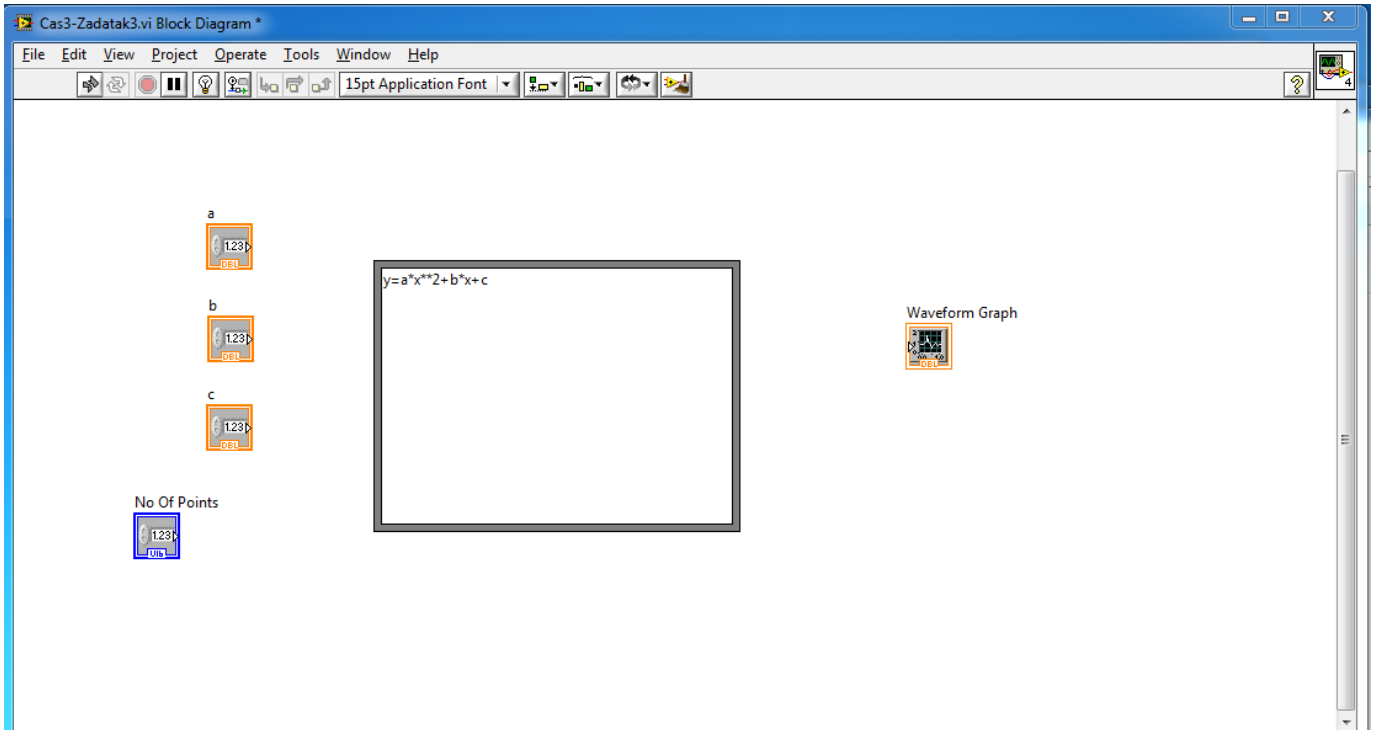
3.1 - Prednji panel VI nakon dodavanja kontrola i Waveform graph-a

Sada je potrebno realizovati funkcionalnost kojom se postiže generisanje grafika proizvoljnih funkcija. Jedan od načina da se ostvari ova funkcionalnost je uz pomoć ***Formula Node*** strukture. Ova struktura se nalazi u okviru podgrupe *Structures* grupe *Programming*. Nakon odabira ove strukture u okviru palete funkcija, a zatim iscrtavanja okvira ove strukture, dobijamo izgled blok dijagrama VI kao na slici 3.2.



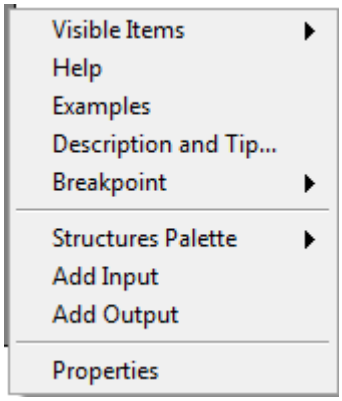
3.2 - Blok dijagram VI nakon dodavanja Formula Node strukture

Sada je potrebno u okviru *Formula Node* strukture napisati funkciju koju generiše ta struktura a zatim definisati ulaze i izlaze te strukture. Dodavanje formule koja će se izvršavati u okviru ove strukture postiže se tako što se levim klikom miša klikne u prostor koji pripada okviru ove strukture nakon čega se, u okviru strukture *Formula Node*, pojavljuje kursor što je indikator da je *Formula Node* struktura spremna za unos formule. Nakon unosa formule, blok dijagram VI izgleda kao na slici 3.3.



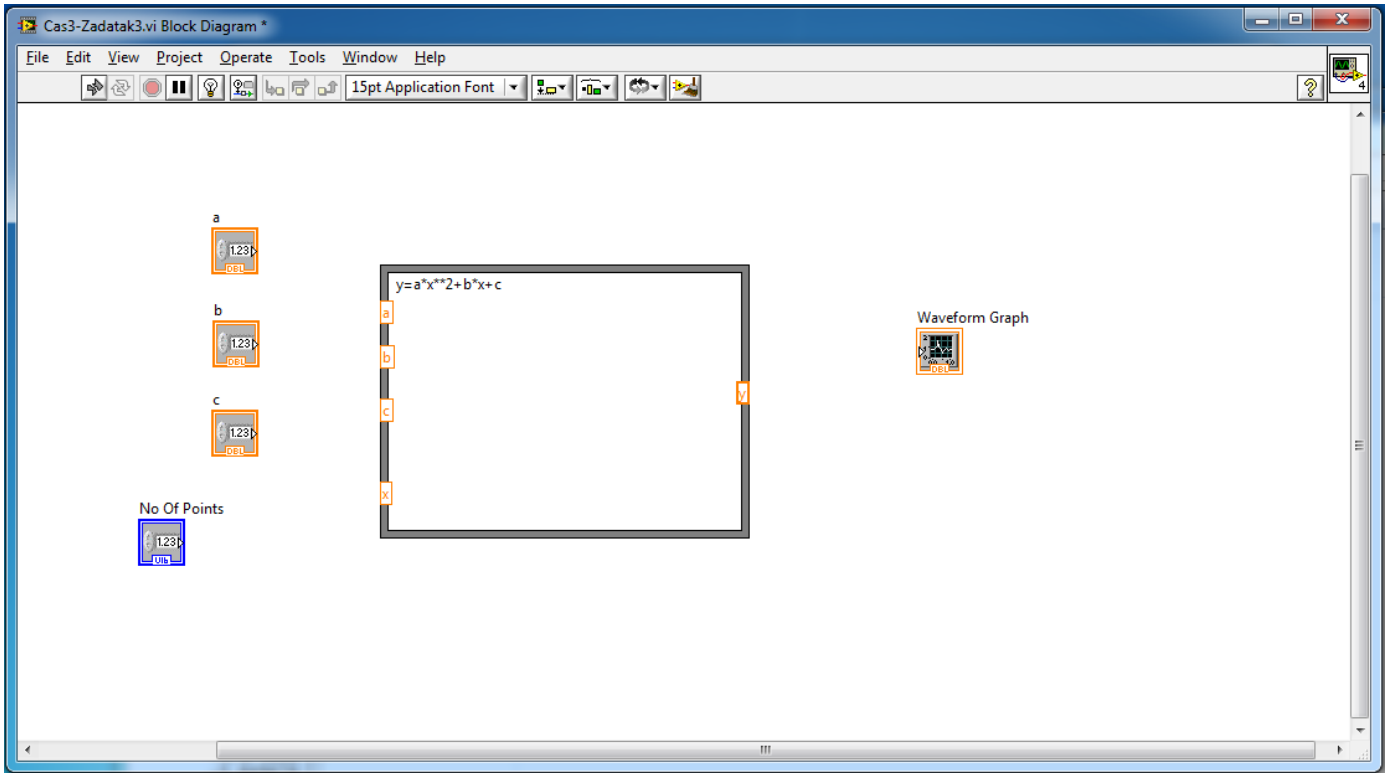
3.3 - Blok dijagram VI nakon unosa formule u okvir Formula Node bloka

