به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

شبیه سازی شی گرای سیستم های الکترونیکی

CA2

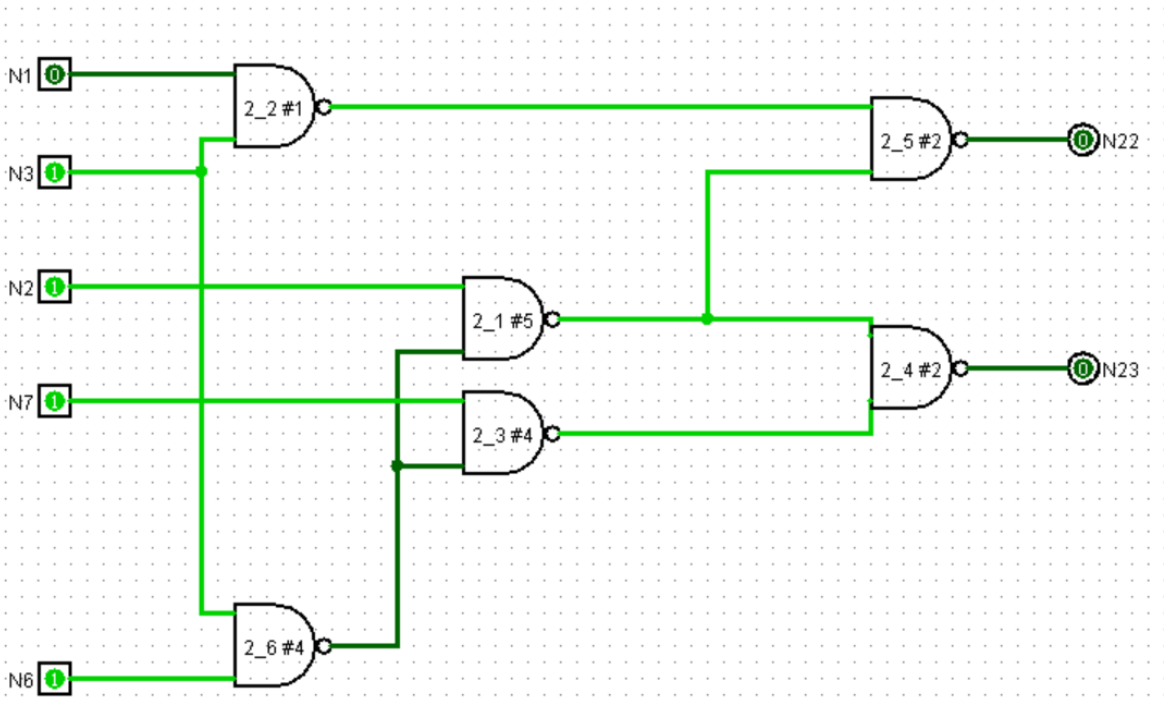
علیرضا جابری راد

810196438

بهار 1401

بخش A

مدار در نظر گرفته شده در این پروژه مطابق تصویر زیر است. تاخیر هر یک از گیت ها کنار نام آن ها پس از علامت # ذکر شده است:



دیاگرام بالا در نرم افزار Logisim رسم شده است و مقادیر مورد انتظار در خروجی را طبق همین مدل، شبیه سازی کردم.

بخش B

انتظار میرود، تاخیر در بدترین حالت 11 واحد زمانی باشد (برای هر دو خروجی). با شروع از سیم N6 و عبور از گیت های 2_6 و 2_1 مجموعا 9 واحد زمانی تاخیر داریم و در گیت متصل به خروجی ها هم 2 واحد زمانی تاخیر داریم، یعنی برای هر دو خروجی، بدترین حالت تاخیر، 11 واحد زمانی است. در تصویر زیر اجرای کد تشخیص دهنده بدترین تاخیر برای خروجی ها را مشاهده میکنید که به وضوح پاسخ آن درست است:

```
worst case delay of N22 is: 11
The path is: N6   NAND2_6   N11   NAND2_1   N16   NAND2_5   N22


worst case delay of N23 is: 11
The path is: N6   NAND2_6   N11   NAND2_1   N16   NAND2_4   N23
```

