6 Pemrograman LabVIEW

6.1 Istilah-Istilah Penting

Sebelum membahas mengenai pemrograman LabVIEW, sebaiknya pembaca mengenal istilah-istilah penting berikut ini.

1. **G:** dari kata "graphical", merupakan sebutan untuk bahasa pemrograman LabVIEW yang menggunakan kode berbentuk grafis.
2. **VI:** dari kata "virtual instruments", merupakan sebutan untuk program yang dibuat dengan LabVIEW.
3. **SubVI:** sebagian besar kode program di LabVIEW adalah subVI. SubVI ini sama seperti subrutin dalam bahasa pemrograman teks, yaitu sebuah VI di dalam VI. SubVI ini berbentuk ikon, atau kotak kecil dengan gambar yang unik di dalamnya, dengan kaki input berada di sebelah kiri dan kaki output berada di sebelah kanan.
4. **Front Panel:** adalah tampilan program. Objek-objek pada jendela ini akan terlihat oleh pengguna saat program dijalankan. Objek-objek pada Front Panel ini, akan secara otomatis memiliki representasi

ikonnya di Block diagram, khususnya untuk objek-objek yang membawa data, baik data yang masuk dari pengguna ke program, maupun data yang keluar dari program ke pengguna.

5. **Block Diagram:** adalah tempat pembuatan program. Jendela ini tidak akan terlihat oleh pengguna saat program dijalankan. Pembuatan program di sini dilakukan dengan cara menempatkan beberapa node dan menghubungkannya satu sama lain.

6. **Node:** adalah semua objek di jendela Block Diagram, yang memiliki input/output dan melakukan operasi tertentu ketika dijalankan, termasuk di dalamnya subVI, terminal, struktur, dan fungsi.

7. **Terminal:** adalah ikon-ikon di Block diagram yang mewakili objek-objek di Front Panel, di mana objek-objek tersebut membawa data, baik data yang masuk dari pengguna ke program, maupun data yang keluar dari program ke pengguna. Contoh Terminal adalah Control dan Indicator.

8. **Control:** adalah semua objek di Front Panel yang memasukkan data dari pengguna ke program. Disebut juga Terminal Input. Contoh Control adalah knob, tombol, sakelar, dan alat input lainnya.

9. **Indicator:** adalah semua objek di Front Panel yang mengeluarkan atau menampilkan data dari program ke pengguna. Disebut juga Terminal Output. Contoh Indicator adalah grafik, LED.

10. **Struktur:** adalah semua bentuk alur pemrograman, seperti perulangan, percabangan, urutan, dan lain-lain. Contoh struktur ini adalah For Loop, While Loop, Sekuensial, Case. Struktur ini hanya ada di jendela Block diagram, berbentuk blok yang dapat diatur luasannya, dan hanya bekerja untuk ikon-ikon yang ditempatkan di dalamnya.

11. **Fungsi:** adalah semua kode-kode dasar yang telah disediakan untuk membuat subVI. Contoh fungsi seperti Add, Subtract.

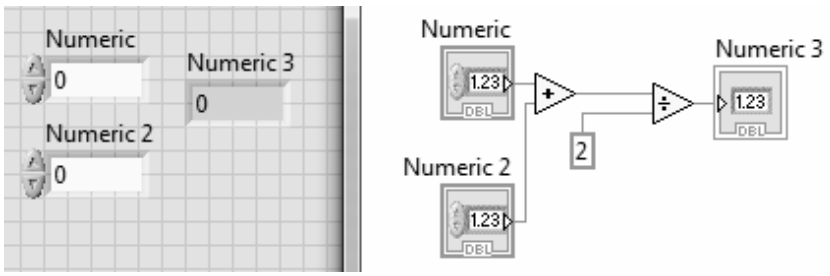
12. **Wire:** atau kawat, digunakan untuk menghubungkan ikon-ikon, sekaligus untuk menunjukkan aliran data dan tipe data. Beberapa warna kawat dan tipe datanya dapat dilihat dalam tabel berikut.

Tipe data	Sakelar	Array 1D	Array 2D	Warna kawat
Numerik				Oranye/biru[*)]
Boolean				Hijau
String				Merah muda

Keterangan [*)]**:** warna kawat oranye untuk tipe data bilangan riil (*float*), sedangkan warna kawat biru untuk tipe data bilangan bulat (integer).

13. **Pemrograman Dataflow (aliran data):** yaitu konsep pemrograman yang akan mengeksekusi node ketika semua inputnya telah tersedia. Ketika node ini telah selesai dieksekusi, maka data akan diteruskan dari output node tersebut ke node berikutnya.

Untuk lebih jelas mengenai konsep pemrograman **Dataflow** ini, perhatikan program mencari rata-rata berikut ini, yang akan menjumlahkan 2 angka dan kemudian membaginya dengan 2. Dalam hal ini, program dieksekusi dari kiri ke kanan, bukan karena objek-objek tersebut ditempatkan dalam urutan demikian, namun karena fungsi **Divide (÷)** tidak akan bekerja sebelum fungsi **Add (+)** dieksekusi.

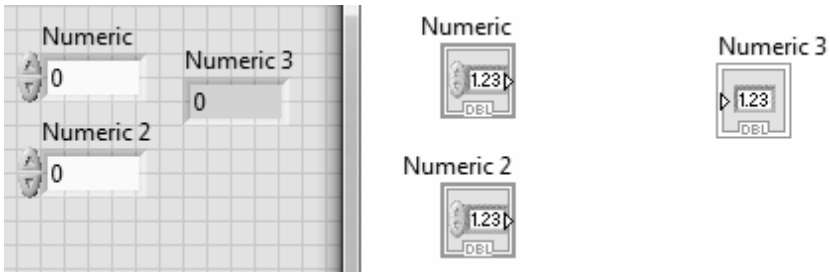


Gambar 6.1 Konsep Dataflow pada program mencari rata-rata

6.2 Membuat SubVI

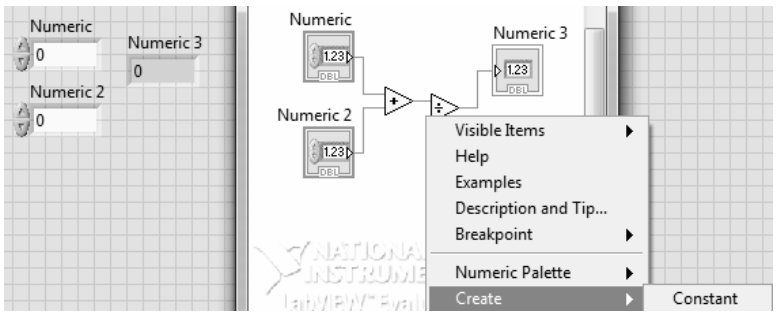
Berikut ini langkah-langkah untuk membuat sebuah subVI:

1. Buatlah program mencari rata-rata seperti Gambar 6.1 di atas, yaitu dengan menempatkan 2 objek Num Ctrl dan sebuah objek Num Ind pada jendela Front Panel. Secara otomatis ketiga objek tersebut memunculkan 3 ikon Terminal di Block diagram.



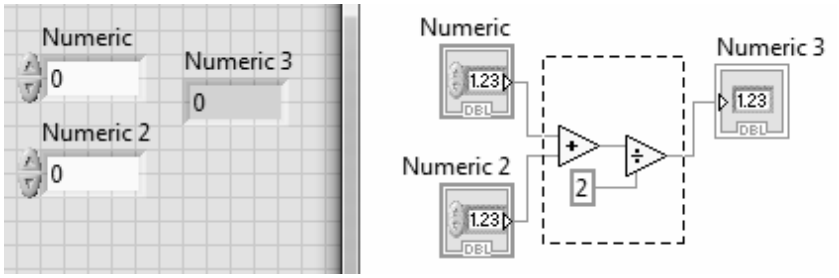
Gambar 6.2 Menempatkan 2 objek Num Ctrl dan sebuah objek Num Ind

2. Kemudian pada jendela Block diagram, tambahkan fungsi **Add**, **Divide** dan konstanta angka 2. Untuk memunculkan angka 2 ini, dapat dilakukan dengan menempatkan pointer mouse pada salah satu kaki input **Divide**, kemudian klik kanan untuk memunculkan menu popup, kemudian pilih **Create Constant**.



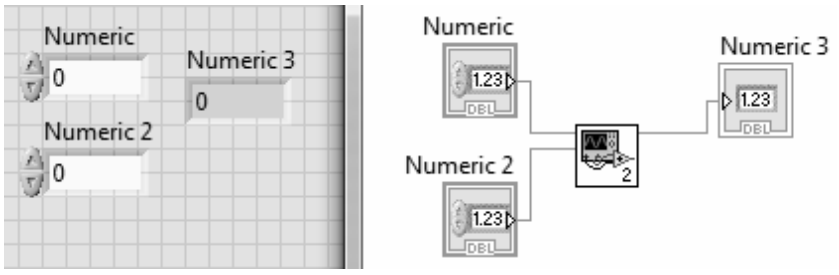
Gambar 6.3 Memunculkan konstanta (angka 2) dari menu popup

3. Setelah program selesai disusun seperti Gambar 6.1, maka pada jendela Block diagram, klik dan gerakkan pointer mouse hingga melingkupi fungsi **Add**, **Divide** dan konstanta angka 2.



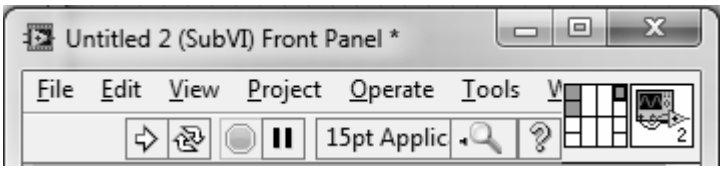
Gambar 6.4 Klik dan gerakkan mouse melingkupi fungsi dan konstanta

4. Pada menu **Edit** di jendela Block diagram, pilih **Create SubVI**, maka fungsi dan konstanta tersebut berubah menjadi sebuah ikon (subVI).