Sada je potrebno definisati ulaze i izlaze ove strukture. Ulazi u strukturu su promenljive koje se javljaju u funkciji, dok je izlaz zapravo vrednost funkcije. Definisanje ulaza i izlaza strukture se postiže tako što se na okvir *Formula Node* strukture klikne desnim klikom miša nakon čega se otvara meni kao na slici 3.4.



3.4 - Opcija Formula Node strukture

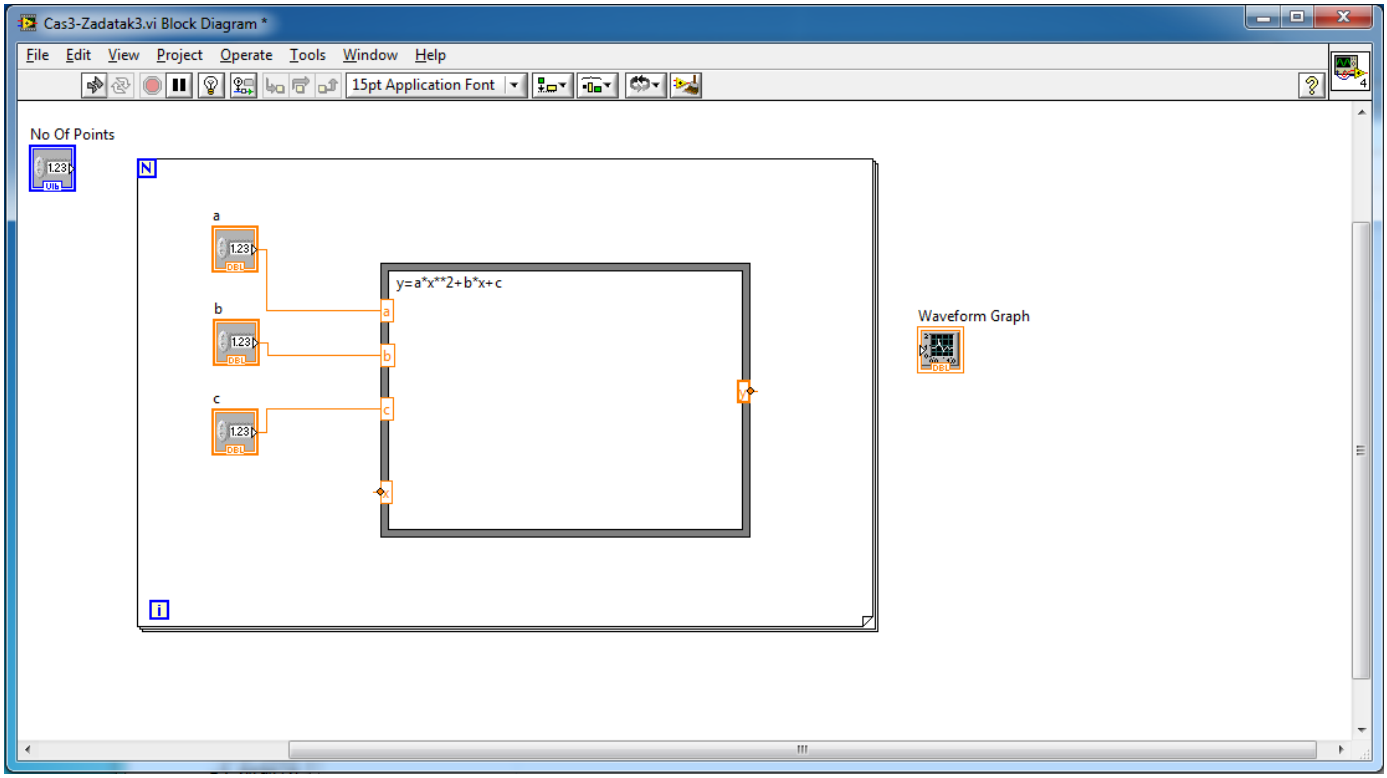
Za definisanje ulaza u strukturu potrebno je odabrati opciju *Add Input*. Ulazi u ovu strukturu su zapravo korisničke promenljive (a, b i c) i promenljiva funkcije označena sa *x*. Pored definisanja ulaza, potrebno je i definisati izlaze sto se postiže odabirom opcije *Add Output*. U ovoj formuli imamo samo jedan izlaz (*y*). Sada je za svaku od promenljivih potrebno definisati poseban ulaz i za vrednost funkcije *y* potrebno je definisati izlaz. Nakon definisanja ulaza i izlaza dobija se izgled blok dijagrama kao na slici 3.5.



