به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

شبیه سازی شی گرای سیستم های الکترونیکی

تمرین شماره ۱

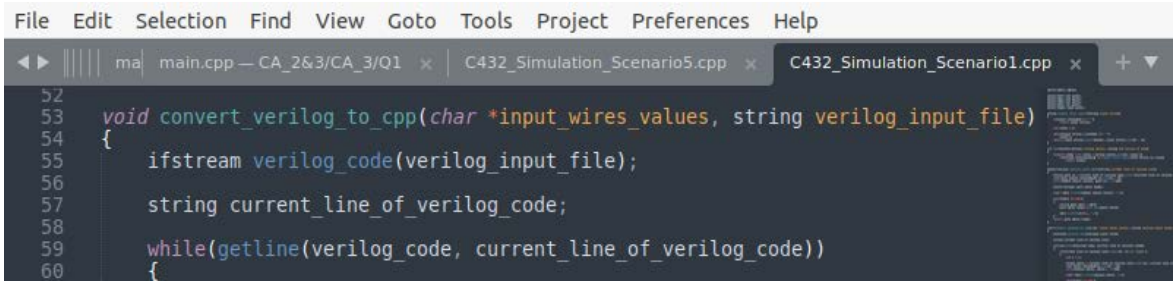
محمد تقی زاده گیوری

۸۱۰۱۹۸۳۷۳

زمستان ۱۴۰۱

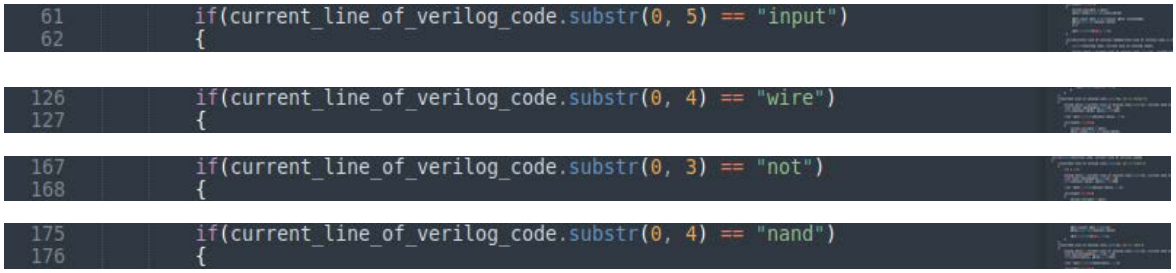
بخش A

ابتدا در یک حلقه با استفاده از دستور getline()، خط به خط کد وریلاگ داده شده را می خوانیم:

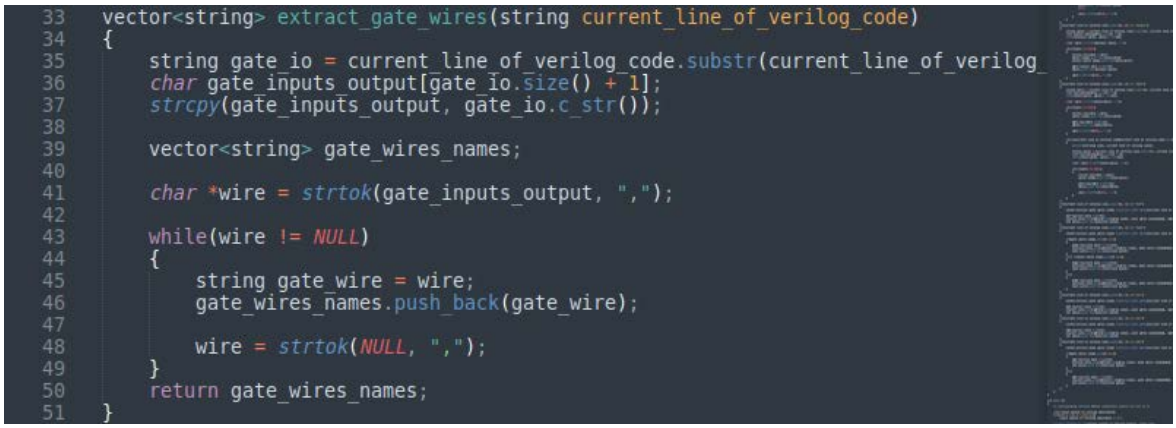


بر اساس ابتدا خط خوانده شده، میفهمیم که در این خط از کد وریلاگ wire، گیت not، and یا...