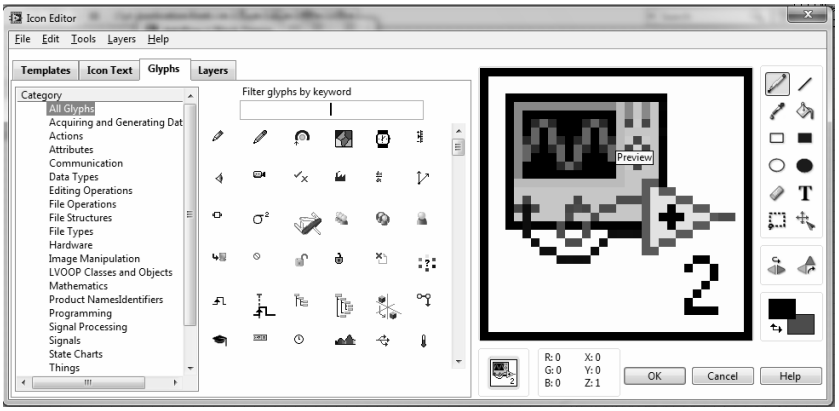
Gambar 6.5 Create subVI maka fungsi-fungsi menjadi sebuah ikon

5. Klik 2 kali pada ikon atau subVI tersebut, maka akan terbuka jendela Front Panel dari subVI tersebut.



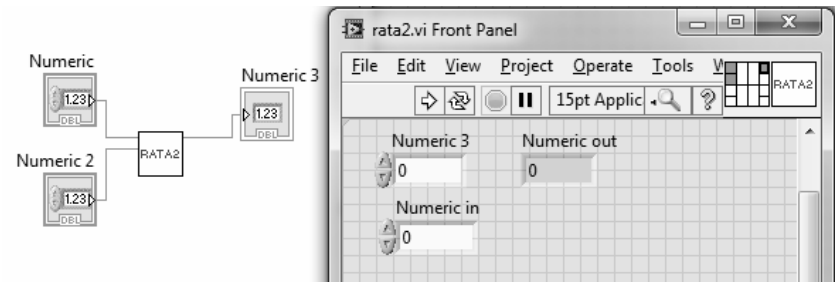
Gambar 6.6 Klik 2 kali subVI, maka muncul jendela Front Panel

6. Perhatikan pojok kanan atas jendela Front Panel tersebut, tampak kotak **Connector Pane** dan kotak **Icon subVI**. Kotak **Connector Pane** tersebut menunjukkan ada 2 buah kaki input di bagian kiri dan sebuah kaki output di bagian kanan subVI. Sedangkan kotak Icon menunjukkan tampilan subVI. Pembaca dapat mengubah tampilan subVI tersebut dengan mengklik 2 kali pada kotak Icon, hingga muncul jendela Icon Editor seperti berikut ini.



Gambar 6.7 Jendela Icon Editor untuk mengubah tampilan subVI

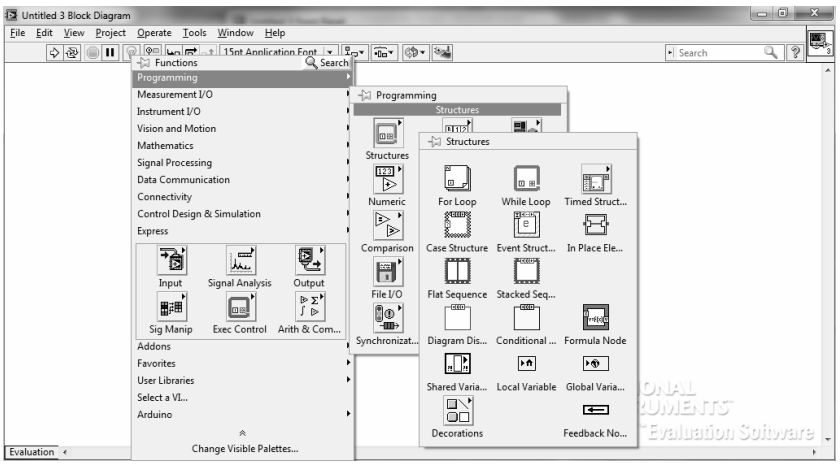
7. Ubah tampilan ikon dengan gambar yang unik kemudian simpan. Maka pembuatan subVI telah selesai.



Gambar 6.8 Tampilan subVI telah diubah

6.3 Struktur Pemrograman

LabVIEW menyediakan bermacam-macam struktur pemrograman. Untuk mengambilnya, klik kanan jendela Block diagram hingga muncul palet **Functions**, kemudian pilih kategori **Programming**, dilanjutkan dengan **Structures**, maka akan terlihat bermacam-macam struktur pemrograman, termasuk di dalamnya While Loop, For Loop, Shift Register, Case Structure.



Gambar 6.9 Kategori Structures di palet Functions

6.3.1 While Loop

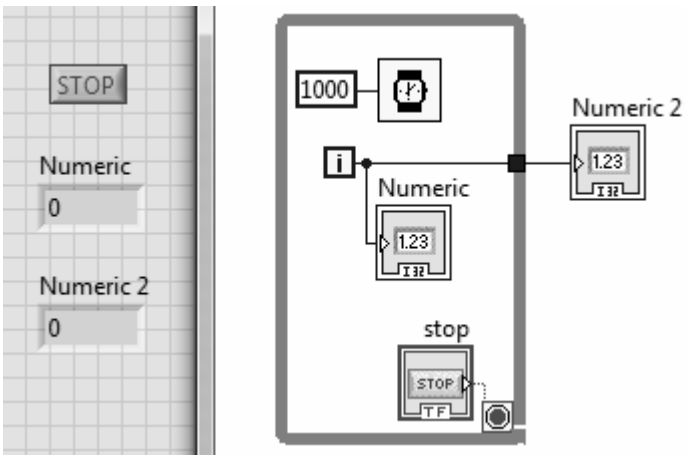
Struktur **While Loop** memiliki 3 komponen utama, yaitu sebuah blok yang dapat diatur luasannya, sebuah terminal input **conditional** (**Stop If True**) dan sebuah terminal output **counter** (**i**).

Struktur While Loop ini akan terus mengeksekusi ikon-ikon di dalam bloknya berulang-kali hingga terminal input **conditional (Stop if True)** mendapat nilai **False**, sebaliknya perulangan akan berhenti ketika mendapat nilai **True**.

Terminal input **conditional** tersebut dapat diubah dari **Stop If True** menjadi **Continue If True**, yang mana akan membuat kondisi yang berlawanan, yaitu hanya akan melakukan perulangan ketika terminal input **conditional** tersebut mendapat nilai **True**, dan berhenti ketika mendapat nilai **False**.

Terminal output **counter (i)** akan memberikan nilai jumlah perulangan yang telah selesai dilakukan, yang dimulai dari 0.

Berikut contoh program dengan struktur **While Loop**. Tampak dalam gambar 2 buah objek Numeric Indicator, yaitu **Numeric** dan **Numeric2** sama-sama mendapat nilai **counter (i)**, hanya saja **Numeric** berada di dalam **While Loop**, sedangkan **Numeric2** berada di luar **While Loop**. Ketika program dijalankan, nilai **Numeric** akan bertambah mengikuti nilai **counter (i)**, sedangkan nilai **Numeric2** sama sekali tidak bertambah, karena nilai counter (i) tidak akan diteruskan ke Numeric2 sebelum **While Loop** berhenti. Ketika tombol Stop ditekan, maka **While Loop** berhenti, barulah nilai **counter (i)** diteruskan ke **Numeric2**.



Gambar 6.10 Numeric2 tidak akan berubah hingga While Loop selesai

6.3.2 For Loop

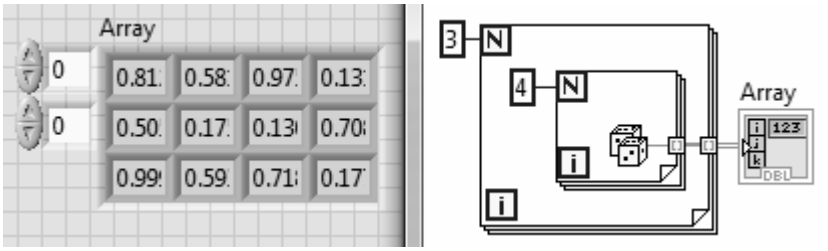
Struktur **For Loop** memiliki 3 komponen utama, yaitu sebuah blok yang dapat diatur luasannya, sebuah terminal input **N** dan sebuah terminal output **counter** (**i**).

Struktur For Loop ini akan mengeksekusi ikon-ikon di dalam bloknya berulang-kali, dengan jumlah perulangan sebanyak nilai yang dimasukkan ke terminal input **N**. Apabila nilai yang dimasukkan ke terminal input **N** sebesar 0, maka For Loop tidak akan melakukan perulangan. Begitu pula apabila tidak ada nilai yang dimasukkan ke terminal input **N**, maka For Loop juga tidak akan melakukan perulangan, kecuali ada sebuah data array yang dimasukkan melalui dinding blok **For Loop** dan dibuat **auto-indexing**. Jumlah perulangan untuk cara auto-indexing ini adalah sebanyak jumlah data array tersebut.

Sama seperti pada **While Loop**, terminal output **counter (i)** akan memberikan nilai jumlah perulangan yang telah dilakukan. Pembaca juga dapat menambahkan terminal input **conditional** seperti pada struktur

While Loop, dengan cara mengklik kanan dinding blok **For Loop**, dan pada menu popup yang muncul, pilih **Conditional Terminal**.

Berikut contoh program **For Loop** untuk membuat sebuah **Array 2D** atau disebut juga **Matriks** dengan ukuran **3x4** (3 baris 4 kolom) dan dengan nilai elemen-elemennya berupa angka **random**.



Gambar 6.11 Membuat Matriks 3x4 dengan For Loop

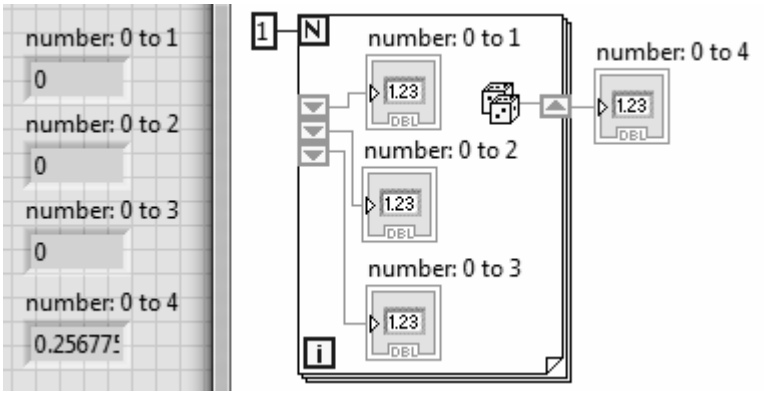
6.3.3 Shift Register

Struktur **Shift Register** ini tidak tersedia di palet **Functions**, karena penggunaannya hanya bisa diterapkan pada struktur **For Loop** dan **While Loop** saja. Untuk menggunakan struktur **Shift Register** ini, klik kanan dinding blok **For Loop** atau **While Loop**, dan kemudian pilih **Add Shift Register** dari menu popup yang muncul.