3.5 - Blok dijagram VI nakon definisanja ulaza i izlaza u Formula Node blok

Sada je potrebno povezati numeričke kontrole a, b i c sa odgovarajućim ulazima *Formula Node* strukture. Nakon toga ostaje ne povezan ulaz *x*. Ovaj ulaz služi za definisanje tačaka u kojima se računa vrednost funkcije. Pošto je tekstom postavke ovog zadatka traženo da realizujemo funkcionalnost koja omogućava iscrtavanje grafika u proizvoljnom opsegu, jedan od načina za realizaciju ovog zadatka je

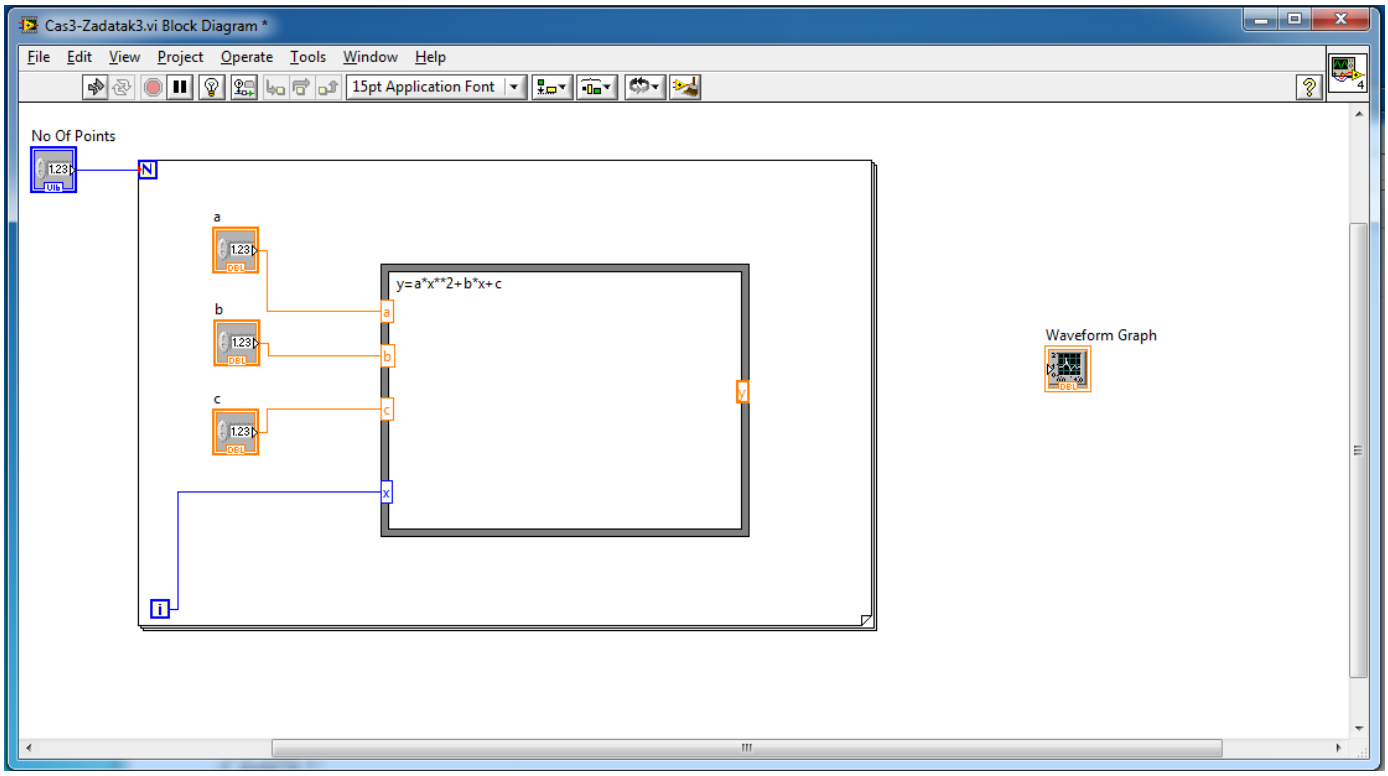
posredstvom *for* programske strukture. Dakle, potrebno je celu *Formula Node* strukturu, kao i kontrole a,b i c, dodati u okvir *for* programske strukture koja je dostupna u okviru podgrupe *Structures* grupe *Programming*. Nakon toga, dobijamo izgled blok dijagrama VI kao na slici 3.6.



3.6 - Blok dijagram VI nakon dodavanja for strukture

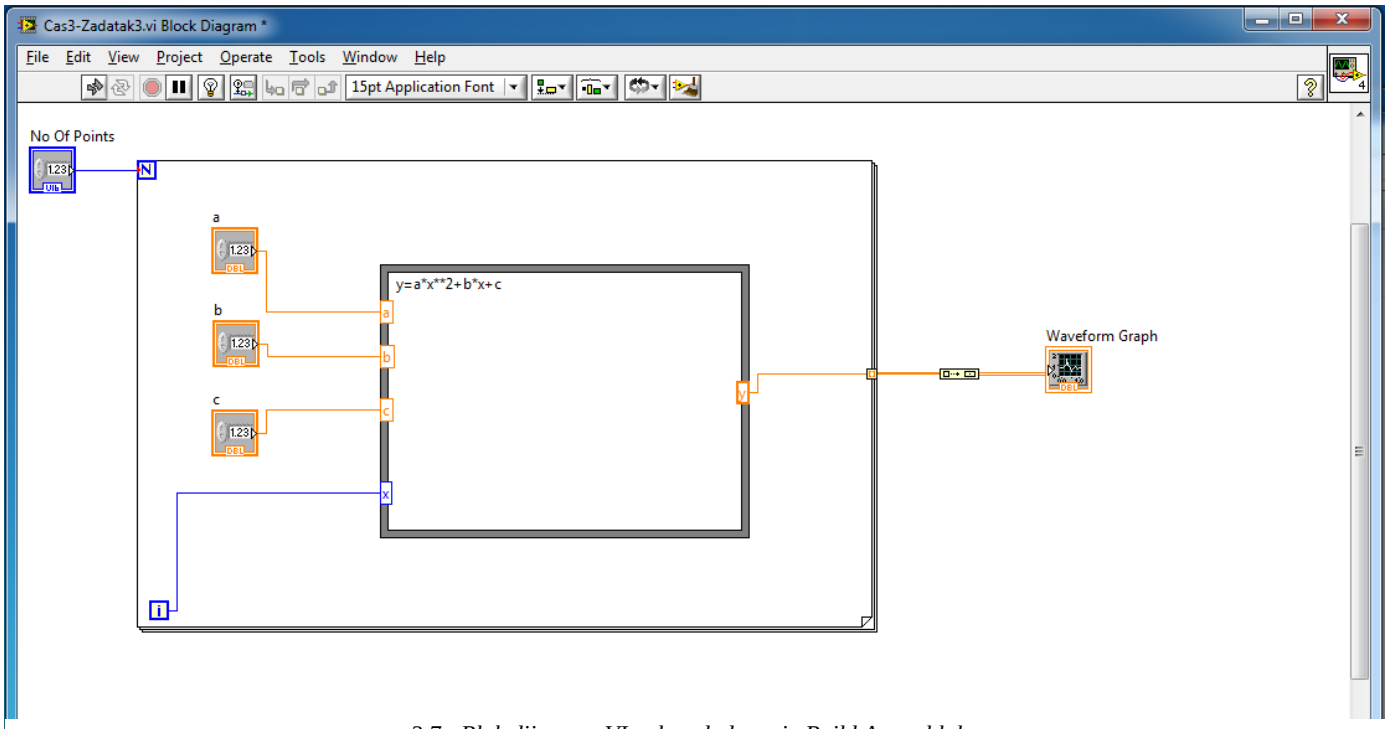
Sa slike 3.6 uočavamo da *for* programska struktura ima ulaz *N* (gornji levi ugao okvira *for* strukture) kojim se definiše broj iteracija u *for* strukturi, tj. zapravo se definiše broj ponavljanja blok dijagrama koji se nalazi u okviru *for* strukture. U našem konkretnom primeru, parametar *N* definisan je vrednošću kontrole *No Of Points* koju definiše korisnik posredstvom prednjeg panela virtuelnog instrumenta. Drugim rečima, broj iteracija ekvivalentan je definisanoj vrednosti krajnjeg opsega za promenljivu *x* koju je korisnik definisao.