تعریف شده است:



سپس برای بدست آوردن اسامی سیم های تعریف شده یا سیم های ورودی و خروجی گیت ها، ادامه خط خوانده شده را با استفاده از تابع substr() در نظر گرفته و سپس با استفاده از تابع strtok()، اسامی که با حرف ‘,’ از هم جدا شدند (که اسامی سیم ها یا سیم های ورودی خروجی گیت ها هستند) را بدست می آوریم:



حال هم نوع گیت بدست آمده است و هم اسامی سیم های ورودی خروجی گیت بدست آمده است،

پس از گیت متناظر یک instance گرفته و سیم های ورودی خروجی آن را بر اساس اسامی

سیم های ورودی و خروجی گیت که بدست آوردیم، مقداردهی می کنیم:

```
171             NOT new_not_gate = NOT();
172             new_not_gate.ios(&Wires[find(wires_names, gate_wires_names[0])], &Wir
173             not_gates.push_back(new_not_gate);
```

```
179             if(gate_wires_names.size() == 3)
180             {
181                 NAND new_nand_gate = NAND(2);
182                 new_nand_gate.ios(&Wires[find(wires_names, gate_wires_names[0])],
183                 nand_gates.push_back(new_nand_gate);
184             }
185             else if(gate_wires_names.size() == 4)
186             {
187                 NAND new_nand_gate = NAND(3);
188                 new_nand_gate.ios(&Wires[find(wires_names, gate_wires_names[0])],
189                 nand_gates.push_back(new_nand_gate);
190             }
191             else
192             {
193                 NAND new_nand_gate = NAND(4);
194                 new_nand_gate.ios(&Wires[find(wires_names, gate_wires_names[0])],
195                 nand_gates.push_back(new_nand_gate);
196             }
```

این کار را آنقدر ادامه می دهیم تا به خطی برسیم که در انتها آن ’;‘ باشد که به معنای اتمام دستور

می باشد:

```
144
145 ile(current_line_of_verilog_code[current_line_of_verilog_code.size() - 2] != ';')
146
```

در آخر یک سری wire، گیت not، and و... بدست می آید که گیت های کد وریلاگ داده شده است

و در vector ای از wire، گیت not، and و... ذخیره می شوند:

```
4   vector<string> wires_names;
5   vector<string> output_wires_names;
6   vector<Wire> Wires;
7
8   vector<NOT> not_gates;
9   vector<AND> and_gates;
10  vector<XOR> xor_gates;
11  vector<NOR> nor_gates;
12  vector<NAND> nand_gates;
```

بخش B

بعد از تبدیل کردن کد وریلاگ داده شده به لیستی از گیت هایی که توسط C++ شناخته شده هستند. با فرخوانی متد(method) evl برای هر کدام از گیت های ساخته شده، کد وریلاگ داده شده شبیه سازی می شود. فراخوانی متد evl می تواند به ترتیب های مختلفی صورت گیرد که ۵ سناریو از ترتیب های مختلف آن در زیر بیان شده است:

سناریو ۱

```
234  int main ()
235  {
236      // initializing verilog module inputs(all inputs are set to 1)
237
238      char input_values_of_verilog_module[36];
239      for(int i = 0; i < 36; i++)
240          input_values_of_verilog_module[i] = '1';
241
242      convert_verilog_to_cpp(input_values_of_verilog_module, "c432.v");
243
244      // gates evaluation in random order(scenario 1)
245
246      for(int i = 0; i < not_gates.size(); i++)
247          not_gates[i].evl();
248      for(int i = 0; i < and_gates.size(); i++)
249          and_gates[i].evl();
250      for(int i = 0; i < xor_gates.size(); i++)
251          xor_gates[i].evl();
252      for(int i = 0; i < nor_gates.size(); i++)
253          nor_gates[i].evl();
254      for(int i = 0; i < nand_gates.size(); i++)
255          nand_gates[i].evl();
256
257      // printing the values of output wires
258
259      for(int i = 0; i < output_wires_names.size(); i++)
260          cout << output_wires_names[i] << " = " << Wires[find(wires_names,
261
262      return 1;
```