Shift Register ini digunakan untuk meneruskan data dari satu perulangan ke perulangan berikutnya. Bentuk **Shift Register** ini berupa pasangan terminal di dinding kiri dan kanan blok, bergambar tanda panah ke bawah dan ke atas.

Untuk mengetahui secara jelas cara kerja **Shift Register** ini, tambahkan **Shift Register** untuk meneruskan data angka **random** pada blok **For Loop**, dengan jumlah perulangan N sebanyak 1. Tarik ke bawah terminal **Shift Register** di dinding kiri hingga muncul tambahan 2 terminal.

Tambahkan **Indicator** untuk setiap terminal **Shift Register** tersebut, baik di dinding kiri maupun di kanan, dengan cara mengklik kanan ujung terminal, sehingga muncul menu popup, dan pilih **Create Indicator**. Kemudian tekan tombol Run, dan perhatikan bagaimana data digeser.



Gambar 6.12 Cara kerja Shift Register menggeser data

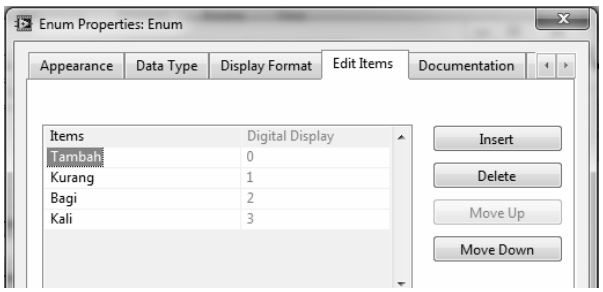
6.3.4 Case Structure

Struktur **Case** memiliki 2 atau lebih blok, namun hanya satu saja yang dieksekusi pada satu waktu. Struktur Case ini sama seperti Instruksi **If Then Else, Switch** atau **Select Case** pada pemrograman berbasis teks.

Blok mana yang akan dieksekusi, tergantung pada nilai input di terminal **selector (?)**. Di bagian atas setiap blok terdapat label yang menunjukkan nilai input yang mengaktifkannya. Tipe data nilai input ini bisa berupa Boolean (True atau False), Integer, String, atau Enumerasi. Enumerasi adalah tipe data yang terdiri atas sekumpulan nilai.

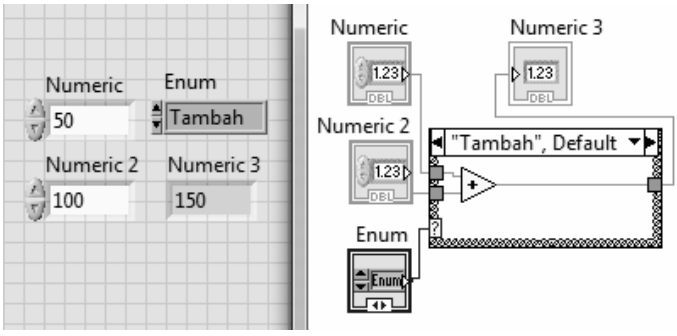
Berikut ini diperlihatkan contoh struktur **Case** menggunakan data Enumerasi, untuk membuat kalkulator sederhana dengan 2 masukan. Objek Enumerasi ini dapat diambil di palet Controls di Front Panel, yaitu

di kategori **Classic. Ring & Enum**. Pembaca juga dapat menemukan objek Enum tersebut dengan bantuan tombol **Search**. Setelah diperoleh, tempatkan objek tersebut di **Front Panel**, dan kemudian klik-kanan hingga muncul menu popup dan pilih **Edit Items**. Setelah itu di kolom tabel pada jendela yang muncul, berturut-turut ketik kata "Tambah", "Kurang", "Bagi" dan "Kali" untuk nilai 0, 1, 2 dan 3.



Gambar 6.13 Data Enumerasi dengan nama Tambah, Kurang, Bagi, Kali

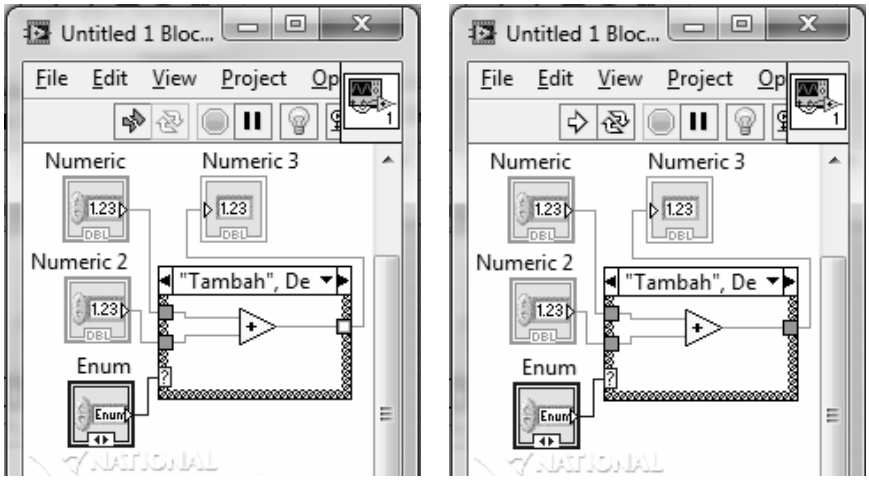
Kemudian pada jendela Block Diagram, hubungkan **icon Enum** tersebut dengan terminal **selector (?)** struktur **Case**, maka secara otomatis, label pada bagian atas blok struktur **Case** akan berubah dari True dan False, menjadi "Tambah", dan "Kurang". Klik kanan pada dinding blok struktur **Case** tersebut dan pilih **Add Case After** pada menu popup yang muncul, maka akan muncul sebuah blok yang berlabelkan "Bagi". Ulangi lagi untuk memunculkan blok yang berlabelkan "Kali".



Gambar 6.14 Struktur Case dengan data Enum sebagai selector-nya

Perhatikan bahwa pada blok berlabel “Tambah” diikuti dengan kata **Default**. Blok Default ini digunakan untuk menyediakan pilihan blok apabila tidak ada input yang sesuai. Jadi, apabila input pada terminal **selector** struktur **Case** di luar keempat data Enum tersebut, maka struktuf **Case** akan mengaktifkan blok berlabel **Default** ini. Pembaca dapat memindahkan blok Default ini ke blok yang lain, dengan memilih **Make This The Default Case** pada blok yang akan dijadikan blok Default.

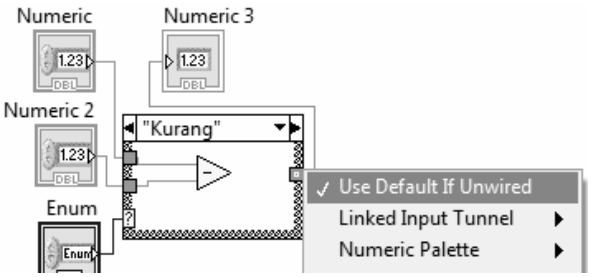
Data dapat dimasukkan ke satu atau beberapa blok di struktur Case. Data juga dapat dikeluarkan dari blok-blok di struktur Case, hanya saja, untuk data yang dikeluarkan, semua blok pada struktur **Case** harus terhubung dengan garis data tersebut. Apabila ada blok tidak terhubung, maka kotak terminal di dinding tempat garis data tersebut diteruskan akan berwarna putih. Sedangkan apabila semua terhubung, maka kotak tersebut berwarna penuh, sesuai warna tipe datanya.



(a) (b)

Gambar 6.15 (a) Tidak semua blok terhubung, (b) Semua blok terhubung

Pembaca dapat mengklik kanan kotak terminal dan memilih *Use Default If Unwired* pada menu popup yang muncul, untuk membuat blok-blok yang tidak terhubung secara otomatis memberikan nilai default (nilai 0).

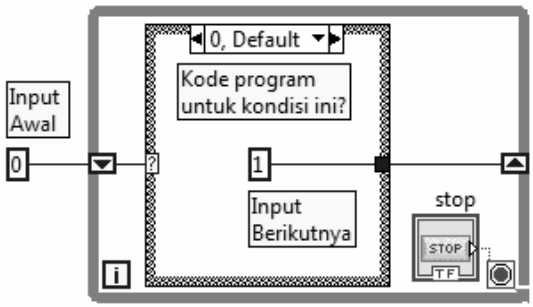


Gambar 6.16 Use Default If Unwired

6.4 State Machine

State Machine adalah sebuah cara pemrograman yang memungkinkan program untuk merespons secara cerdas terhadap input yang diberikan. State Machine yang baik adalah yang dapat memberikan semua kemungkinan kondisi yang akan terjadi untuk setiap keadaan awal dan input pemicuan tertentu.

Di LabVIEW, cara pemrograman State Machine dapat diimplementasikan secara mudah, yaitu hanya dengan menggabungkan 3 buah struktur: While Loop, Case Structure dan Shift Register.



Gambar 6.17 Diagram dasar sebuah State Machine

Sebagai gambaran mengenai manfaat State Machine ini, perhatikan contoh aplikasi berikut.