Pored parametra *N*, *for* programska struktura ima i parametar *i* koji predstavlja brojač iteracija. Taj brojač, u ovom primeru, zapravo određuje vrednost promenljive *x*. Nakon iznetih zahteva, potrebno je povezati blok dijagram tako da odgovara blok dijagramu sa slike



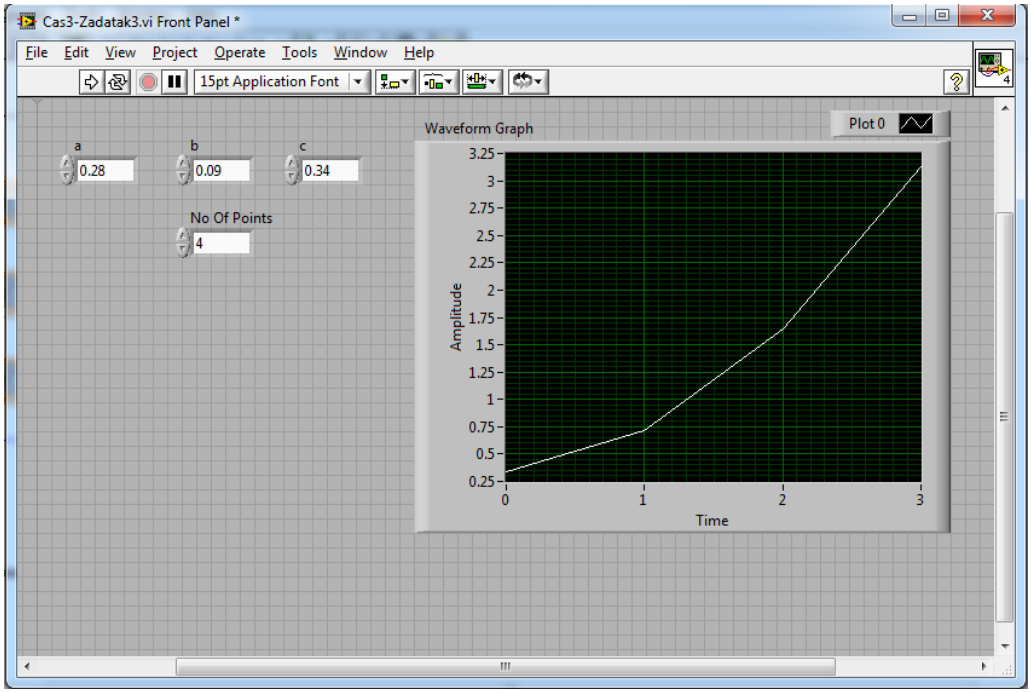
3.8 - Blok dijagram VI nakon definisanja parametara N i x

Dakle, na osnovu slike 3.6 zaključujemo da je još ostao nepovezan izlaz *y* koji predstavlja vrednost funkcije. Pre nego objasnimo kako treba povezati izlaz *y* sa blokom *Waveform graph*, potrebno je primetiti da svaka iteracija u *for* programskoj strukturi generiše jednu vrednost *y*. Sa druge strane,za grafik koji želimo da iscrtamo neophodan je skup svih tih vrednosti. Dakle, potrebno je nekako pamtiti sve generisane vrednosti u nekom nizu, a zatim na osnovu vrednosti u nizu nacrtati grafik. Da bi ostvarili tu funkcionalnosti, pre ulaza



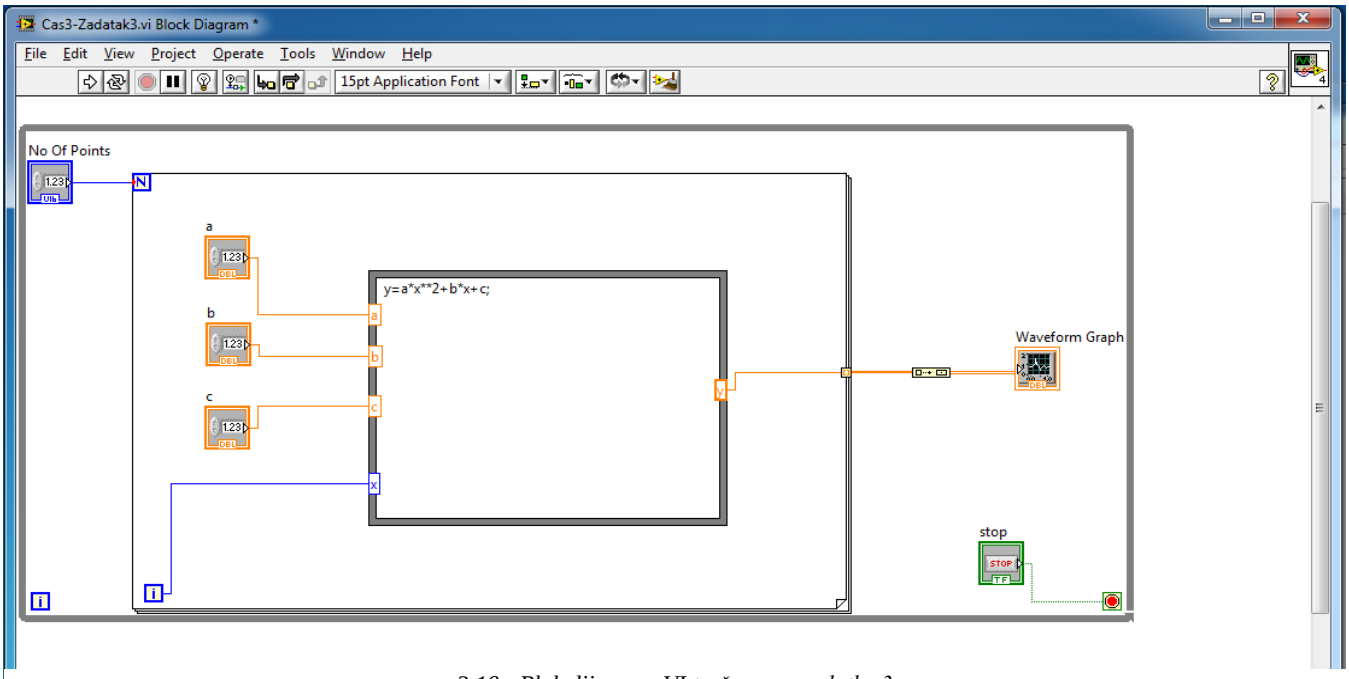
3.7 - Blok dijagram VI nakon dodavanja Build Array bloka