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_A_to_B$ ./a.out
N223 = X
N329 = X
N370 = X
N421 = X
N430 = X
N431 = X
N432 = X
```

همان طور که میبینید از جایی که C++ به صورت procedural است، در صورتی خروجی های مدار درست خواهند بود، که، ترتیب فراخوانی متد evl گیت ها، با ترتیب خاصی باشد.

در نتیجه چون این ترتیب خاص در این سناریو رعایت نشده، خروجی ها درست ساخته نشده

و در نتیجه بعضی از خروجی ها مقدار شان نامعلوم (X) می شود.

سناریو ۲

```cpp
int main ()
{
    // initializing verilog module inputs(all inputs are set to 1)

    char input_values_of_verilog_module[36];
    for(int i = 0; i < 36; i++)
        input_values_of_verilog_module[i] = '1';

    convert_verilog_to_cpp(input_values_of_verilog_module, "c432.v");

    // gates evaluation in random order(scenario 2)

    for(int i = 0; i < and_gates.size(); i++)
        and_gates[i].evl();
    for(int i = 0; i < not_gates.size(); i++)
        not_gates[i].evl();
    for(int i = 0; i < xor_gates.size(); i++)
        xor_gates[i].evl();
    for(int i = 0; i < nor_gates.size(); i++)
        nor_gates[i].evl();
    for(int i = 0; i < nand_gates.size(); i++)
        nand_gates[i].evl();

    // printing the values of output wires

    for(int i = 0; i < output_wires_names.size(); i++)
        cout << output_wires_names[i] << " = " << Wires[find(wires_names,

    return 1;
```

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_A_to_B$ ./a.out
N223 = X
N329 = X
N370 = X
N421 = X
N430 = X
N431 = X
N432 = X
```

همان طور که میبینید از جایی که C++ به صورت procedural است، در صورتی خروجی های مدار درست خواهند بود، که، ترتیب فراخوانی متد evl گیت ها، با ترتیب خاصی باشد.

در نتیجه چون این ترتیب خاص در این سناریو رعایت نشده، خروجی ها درست ساخته نشده و در نتیجه بعضی از خروجی ها مقدار شان نامعلوم (X) می شود.

سناریو ۳

```cpp
int main ()
{
    // initializing verilog module inputs(all inputs are set to 1)

    char input_values_of_verilog_module[36];
    for(int i = 0; i < 36; i++)
        input_values_of_verilog_module[i] = '1';

    convert_verilog_to_cpp(input_values_of_verilog_module, "c432.v");

    // gates evaluation in random order(scenario 3)

    for(int i = 0; i < and_gates.size(); i++)
        and_gates[i].evl();
    for(int i = 0; i < xor_gates.size(); i++)
        xor_gates[i].evl();
    for(int i = 0; i < not_gates.size(); i++)
        not_gates[i].evl();
    for(int i = 0; i < nor_gates.size(); i++)
        nor_gates[i].evl();
    for(int i = 0; i < nand_gates.size(); i++)
        nand_gates[i].evl();

    // printing the values of output wires

    for(int i = 0; i < output_wires_names.size(); i++)
        cout << output_wires_names[i] << " = " << Wires[find(wires_names,

    return 1;
```

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_A_to_B$ ./a.out
N223 = X
N329 = X
N370 = X
N421 = X
N430 = X
N431 = X
N432 = X
```

همان طور که میبینید از جایی که C++ به صورت procedural است، در صورتی خروجی های مدار درست خواهند بود، که، ترتیب فراخوانی متد evl گیت ها، با ترتیب خاصی باشد.

در نتیجه چون این ترتیب خاص در این سناریو رعایت نشده، خروجی ها درست ساخته نشده و در نتیجه بعضی از خروجی ها مقدار شان نامعلوم (X) می شود.

سناریو ۴