Diinginkan sebuah simulasi kalkulator sederhana, yang memiliki tombol angka 0 sampai 9, dengan operasi Tambah (+), Kurang (-), Kali (x), dan Bagi (/). Cara kerja kalkulator ini sebagai berikut:

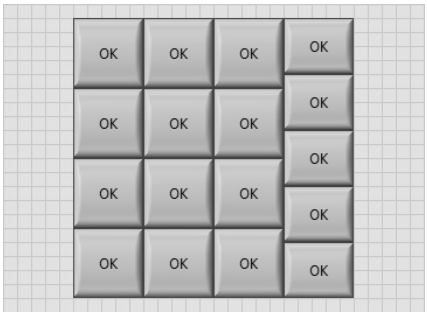
1. Mula-mula pengguna memasukkan angka pertama, dengan menekan salah satu tombol angka (0-9). Kalkulator akan menampilkan angka pertama tersebut.
2. Kemudian pengguna menekan salah satu tombol operator perhitungan (+,=,x,/). Tampilan kalkulator tidak berubah, tetap menampilkan angka pertama.
3. Kemudian pengguna memasukkan angka kedua, dengan menekan salah satu tombol angka. Kalkulator menampilkan angka kedua ini.
4. Kemudian pengguna menekan tombol sama dengan (=), maka kalkulator akan menghitung dan menampilkan hasilnya.

Dari cara kerja di atas, dapat ditentukan komponen dari State Machine:

- **Keadaan**: tampilan kalkulator setiap tahapnya.
- **Input pemicu**: semua tombol.

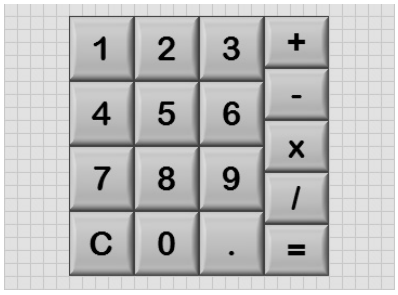
Berikut ini langkah-langkah pembuatan simulasi kalkulator sederhana menggunakan konsep State Machine:

1. Ambil tombol OK Button dari palet **Controls** sebanyak 17 buah, dan tempatkan pada jendela Front Panel dan susun seperti gambar berikut. Hilangkan label setiap tombol dengan mengklik kanan tombol, pilih **Visible Items**, dan hilangkan centang pada **Label**.



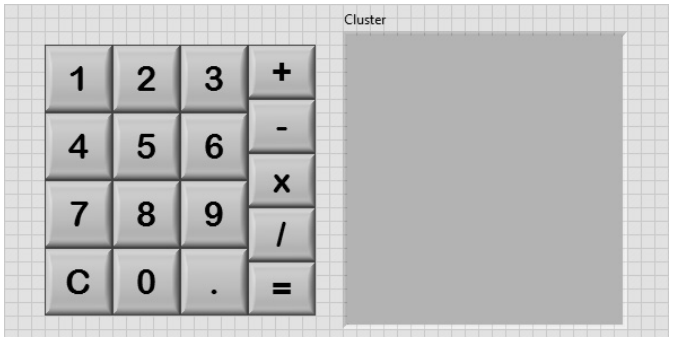
Gambar 6.18 Menempatkan 17 buah tombol OK Button

2. Ganti tulisan OK pada setiap tombol dengan angka dan tanda operator seperti gambar berikut ini.



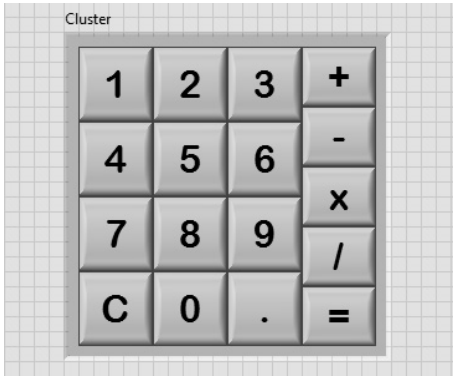
Gambar 6.19 Mengganti tulisan OK dengan angka dan tanda operator

3. Ambil objek Cluster dari palet Controls di kategori **Array, Matrix & Cluster**. Tempatkan pada Front Panel dan perbesar ukurannya.



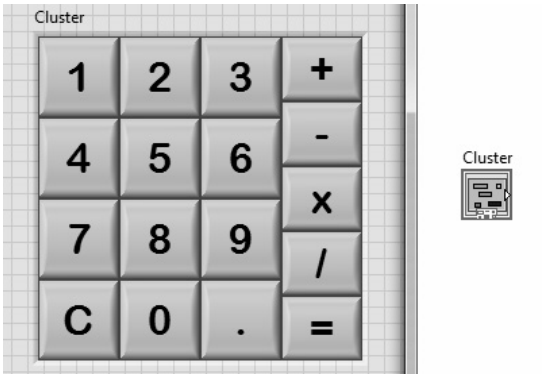
Gambar 6.20 Mengambil Cluster dan memperbesar ukurannya

4. Setelah ukuran Cluster cukup besar untuk memuat semua tombol, berikutnya pindahkan semua tombol ke dalam Cluster.



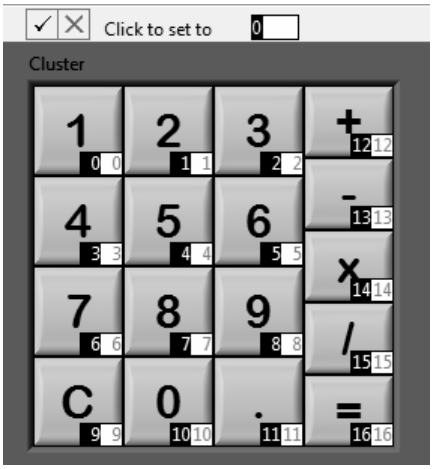
Gambar 6.21 Menempatkan semua tombol ke dalam Cluster

5. Klik kanan pada Cluster, dan pilih **AutoSizing** kemudian **Size to Fit** untuk menyesuaikan ukuran Cluster hingga tepat sama dengan blok tombol. Hilangkan pula Label Cluster. Perhatikan jendela Block Diagram, tampak bahwa semua ikon tombol dapat digabungkan hingga menjadi sebuah ikon saja, yaitu ikon Cluster.



Gambar 6.22 Menempatkan semua tombol ke dalam Cluster

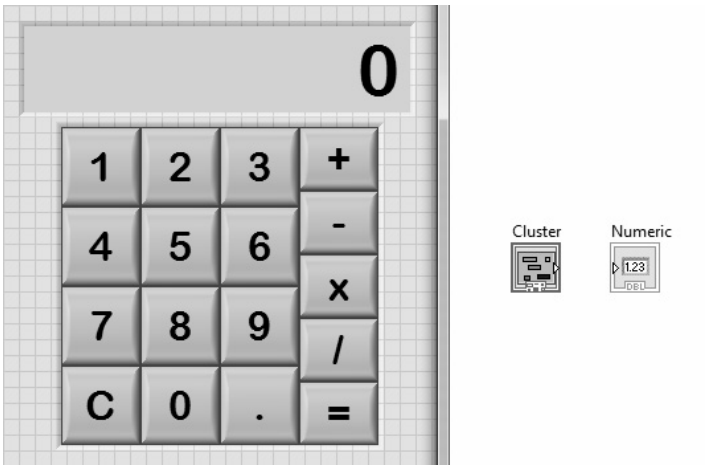
6. Klik kanan pada Cluster, dan pilih **Reorder Controls in Cluster**, maka akan muncul nomor indeks Cluster di tiap tombol secara berurutan.



Gambar 6.23 Muncul nomor indeks ketika Reorder Controls in Cluster

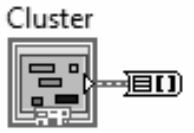
7. Atur urutan nomor indeks Cluster sehingga sesuai dengan gambar di atas, yaitu dimulai dari tombol 1 untuk nomor indeks 0, hingga tombol sama dengan (=) untuk nomor indeks 16.

8. Tambahkan objek Numeric Indicator, dan dengan tool **Text Settings** pada Toolbar, perbesar ukuran angkanya dan atur rata kanan.



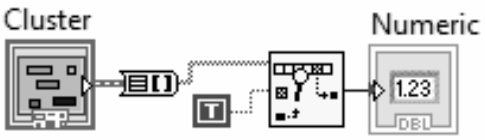
Gambar 6.24 Tambahkan objek Num Ind sebagai tampilan kalkulator

9. Diinginkan setiap kali salah satu tombol ditekan, objek **Num Ind** dapat menampilkan nomor indeks tombol tersebut. Untuk itu tambahkan pada ikon Cluster fungsi **Cluster to Array** (dapat diambil di kategori Array), untuk mengubah Cluster menjadi Array.



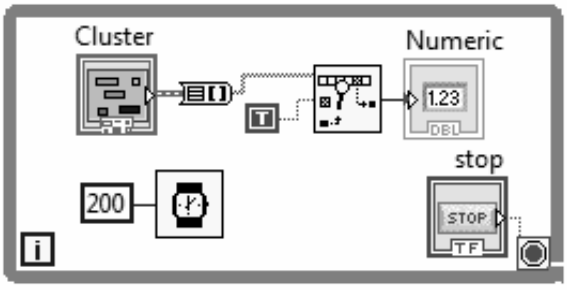
Gambar 6.25 Tambahkan Cluster to Array

10. Kemudian tambahkan fungsi **Search 1D Array** (dapat diambil di kategori Array) untuk menemukan manakah tombol dengan indeks yang bernilai **True,** dalam arti saat itu sedang ditekan, dan kemudian hubungkan output dari fungsi tersebut ke ikon **Num Ind**.



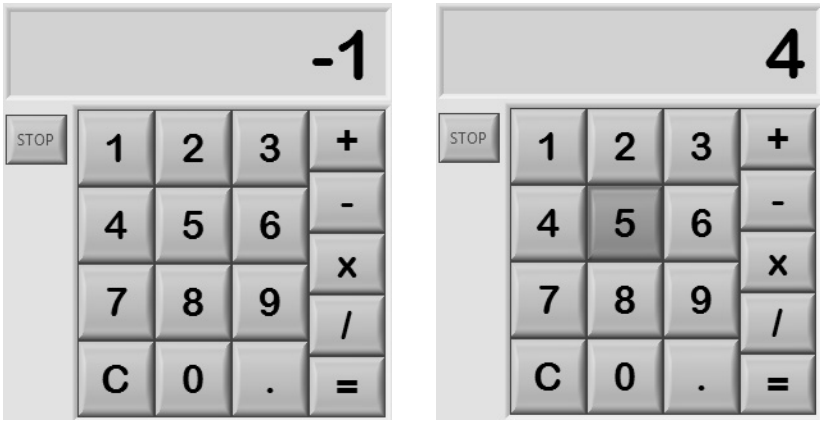
Gambar 6.26 Search 1D Array untuk mencari tombol yang ditekan

11. Tambahkan pula struktur **While Loop** untuk menjalankan terus-menerus program tersebut, dan sebuah fungsi **wait (ms)** di kategori Timing, untuk memberi tunda waktu sebesar 200 ms.