u *Waveform graph*, dodajemo blok ***Build Array*** koji se nalazi u okviru podgrupe ***Array*** grupe ***Programming.*** Nakon ubacivanja tog bloka i povezivanja tog bloka sa izlazom *y* i ulazom u *Waveform Graph*, dobijamo blok dijagram VI kao na slici 3.7. Sada je potrebno pokrenuti program, i ukoliko je sve urađeno kao što je navedeno u ovom uputstvu, dobija se izgled prednjeg panela kao na slici 3.9.



3.9 - Izgled prednjeg panela VI

Nakon pokretanja programa, iscrta se grafik i program se zaustavi. Pošto je postavkom zadatka traženo da se program kontinualno izvršava, potrebno je ceo blok dijagram smestiti u okvir while petlje i dodati stop dugme. Nakon toga, blok dijagram sistema izgleda kao na slici 3.10. i predstavlja krajnji blok dijagram kojim se realizuje traženi VI.



3.10 - Blok dijagram VI traženog u zadatku 3

Zadaci za samostalni rad:

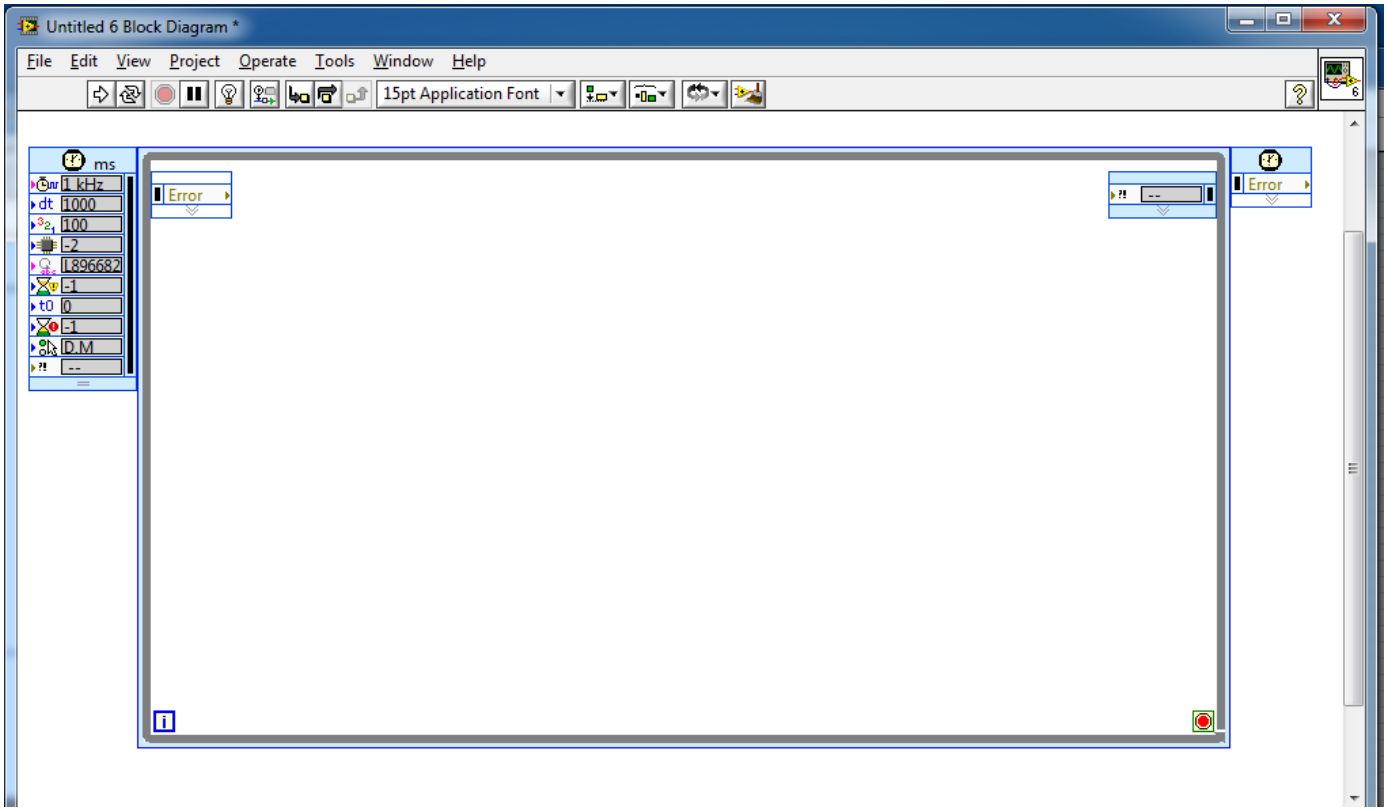
1. Modifikovati realizovani VI tako da se korisniku nudi mogućnost definisanja broja tačaka (rezolucije) za iscrtavanje grafika u prethodno definisanom opsegu
2. Modifikovati realizovani VI tako da se na grafiku iscrtava i funkcija $y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$.

4. Zadatak

Potrebno je realizovati VI koji implementira funkcionalnost četvorobitnog brojača. Rezultat brojanja se prikazuje na diodama u okviru prednjeg panela VI u okviru koga je takođe moguće i podešavati periodu brojanja.

4.1. Rešenje

Osnovu realizacije ovog brojača predstavlja mogućnost implementacije vremenski periodičnih događaja. Ti događaji se, između ostalog, mogu implementirati i posredstvom programske strukture označene kao ***Timed Loop*** koja se nalazi u okviru podgrupe *Structures* grupe *Programming*. Nakon označavanja okvira ove strukture, blok dijagram VI izgleda kao na slici 4.1.