```
int main ()
{
    // initializing verilog module inputs(all inputs are set to 1)

    char input_values_of_verilog_module[36];
    for(int i = 0; i < 36; i++)
        input_values_of_verilog_module[i] = '1';

    convert_verilog_to_cpp(input_values_of_verilog_module, "c432.v");

    // gates evaluation in random order(scenario 4)

    for(int i = 0; i < not_gates.size(); i++)
        not_gates[i].evl();
    for(int i = 0; i < nor_gates.size(); i++)
        nor_gates[i].evl();
    for(int i = 0; i < nand_gates.size(); i++)
        nand_gates[i].evl();
    for(int i = 0; i < xor_gates.size(); i++)
        xor_gates[i].evl();
    for(int i = 0; i < and_gates.size(); i++)
        and_gates[i].evl();

    // printing the values of output wires

    for(int i = 0; i < output_wires_names.size(); i++)
        cout << output_wires_names[i] << " = " << Wires[find(wires_names,

    return 1;
```

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_A_to_B$ ./a.out
N223 = X
N329 = X
N370 = X
N421 = X
N430 = X
N431 = X
N432 = X
```

همان طور که میبینید از جایی که C++ به صورت procedural است، در صورتی خروجی های مدار درست خواهند بود، که، ترتیب فراخوانی متد evl گیت ها، با ترتیب خاصی باشد.

در نتیجه چون این ترتیب خاص در این سناریو رعایت نشده، خروجی ها درست ساخته نشده و در نتیجه بعضی از خروجی ها مقدار شان نامعلوم (X) می شود.

سناریو ۵

```cpp
int main ()
{
    // initializing verilog module inputs(all inputs are set to 1)

    char input_values_of_verilog_module[36];
    for(int i = 0; i < 36; i++)
        input_values_of_verilog_module[i] = '1';

    convert_verilog_to_cpp(input_values_of_verilog_module, "c432.v");

    // gates evaluation in random order(scenario 5)

    for(int i = 0; i < 18; i++)
        not_gates[i].evl();
    nand_gates[0].evl();
    nor_gates[0].evl();
    nor_gates[1].evl();
    for(int i = 1; i < 9; i++)
        nand_gates[i].evl();
    for(int i = 2; i < 18; i++)
        nor_gates[i].evl();
    and_gates[0].evl();
    for(int i = 18; i < 21; i++)
        not_gates[i].evl();
    for(int i = 0; i < 6; i++)
        xor_gates[i].evl();
    nand_gates[9].evl();
    xor_gates[6].evl();
    nand_gates[10].evl();
```

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_A_to_B$ ./a.out
N223 = 0
N329 = 0
N370 = 0
N421 = 0
N430 = 1
N431 = 1
N432 = 1
```

در این سناریو بر خلاف سناریو های قبلی، ترتیب گیت ها را مطابق با ترتیب گیت ها، در کد وریلاگ داده شده، در نظر گرفتیم. درنتیجه از جایی که ترتیب فراخوانی گیت ها مطابق با ترتیب درستشان است، این دفعه خروجی های درست دیده می شوند.

بخش C

ابتدا در class گیت ها، یک flag به نام changed_flag تعریف می کنیم. که بیان گر این است که خروجی گیت نسبت به آخرین مقدار خود(last_value) تغییر کرده است یا خیر:

```
10        int number_of_inputs, changed_flag;
11        char last_value = 'X';
```

سپس در متد evl تمامی گیت ها، پس از محاسبه شدن خروجی گیت، ابتدا بررسی می کنیم که آیا خروجی محاسبه شده متفاوت از آخرین مقدار خروجی(last_value) هست یا خیر، در صورتی که خروجی محاسبه شده متفاوت از خروجی قبلی گیت باشد، درنتیجه خروجی گیت تغییر کرده است و مقدار changed_flag را برابر با 1 می کنیم و در غیر این صورت خروجی گیت تغییر نکرده است و مقدار changed_flag را برابر با 0 قرار می دهیم:

```
95        changed_flag = (o1->get_value() != last_value) ? 1 : 0;
```

در انتها، آخرین مقدار خروجی(last_value) را آپدیت کرده و برابر با خروجی محاسبه شده در متد evl قرار می دهیم.