Gambar 6.27 Menambahkan While Loop dan Wait(ms) sebesar 200

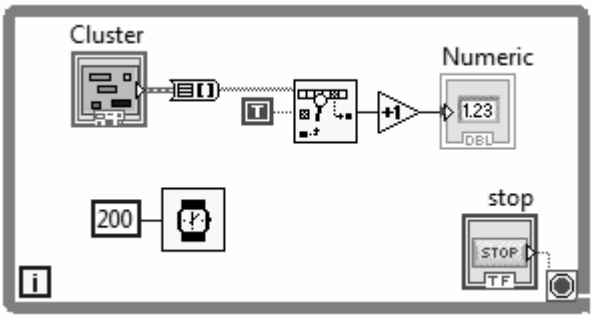
12. Jalankan program (tekan tombol Run), dan perhatikan bahwa ketika tombol tidak ditekan, maka ditampilkan angka -1, sedangkan ketika salah satu tombol ditekan, dan kemudian dilepas, muncul nomor indeksnya sesaat, kemudian kembali ke angka -1.



(a) (b)

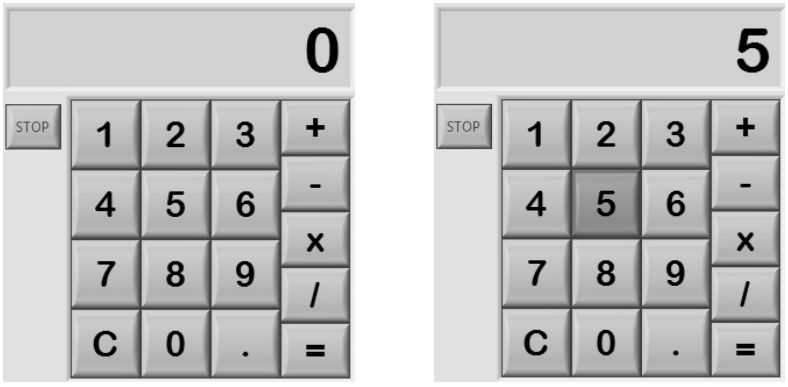
Gambar 6.28 (a) Tombol tidak ditekan, (b) Tombol angka 5 ditekan

13. Agar ketika tombol tidak ditekan, muncul angka 0, dan ketika tombol angka 5 ditekan, muncul angka 5, maka tambahkan fungsi **increment** (+1), yang dapat diambi di kategori **Numeric**.



Gambar 6.29 Menyisipkan fungsi increment untuk menambah 1 angka

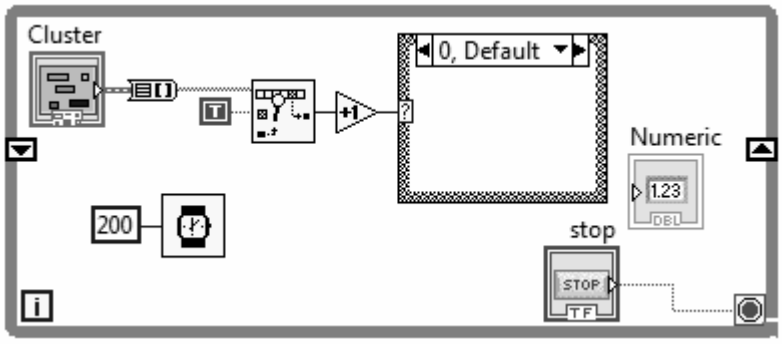
14. Jalankan program, maka ketika tombol tidak ditekan, muncul angka 0, dan ketika tombol 5 ditekan dan dilepas, muncul angka 5 sesaat.



(a) (b)

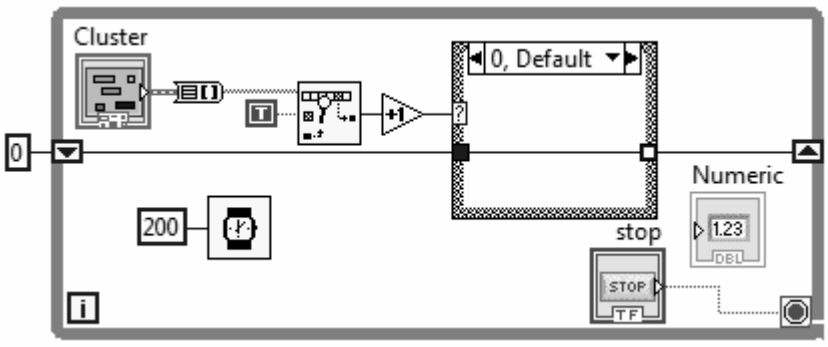
Gambar 6.30 (a) Tombol tidak ditekan, (b) Tombol angka 5 ditekan

15. Agar angka yang ditampilkan tidak hanya sesaat, tetapi dapat bertahan hingga tombol lain ditekan, maka tambahkan sebuah **Shift Register** dan sebuah struktur **Case**. Hubungkan output increment ke terminal **selector** struktur **Case,** maka secara otomatis Labelnya berubah dari **True/False** ke **0/1**, dengan nilai 0 sebagai **Default**.



Gambar 6.31 Menambahkan Shift Register dan struktur Case

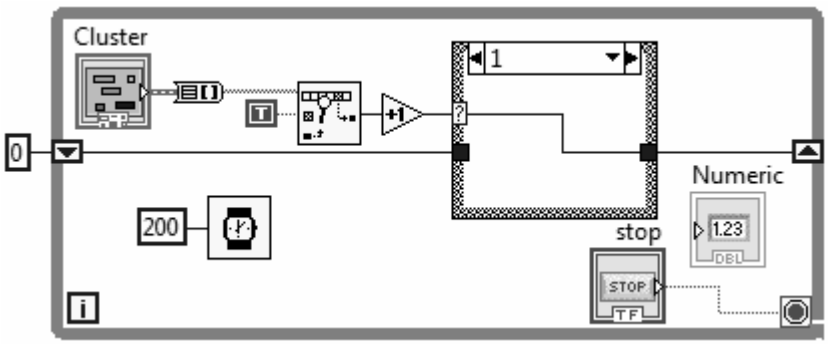
16. Beri nilai 0 sebagai nilai awalan **Shift Register**, dengan cara mengambil fungsi **Num Const** di kategori **Numeric** dan menghubungkannya ke terminal kiri **Shift Register**. Kemudian hubungkan output terminal kiri ini ke input terminal kanan **Shift Register** melalui struktur **Case** pada blok **Case 0** (Default). Hubungan ini berarti ketika tidak ada tombol yang ditekan (Default), maka data yang ada di loop sebelumnya akan digunakan untuk loop saat ini.



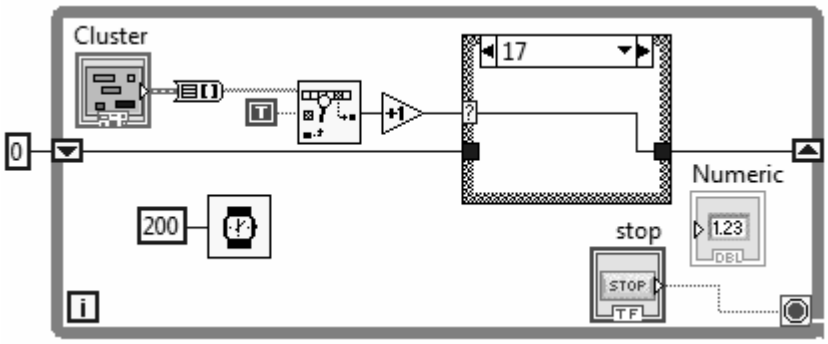
Gambar 6.32 Memberi nilai awal 0 dan mengisi nilai Shift Register saat Case Default dengan nilai data pada Loop sebelumnya

17. Perhatikan kotak terminal putih pada dinding kanan (output) struktur **Case**, yang menunjukkan bahwa ada blok yang outputnya tidak terhubung dengan kotak tersebut. Untuk itu, buka blok **Case 1**, dan hubungkan terminal kotak putih tersebut dengan terminal **selector**, yang berisi nilai tombol yang ditekan.

18. Karena ada 17 buah tombol, maka buat penambahan 16 blok lagi pada Struktur **Case**, dengan cara mengklik kanan dinding struktur **Case** dan memilih **Add Case After** pada menu popup yang muncul sebanyak 16 kali. Kemudian hubungkan terminal **selector** untuk blok **Case** 2 hingga 17 tersebut ke terminal kotak putih, sehingga terminal kotak tersebut berwarna penuh sesuai tipe datanya.



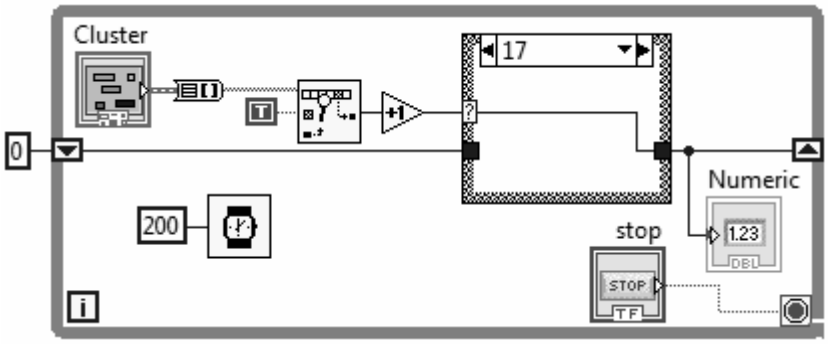
Gambar 6.33 Memberikan nilai terminal selector atau nilai tombol ke terminal kanan Shift Register pada saat blok Case 1



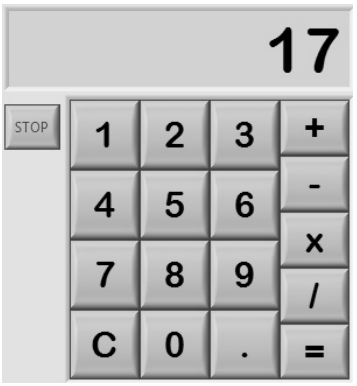
Gambar 6.34 Menambah blok Case dan mengulangi hal yang sama seperti pada blok Case 1 untuk blok Case 2 hingga blok Case 17

19. Hubungkan output struktur **Case** ke ikon **Num Ind** untuk menampilkan hasilnya. Jalankan program dan perhatikan tampilan kalkulator setiap kali tombol ditekan dan dilepas. Seharusnya tampilan tersebut akan bertahan hingga tombol berikutnya ditekan. Sampai di sini, program telah sesuai dengan cara kerja pertama.