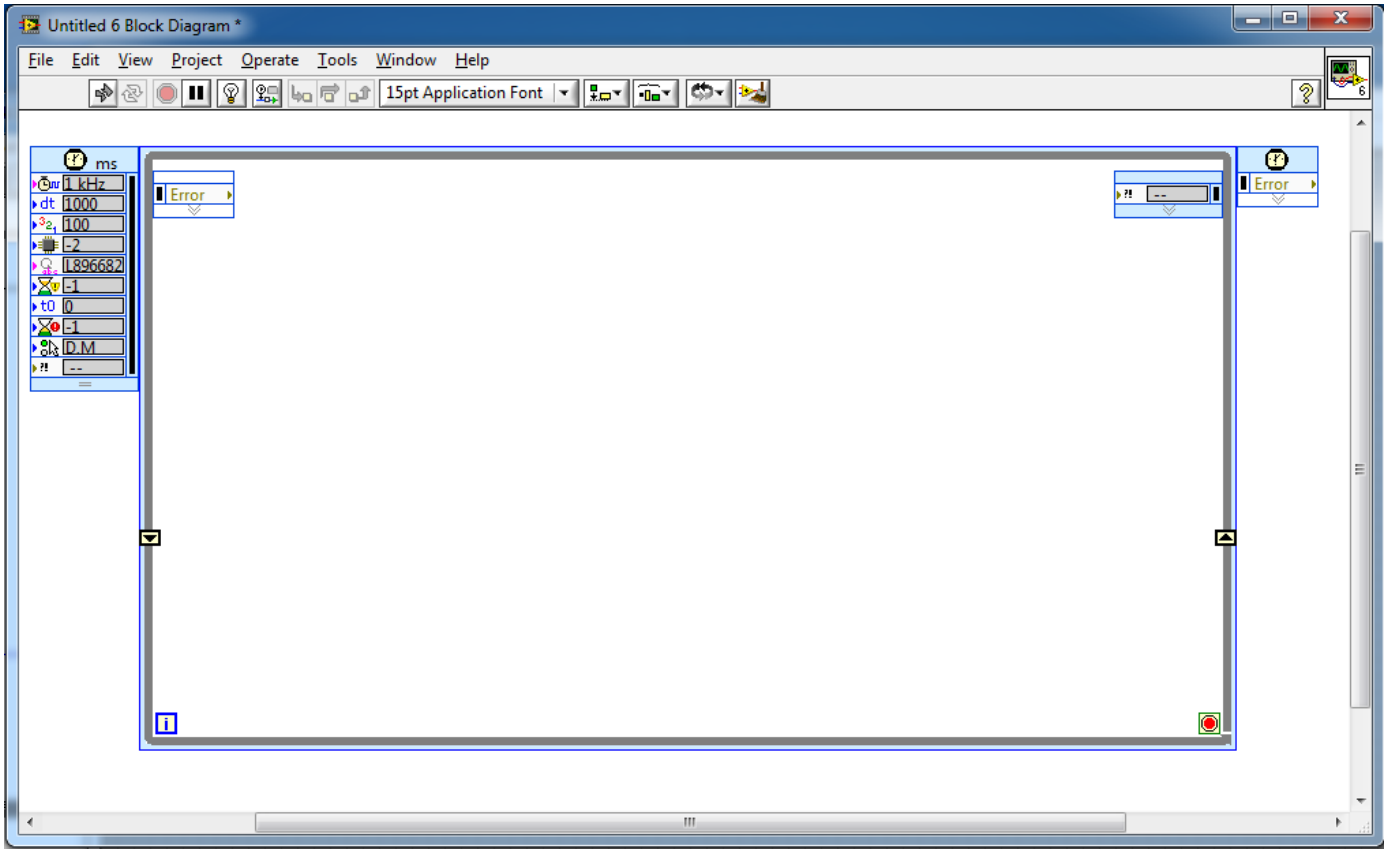


4.1 - Timed Loop struktura

Sa slike 4.1. možemo primetiti da je ova struktura slična *for* programskoj strukturi. Za razliku od *for* programske strukture, gde se naredna iteracija nastavlja odmah nakon kraja prve iteracije, u okviru ove programske strukture naredna iteracija se nastavlja nakon isteka određenog vremenskog intervala. Vremenski parametri se mogu definisati proizvoljno, a deo strukture u okviru koga se nalaze vremenski

parametri smešten je u gornjem levom uglu okvira *Timed loop* strukture. Na osnovu dosadašnjeg znanja iz LABView softverskog paketa, moguće je uočiti da se svi ti parametri mogu definisati na osnovu kontrola u okviru prednjeg panela virtuelnog instrumenta. Međutim, pre nego dođenmo do implementacije tog dela funkcionalnosti VI, potrebno je još malo diskutovati o samim funkcionalnim mogućnostima *Timed Loop* strukture.

Jedna od funkcionalnosti koju je moguće simulirati koristeći ovu strukturu jeste simulacija sekvencijalnog bloka označenog kao D flip flop. Ovu sekvencijalnu mrežu moguće je simulirati tako što se desnim klikom miša klikne na okvir ove strukture a zatim se odabere opcija ***Add Shift Register*.** Nakon toga ova struktura izgleda kao na slici 4.2.



4.2 - Timed loop struktura sa dodatim Shift registrom

Sa slike 4.2. vidimo da su se na levom i desnom okviru pojavile strelice. Strelica na gore predstavlja ulaz u D flip-flop dok strelica na dole predstavlja izlaz D flip flopa. Vrednost koja se pojavi na ulazu pojaviće se i na izlazu posle vremenskog intervala definisanog u okviru ***Timed loop*** strukture. Sada, da bi kreirali četvorobitni brojač potrebno je prisetiti se formula za svaki od izlaza tog četvorobitnog brojača koje glase:

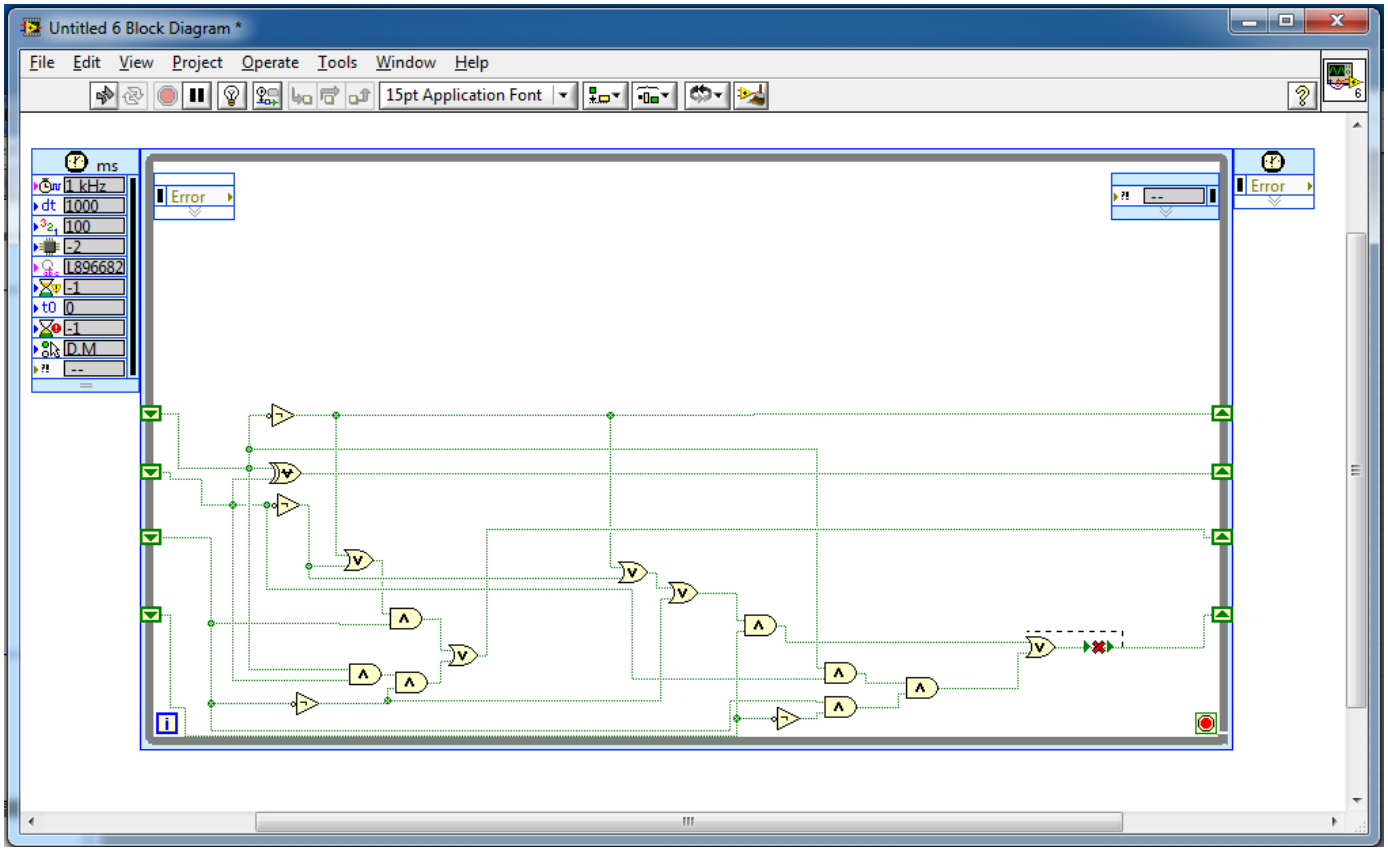
$$D_0 = \overline{Q_0}$$

$$D_1 = Q_0 \oplus Q_1$$

$$D_2 = Q_2\overline{Q_0} + Q_2\overline{Q_1} + \overline{Q_2}Q_1Q_0$$

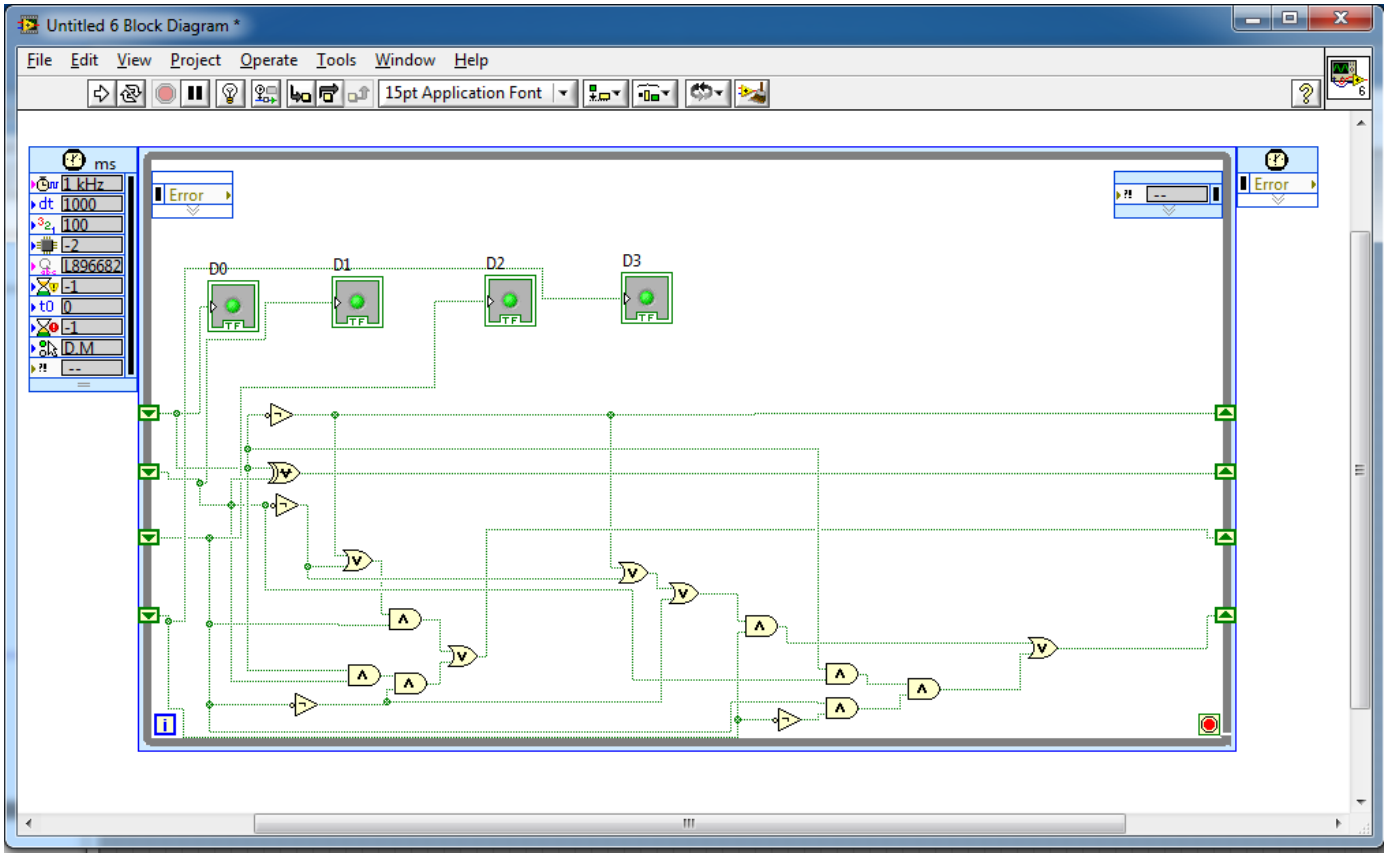
$$D_2 = Q_3\overline{Q_2} + Q_3\overline{Q_0} + \overline{Q_3}Q_1 + \overline{Q_3}Q_2Q_1Q_0$$

Svi potrebni logički operatori nalaze se u okviru podgrupe *Booelan* grupe programing. Nakon primene tih operatora u cilju realizacije navedenih logičkih funkcija dobijamo izgled blok dijagrama kao na slici 4.3.