```
96        last_value = o1->get_value();
```

بخش D

در یک حلقه متد evl گیت ها را فرخوانی می کنیم.

گیت هایی که ورودی آن ها آماده است، با فراخوانی evl، خروجی شان محاسبه شده و از X به 0 یا 1 تغییر می کند. درنتیجه changed_flag آن ها برابر با 1 می شود.

سپس در تکرار بعدی حلقه، ورودی برخی از گیت ها (که در واقع خروجی گیت هایی بود که در تکرار قبلی حلقه محاسبه شد) آماده شده است و با فراخوانی evl خروجی برخی دیگر از گیت ها هم محاسبه می شود. در نتیجه پس از مدتی ورودی تمامی گیت ها آماده شده و خروجی گیت ها و مدار محاسبه می شود. در این حالت با فراخوانی evl مقدار خروجی گیت، دوباره همان مقدار قبلی شده و تغییر نمی کند. در نتیجه changed_flag برابر با 0 می شود.

بنابر توضیحات فوق در یک حلقه آنقدر متد evl گیت ها را فراخوانی می کنیم تا changed_flag تمامی گیت ها 0 شود و همه گیت ها خروجی شان محاسبه شده باشد.

بخش E

سناریو ۱

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_C_to_E$ ./a.out
N223 = 0
N329 = 0
N370 = 0
N421 = 0
N430 = 1
N431 = 1
N432 = 1
```

همان طور که مشاهده می کنید همان ترتیبی که در قسمت B برخی خروجی ها را برابر با X نشان داد در اینجا دیگر ترتیب مهم نبوده و خروجی های درست، بر اساس ورودی(که تمامش را برابر با 1 کردیم) محاسبه شده است.

سناریو ۲

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_C_to_E$ ./a.out
N223 = 0
N329 = 0
N370 = 0
N421 = 0
N430 = 1
N431 = 1
N432 = 1
```

همان طور که مشاهده می کنید همان ترتیبی که در قسمت B برخی خروجی ها را برابر با X نشان داد در اینجا دیگر ترتیب مهم نبوده و خروجی های درست، بر اساس ورودی(که تمامش را برابر با 1 کردیم) محاسبه شده است.

سناریو ۳

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_C_to_E$ ./a.out
N223 = 0
N329 = 0
N370 = 0
N421 = 0
N430 = 1
N431 = 1
N432 = 1
```

همان طور که مشاهده می کنید همان ترتیبی که در قسمت B برخی خروجی ها را برابر با X نشان داد در اینجا دیگر ترتیب مهم نبوده و خروجی های درست، بر اساس ورودی(که تمامش را برابر با 1 کردیم) محاسبه شده است.

سناریو ۴

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_C_to_E$ ./a.out
N223 = 0
N329 = 0
N370 = 0
N421 = 0
N430 = 1
N431 = 1
N432 = 1
```

همان طور که مشاهده می کنید همان ترتیبی که در قسمت B برخی خروجی ها را برابر با X نشان داد در اینجا دیگر ترتیب مهم نبوده و خروجی های درست، بر اساس ورودی(که تمامش را برابر با 1 کردیم) محاسبه شده است.

سناریو ۵

```
ariyataghizadeh@ariyataghizadeh-VirtualBox:~/Object_Oriented - Navabi/Home Work
 1/Part_C_to_E$ ./a.out
N223 = 0
N329 = 0
N370 = 0
N421 = 0
N430 = 1
N431 = 1
N432 = 1
```

همان طور که مشاهده می کنید خروجی ها برابر با خروجی هایی شد که هنگام اجرا با ترتیب درست در بخش B دریافت کردیم که نشان از درست بودن خروجی نشان داده شده در این حالت است.