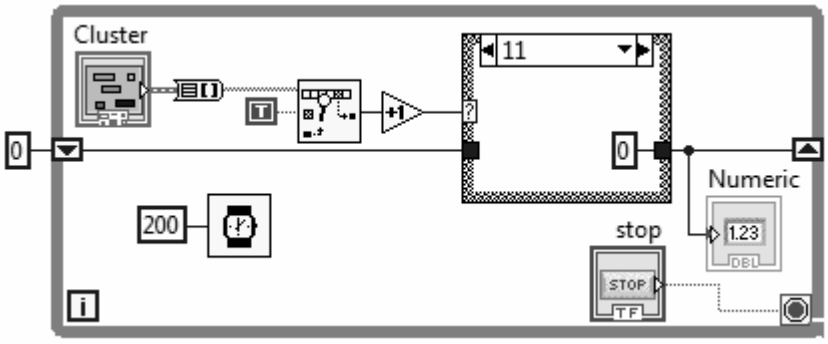
20. Sebelum melanjutkan ke cara kerja kalkulator kedua, perbaiki dulu tombol angka 0, di mana ketika tombol angka 0 ditekan, seharusnya ditampilkan angka 0, bukan angka 11. Hal ini dapat dilakukan dengan menghubungkan output blok Case 11 dengan nilai 0.



Gambar 6.35 Menghubungkan output struktur Case dengan Num Ind

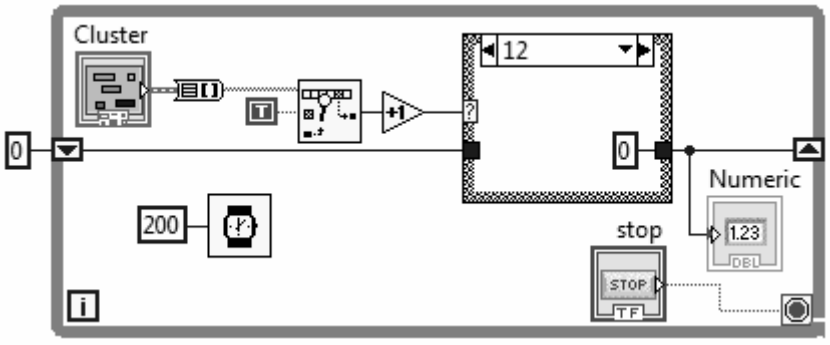


Gambar 6.36 Tampilan bertahan setelah tombol = ditekan dan dilepas



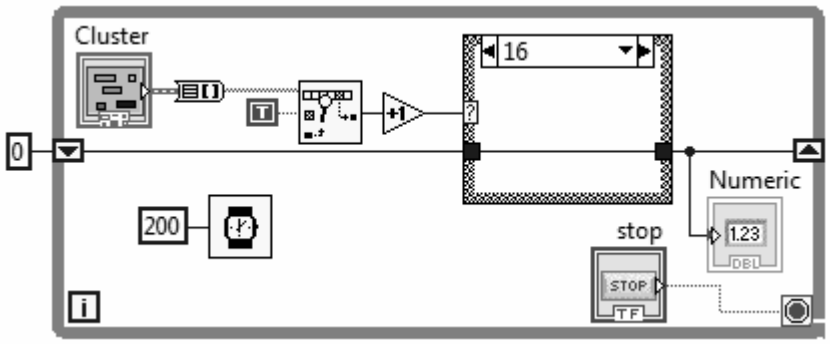
Gambar 6.37 Menghubungkan output blok Case 11 dengan nilai 0

21. Lakukan juga hal serupa dengan tombol C dan tombol . (titik), yaitu menghubungkan output blok Case 10 dan 12 dengan nilai 0.



Gambar 6.38 Menghubungkan output blok Case 12 dengan nilai 0

22. Berikutnya cara kerja kalkulator kedua, adalah membuat agar ketika tombol operator (+,-,x,/) ditekan, angka yang ditampilkan sebelumnya tidak berubah. Hal ini mudah saja dilakukan, yaitu dengan cara meniru isi blok Case 0 untuk blok Case 13, 14, 15, dan 16, yang merupakan blok Case untuk tombol operator (+,-,x,/).



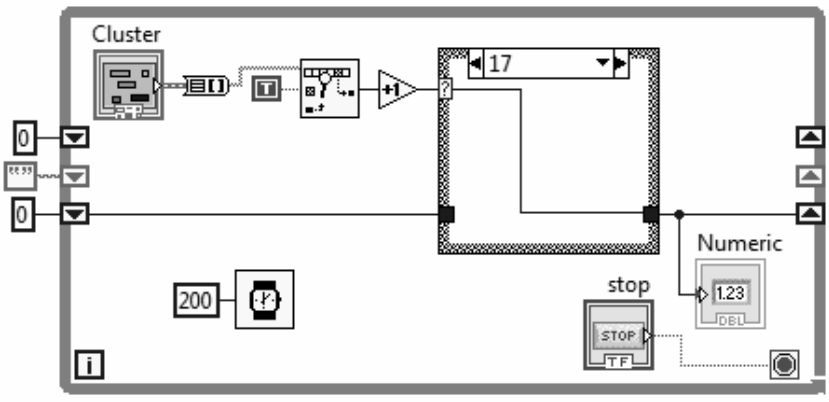
Gambar 6.39 Agar tampilan tidak berubah, blok Case 13, 14, 15 dan 16 meneruskan data Loop sebelumnya untuk data Loop berikutnya

23. Karena cara kerja kalkulator ketiga sama dengan cara kerja pertama, maka berikutnya adalah cara kerja keempat.

    Cara kerja keempat ini akan menampilkan hasil perhitungan ketika tombol (=) ditekan. Berarti ketika tombol (=) ini ditekan, blok Case 17 akan bekerja melakukan perhitungan dengan 3 input yang sudah tersedia, yaitu angka pertama, kedua, dan operator perhitungan.

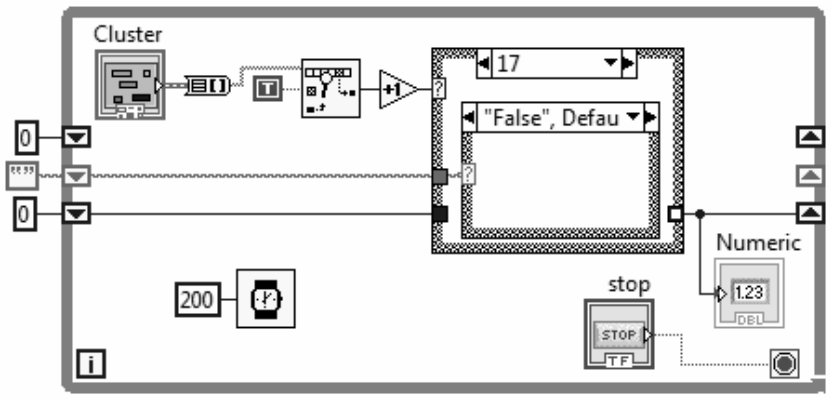
    Pada langkah inilah konsep **State Machine** diperlukan. Jadi pada kasus ini, ada 3 keadaan yang telah dibuat, yaitu: angka pertama, angka kedua, dan operator perhitungan yang dipilih. Sedangkan input pemicunya adalah tombol (=). Karena ketiga keadaan tersebut tidak dibuat pada saat tombol (=) ditekan, tetapi dibuat sebelumnya, maka Shift Register diperlukan. Karena Shift Register hanya bisa membawa satu jenis data, berarti ada 3 buah Shift Register yang diperlukan untuk membawa ketiga keadaan tersebut.

    Untuk itu, tambahkan 2 buah Shift Register lagi. Untuk Shift Register pertama, beri nilai awal dengan Num Const (angka 0), sedangkan untuk Shift Register kedua, beri nilai awal dengan Empty String.



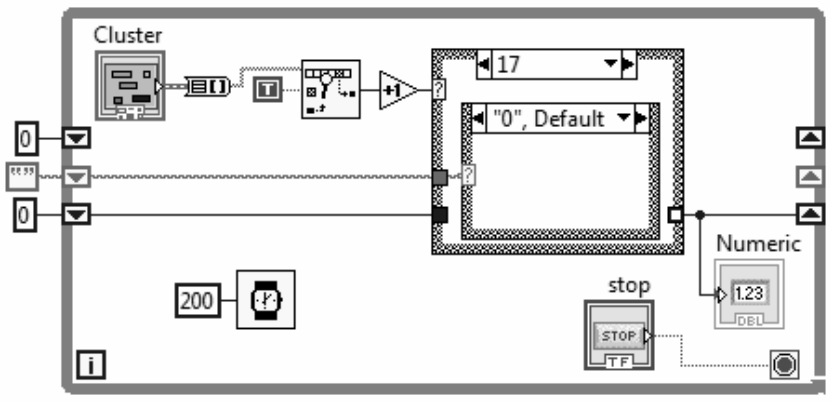
Gambar 6.40 Menambahkan 2 buah Shift Register dengan nilai awal Numeric Constant (angka 0) dan Empty String Constant

24. Pada blok Case 17, putus garis dari terminal selector ke output blok, dan kemudian tambahkan sebuah struktur Case di dalamnya. Hubungkan terminal selector struktur Case yang baru ini dengan terminal kiri Shift Register yang memiliki nilai awal Empty String.