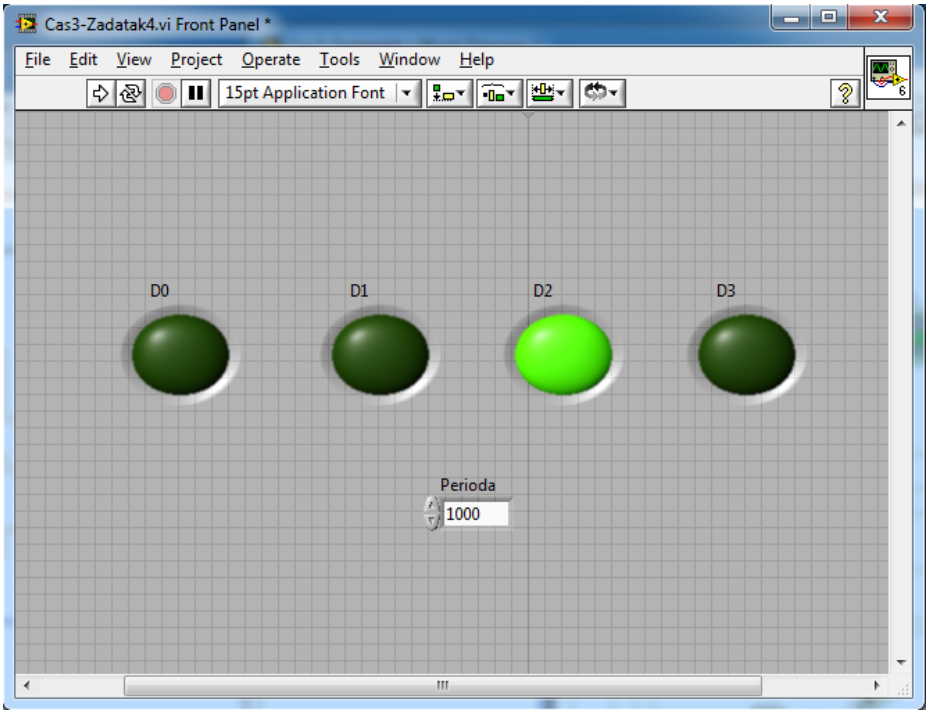
4.3 - Blok dijagram VI nakon implementacije logičkih funkcija

Sada je na svaki od izlaza D flip flopa potrebno dodati po jedan *boolean* indikator. Nakon toga blok šema ozgleda kao na slici 4.5.



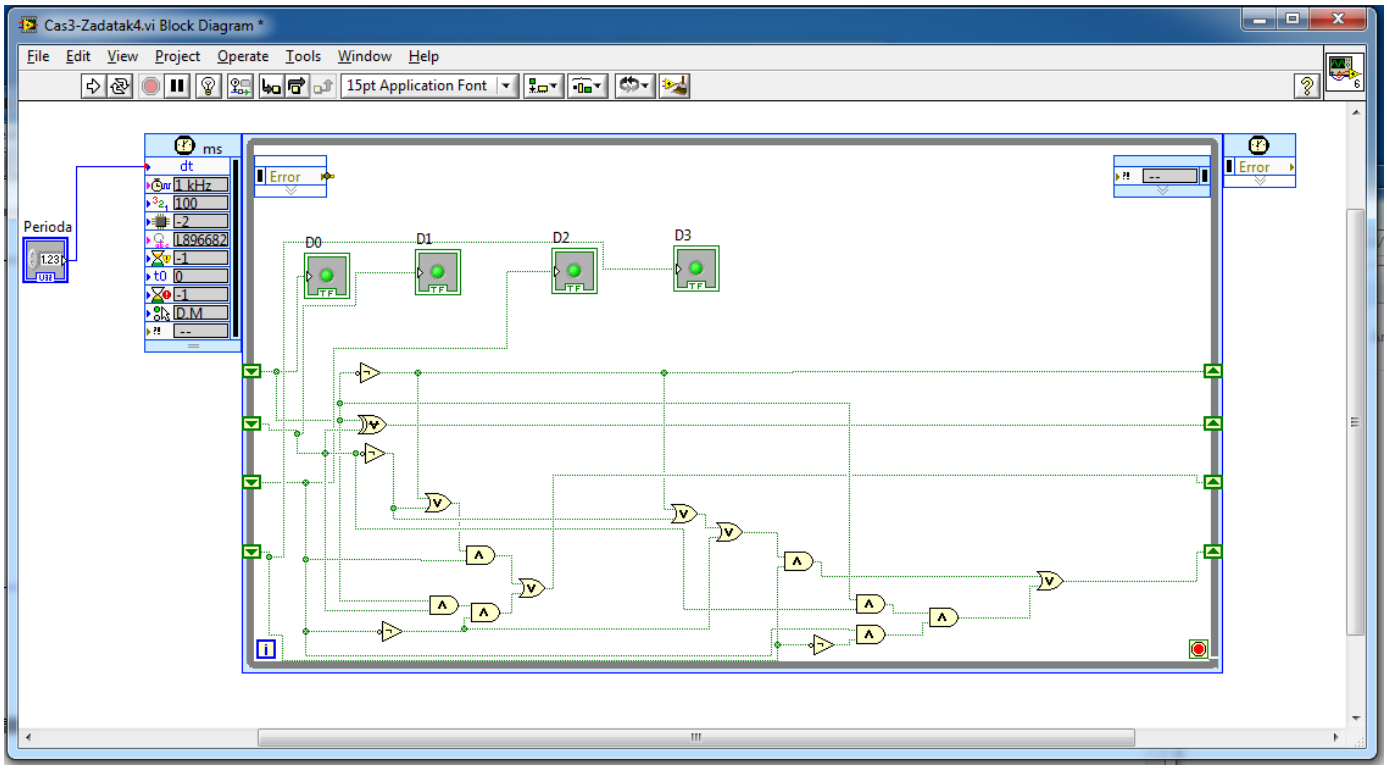
4.5 - Blok dijagrama VI nakon dodavanja dioda

Sada je potrebno pokrenuti program nakon čega prednji panel VI uključuje diode u redosledu koji odgovara sekvenci brojanja od 0-15. U okviru prednjeg panela potrebno je još dodati kontrolu kojom se definiše perioda inkrementiranja brojača. Nakon toga prednji panel VI izgleda kao na slici 4.4.



4.4- Izgled prednjeg panela VI nakon dodavanja kontrole za kontrolu periode brojanja

Blok dijagram koji implementira funkcionalnost podešavanja periode prikazan je na slici 4.6.



4.6 - Blok dijagrama traženog VI iz zadatka 4

Nakon pokretanja programa možemo uočiti da je perioda brojanja podešena na onu vrednost na kojoj je numerička kontrola bila neposredno pre pokretanja. Dakle, da bi podesili periodu brojanja neophodno je najpre podesiti periodu pa restartovati program.

Zadaci za samostalni rad

1. Modifikovati zadatak 4 tako da se ostvari funkcionalnost četvorobitnog brojača po modulu 3
2. Istražiti mogućnost modifikovati zadatka 4 tako da umesto *Timed loop* strukture bude korišćena *While loop* struktura.