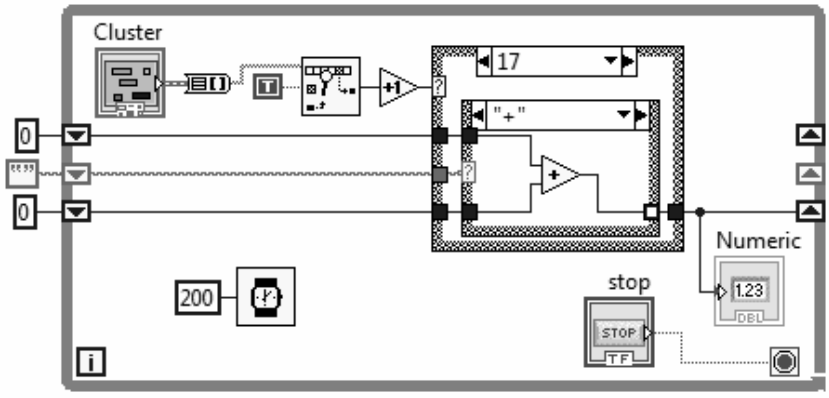
Gambar 6.41 Menambahkan struktur Case di dalam blok Case 17

25. Struktur Case yang baru ini digunakan untuk melakukan operasi perhitungan. Untuk itu ganti nama label “False”, Default dengan “0”, Default, dan blok berikutnya berlabel “True” dengan “+”.



Gambar 6.42 Mengganti label “False” dengan “0”, “True” dengan “+”

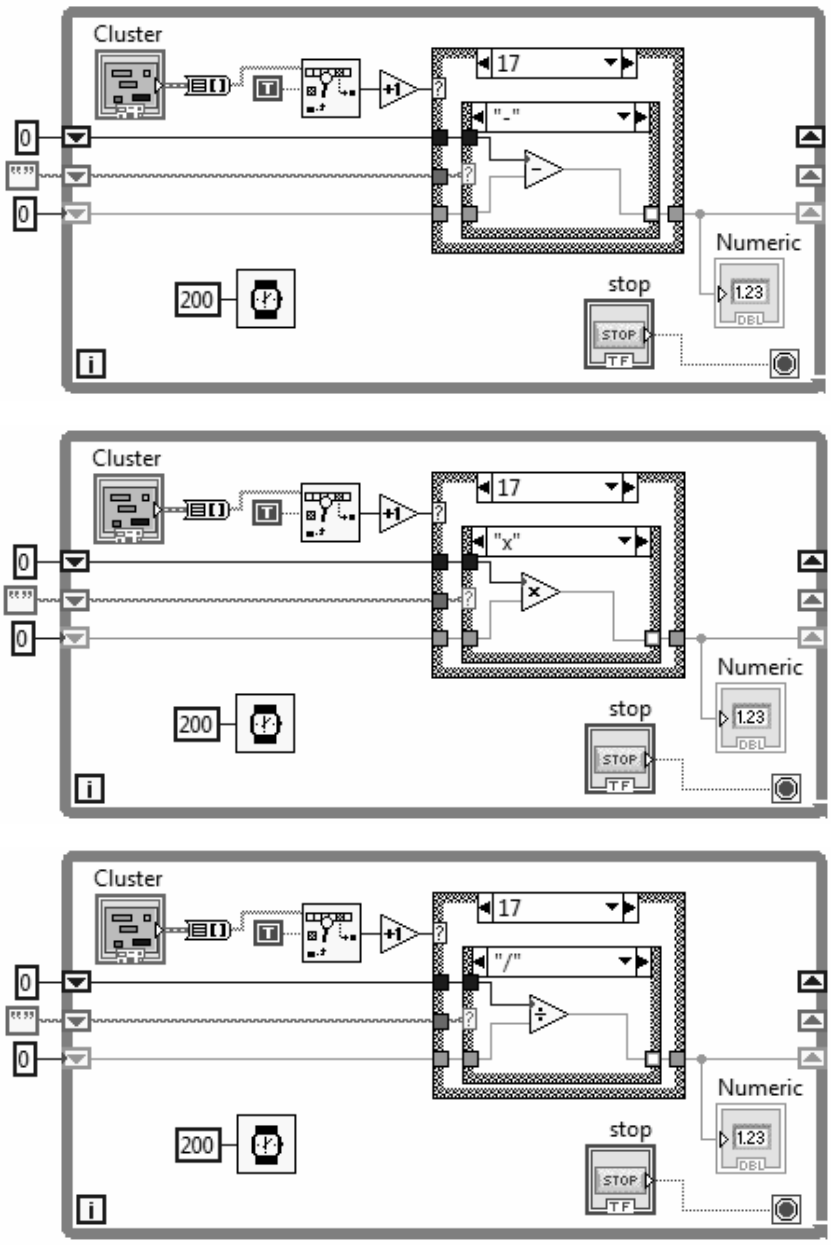
26. Kemudian pada blok **"+"**, tambahkan fungsi Add, di mana input pertama fungsi Add ini dihubungkan dengan terminal kiri Shift Register yang baru dengan nilai awal 0, dan input kedua fungsi Add ini dihubungkan dengan terminal kiri Shift Register yang lama. Output dari fungsi Add ini dihubungkan dengan output blok Case 17.



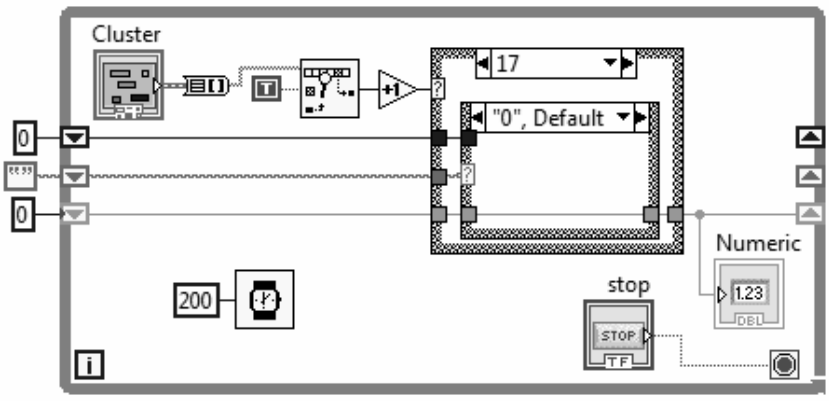
Gambar 6.43 Menambah fungsi Add di blok Case "+"

27. Tambahkan 3 buah blok Case lagi dengan mengklik kanan dinding struktur Case yang baru dan memilih **Add Case After** pada menu popup yang muncul. Beri label berturut-turut **"-"**, **"x"** dan **"/"** dan tambahkan fungsi **Subtract** pada blok **Case "-"**, fungsi **Multiply** pada blok **Case "x"**, fungsi **Divide** pada blok **Case "/"**. Kemudian hubungkan input dan output untuk ketiga fungsi tersebut sama seperti pada input dan output fungsi Add.

28. Perhatikan bahwa kotak output struktur Case yang baru masih berwarna putih, karena ada blok yang belum menghubungkannya, yaitu blok **Case "0", Default**. Untuk itu pada blok **Case "0", Default**, hubungkan kotak terminal putih tersebut dengan input dari Shift Register yang lama. Ini berarti ketika tidak ada tombol operator yang dipilih (atau Default), maka data sebelumnya akan diteruskan.

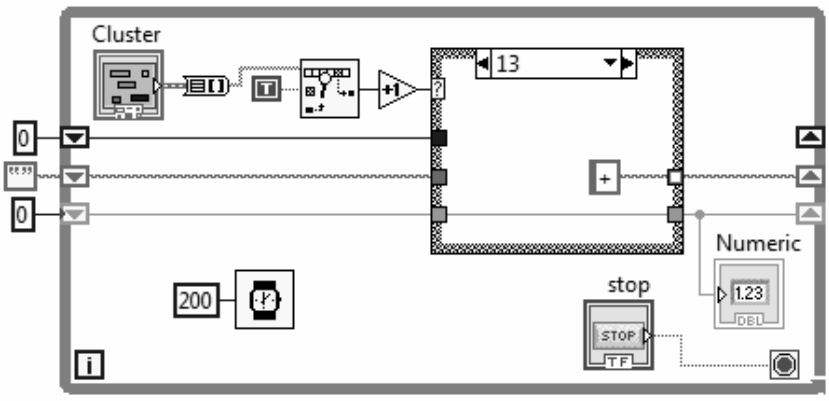


Gambar 6.44 Menambah fungsi Subtract di blok Case "-", fungsi Multiply di blok Case "x", dan fungsi Divide di blok Case "/"



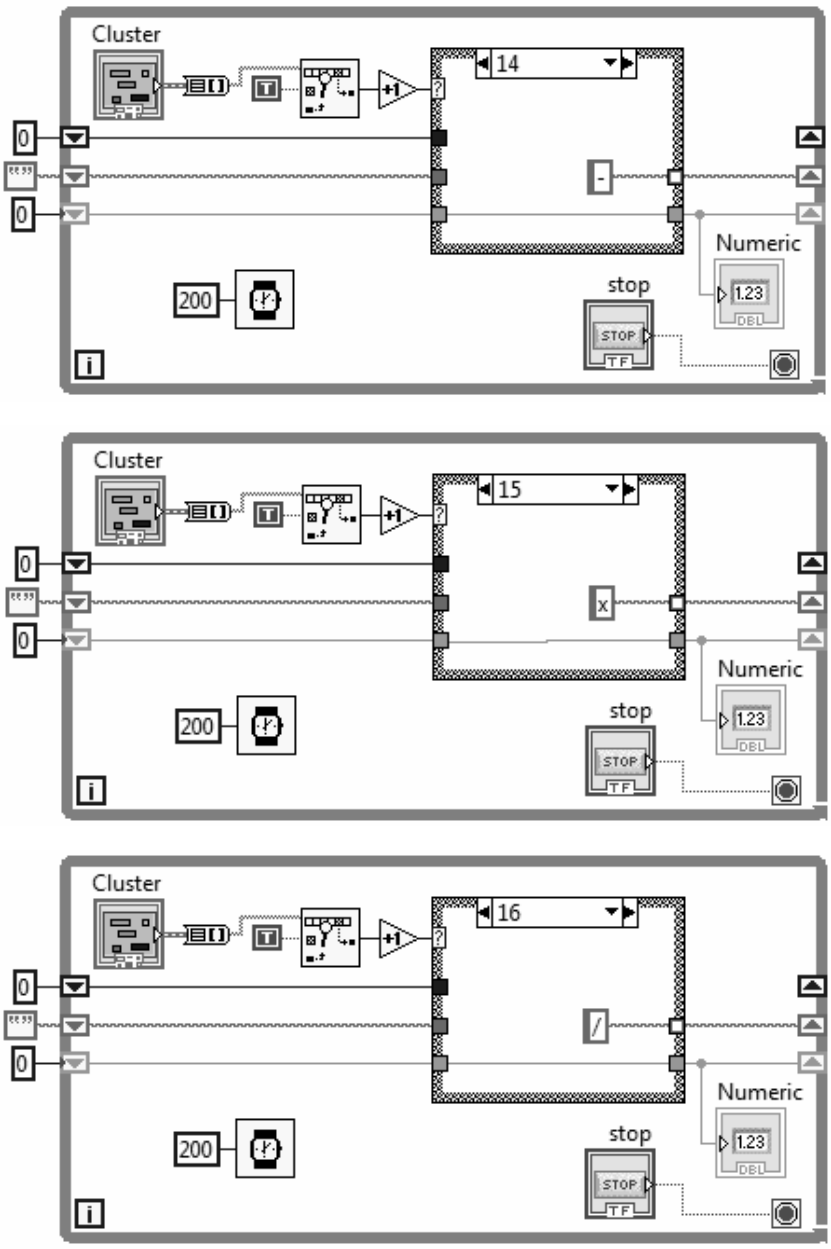
Gambar 6.45 Meneruskan data pada blok Case "0", Default

29. Karena Shift Register dengan nilai awal Empty String digunakan untuk membawa data jenis operator, maka tambahkan pada blok Case 13 yang merupakan blok untuk tombol +, sebuah string dengan karakter **+** yang dihubungkan dengan terminal kanan Shift Register.



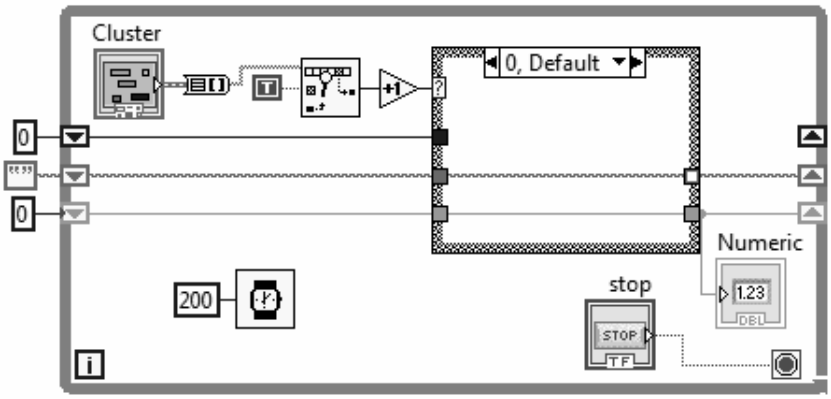
Gambar 6.46 Memberikan karakter + pada Shift Register ketika blok Case 13 bekerja atau apabila tombol + ditekan

30. Ulangi langkah di atas untuk blok Case 14, 15 dan 16 yang berturut-turut merupakan blok untuk tombol -, x dan /, yaitu dengan menghubungkan karakter -, x, dan / dengan Shift Register.

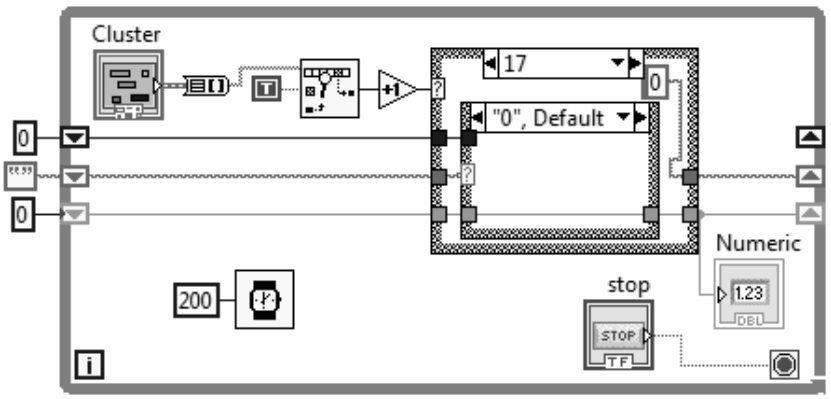


Gambar 6.47 Memberikan karakter -, x, atau / pada Shift Register apabila blok Case 14, 15 atau 16 bekerja, atau ketika tombol -,x atau / ditekan

31. Tampak bahwa kotak terminal operator masih berwarna putih, karena tidak semua blok terhubung dengan kotak terminal tersebut. Untuk itu hubungkan kotak terminal tersebut dengan input dari terminal kiri Shift Register yang membawa karakter operator, pada semua sisa blok Case, yaitu blok Case 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 dan 12. Khusus untuk blok Case 17, beri karakter 0.

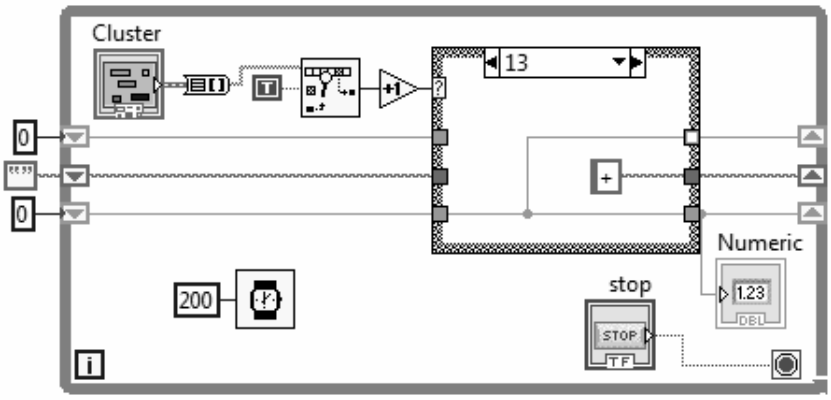


Gambar 6.48 Untuk blok Case 0 sampai 12, teruskan data dari terminal kiri ke terminal kanan Shift Register untuk membawa karakter operator



Gambar 6.49 Untuk blok Case 17, beri karakter 0 pada kotak terminal

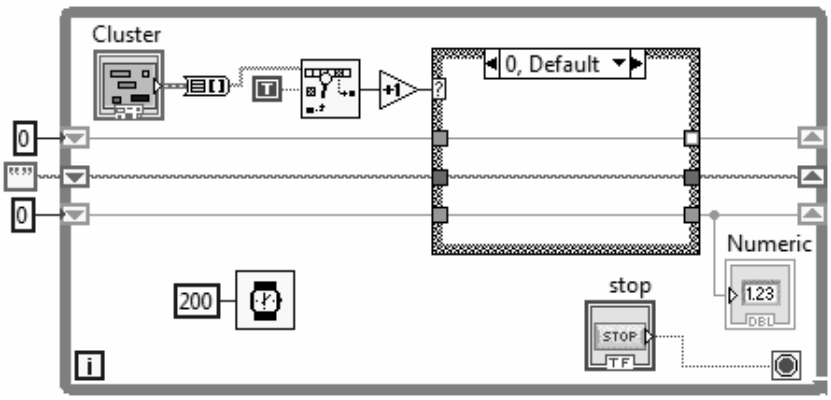
32. Kemudian ketika tombol operator ditekan, seharusnya data angka pertama tidak hanya diteruskan untuk ditampilkan, tetapi juga disimpan pada Shift Register yang baru, agar ketika tombol (=) ditekan, data angka pertama tersebut dapat diambil. Maka tambahkan garis data yang menghubungkan data dari Shift Register lama ke Shift Register baru untuk blok Case 13, 14, 15 dan 16.



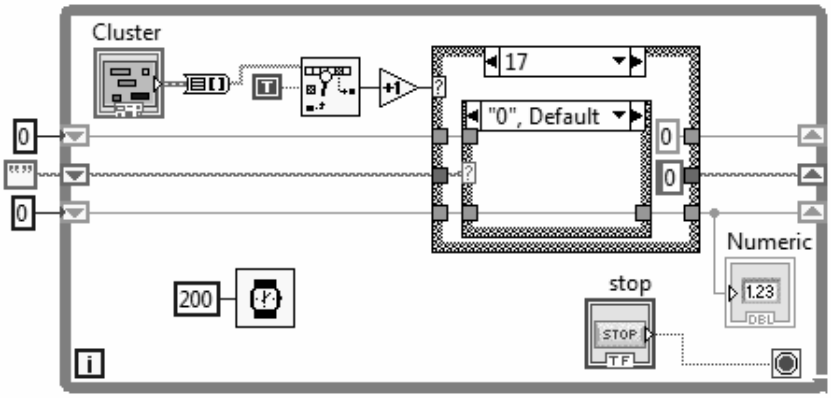
Gambar 6.50 Untuk Blok Case 13-16, hubungkan kedua Shift Register

33. Tampak kotak masih berwarna putih. Untuk itu, pada blok Case 0-12, hubungkan input dari terminal kiri Shift Register baru ke kotak terminal putih, untuk meneruskan data ketika tidak ada tombol operator yang ditekan. Khusus untuk blok Case 17, beri angka 0.

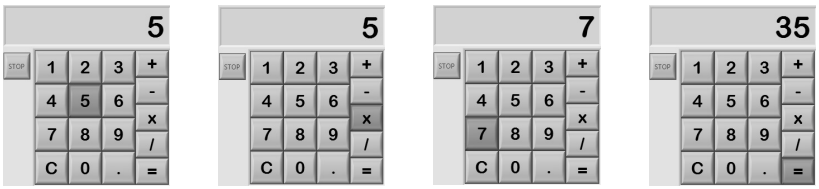
34. Jalankan program, maka simulasi kalkulator akan bekerja sesuai cara kerja yang diminta. Namun demikian, masih banyak keterbatasan dalam program ini, salah satunya adalah hanya bisa memasukkan 1 digit angka saja, yaitu angka antara 0-9. Begitu pula angka pecahan belum bisa, karena tombol titik belum berfungsi. Bab 8 akan memberikan solusi untuk permasalahan ini. Terlepas dari semua keterbatasan itu, pembaca telah mempelajari konsep **State Machine**, yang banyak dipakai dalam pemrograman LabVIEW.



Gambar 6.51 Untuk blok Case 0-12, teruskan data Shift Register



Gambar 6.52 Untuk Blok Case 17, beri angka 0 pada Shift Register



(a) tombol 5 (b) tombol x (c) tombol 7 (d) tombol =

Gambar 6.53 Program dijalankan untuk menghitung 5x7