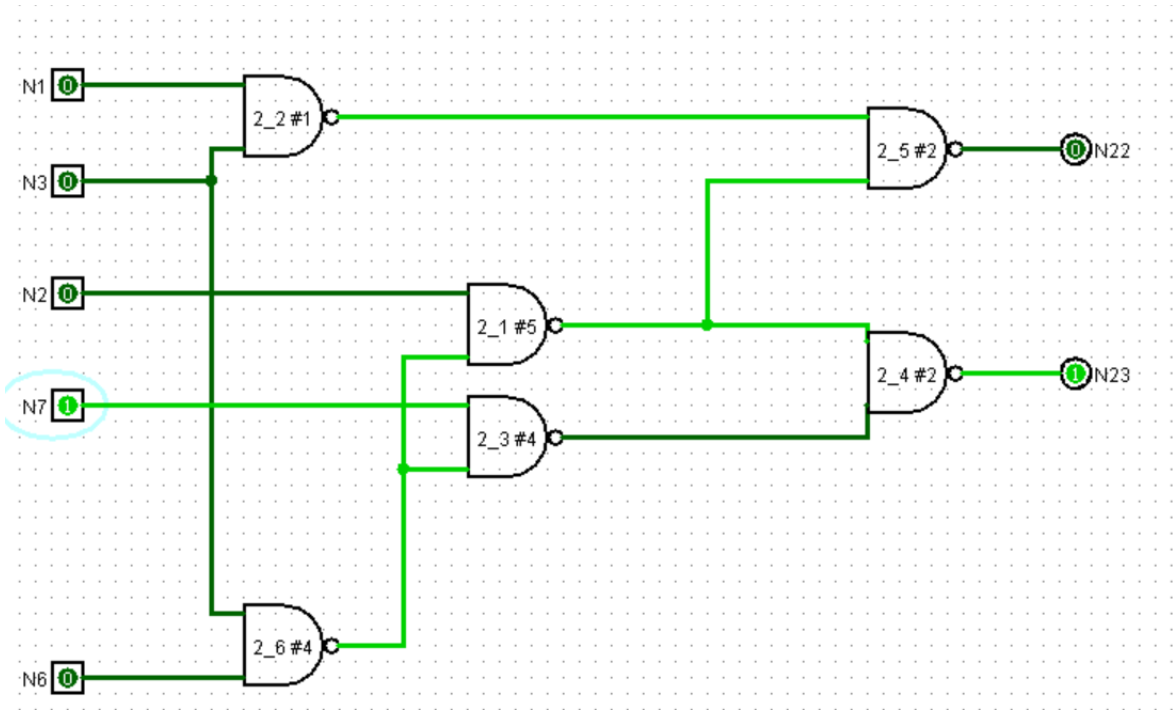
کد این بخش به صورت بازگشتی نوشته شده، یعنی از خروجی شروع کرده و با پیمایش گره های ورودی هر گیت و ماکسیمم گیری از تاخیر آن ها، تاخیر خروجی ها مشخص شده و طی این محاسبات، یک وکتور از مسیری که باید طی شود تا بدترین تاخیر ایجاد شود هم گزارش میشود. تصویر کد مربوطه را در زیر مشاهده میکنید:

```
void findCriticalPath(vector<string> &criticalPath, vector<string> &intputs, vector<string> &outputs, map<string,gates*> &GATES, map<string,wire> &wires){
    int worstCaseDelay=0;
    for(int i=0; i<outputs.size(); i++){
        cout<<"\n\nworst case delay of "<<outputs[i]<<" is: "<<wireCriticalPath(criticalPath, wires[outputs[i]], wires, intputs, GATES)<<endl;
        cout<<"The path is: ";
        for(int i=criticalPath.size()-1; i>=0; i--)
            cout<<criticalPath[i]<<"   ";
        cout<<endl;
        criticalPath.clear();
    }
}

int wireCriticalPath(wire WIRE, map<string,wire> &wires, vector<string>&inputs, map<string,gates*> &GATES){
    gates* gate;
    if(find(inputs.begin(), inputs.end(), findWireName(WIRE,wires))==inputs.end()){
    gate = findGateFromOutputWire(WIRE, GATES);
    return gate->delay()+MAX(wireCriticalPath(gate->in1(),wires,inputs,GATES), wireCriticalPath(gate->in2(),wires,inputs,GATES));
    }
    else
        return 0;
}

int wireCriticalPath(vector<string>&criticalPath, wire WIRE, map<string,wire> &wires, vector<string>&inputs, map<string,gates*> &GATES){
    gates* gate;
    criticalPath.push_back(findWireName(WIRE,wires));
    if(find(inputs.begin(), inputs.end(), findWireName(WIRE,wires))==inputs.end()){
    gate = findGateFromOutputWire(WIRE, GATES);
    criticalPath.push_back(findGateName(gate, GATES));
    if(wireCriticalPath(gate->in1(),wires,inputs,GATES) > wireCriticalPath(gate->in2(),wires,inputs,GATES))
        return gate->delay()+wireCriticalPath(criticalPath,gate->in1(),wires,inputs,GATES);
    else
        return gate->delay()+wireCriticalPath(criticalPath,gate->in2(),wires,inputs,GATES);
    }
    else
        return 0;
}
```

شبیه سازی بر روی این مدار، بر اساس ورودی زیر صورت گرفته است:

```
#00 00000
#30 00001
#15 00010
#20 00110
#25 00111
#25 01111
```

طبق نتایج بدست آمده از شبیه سازی در نرم افزار Logisim، انتظار میرود به ازای همه ی ورودی ها به جز ورودی 00001، خروجی ها صفر باشند. خروجی به ازای ورودی 00001 انتظار میرود به ترتیب زیر باشد:



خروجی حاصل از شبیه سازی، علاوه بر کنسول(در کنسول جزئیات بیشتری قید شده، هم نام گره ها و هم ورودی اعمالی که در فایل تکست فقط خروجی ها به ترتیب alphanumerical ذکر شده اند)، در یک فایل تکست به نام simulationResult هم گزارش میشود که خروجی فایل تکست را در تصویر زیر مشاهده میفرمایید(time_resolution = 5):

```
At 0: 00
At 5: 00
At 10: 00
At 15: 00
At 20: 00
At 25: 00
At 30: 01
At 35: 01
At 40: 01
At 45: 00
At 50: 00
At 55: 00
At 60: 00
At 65: 00
At 70: 00
At 75: 00
At 80: 00
At 85: 00
At 90: 00
At 95: 00
At 100: 00
At 105: 00
At 110: 00
```

همانطور که میبینید، نتایج حاصل از شبیه سازی مطابق انتظار بوده و فقط به ازای ورودی 00001 خروجی ما 00 نبوده و خروجی 01، یعنی سیم N23 یک و N22 صفر بوده است.

نکته: اگر میخواهید خودتان شبیه سازی کنید، پیش از اجرای برنامه فایل simulationResult.txt را پاک کنید.

بخش C

شبیه سازی تست بنچ با استفاده از کد زیر انجام شده است که در آن، تابع simulate نقش اصلی مدیریت اعمال ورودی به مدار را بر عهده دارد:

```
string line;
int targetTime;
fstream TBFile;
TBFile.open("input.txt");
string inputValuesStr="XXXXX";
while(!TBFile.eof()){
    getline(TBFile, line);
    regex ex("^[ ]*[#]([[:w:]]+)[ ]+([[:w:]]+)");
    smatch match;
    regex_search(line,match,ex);

    targetTime+=stoi(match[1]);
    for(;present_time<targetTime;present_time+=time_resolution)
        simulate(present_time,inputValuesStr, inputs, outputs, GATES, allWires);
    inputValuesStr=match[2];
}
```

در تصویر زیر بدنه ی تابع simulate را مشاهده میکنید(simulate، ورودی ها و present_time آن ها را ست میکند و evl گیت ها را به ترتیب Level اشان فراخوانی میکند و نتایج را در فایل تکست و کنسول نمایش میدهد):

```
void simulate(int present_time, string str, vector<string> &inputs, vector<string> &outputs, map<string,gates*> &GATES, map<string,wire> &wires){
    for(int i=0; i<str.size(); i++){
        wires[inputs[i]].put(str[i],present_time);
    }

    int gatesEncountered=0;
    int levelEncountered=0;
    while(gatesEncountered!=GATES.size()){
        levelEncountered++;
        for(auto it=GATES.begin(); it!=GATES.end(); ++it){
            if(it->second->Level == levelEncountered){
                gatesEncountered++;
                it->second->evl();
            }
        }
    }


    ofstream outputFile;
    outputFile.open("simulationResult.txt", ios_base::app);

    cout<<"\nSimulation Input Values at "<<present_time<<": "<<endl;
    for(int i=0; i<inputs.size(); i++){
        cout<<inputs[i]<<" is : "<<wires[inputs[i]].value<<endl;
    }

    cout<<"\nSimulation Output Values at "<<present_time<<": "<<endl;
    outputFile<<"At "<<present_time<<": ";
    for(int i=0; i<outputs.size(); i++){
        cout<<outputs[i]<<" is : "<<wires[outputs[i]].value<<endl;
        outputFile<<wires[outputs[i]].value;
    }
    outputFile<<'\n';
    outputFile.close();

    return;
}
```

بخش D

شبیه سازی انجام شده در تست بنچ، بر اساس time_resolution=5 صورت گرفته و هر گام از شبیه سازی در 5 واحد زمانی صورت میگیرد. سیگنال های ورودی و خروجی را در تصویر زیر مشاهده میکنید (میشود با تغییرات اندکی سیگنال های میانی را هم نمایش داد، ولی برای شلوغ نشدن خروجی کنسول، فقط به نمایش مقادیر ورودی و خروجی ها اکتفا کردم):

```
Simulation Input Values at 40:
N1 is : 0
N2 is : 0
N3 is : 0
N6 is : 0
N7 is : 1

Simulation Output Values at 40:
N22 is : 0
N23 is : 1

Simulation Input Values at 45:
N1 is : 0
N2 is : 0
N3 is : 0
N6 is : 1
N7 is : 0

Simulation Output Values at 45:
N22 is : 0
N23 is : 0
```

سایر موارد قید شده در پروژه

محاسبه ی observability طبق جدول زیر صورت گرفته است:

TABLE 2.4 ■ Probability-Based Observability Calculation Rules

	Observability (Primary Output, Input, Stem)
Primary Output	**1**
AND/NAND	∏ (output observability, 1-controllabilities of other inputs)
OR/NOR	∏ (output observability, 0-controllabilities of other inputs)
NOT/BUFFER	Output observability
XOR/XNOR	*a*: ∏ (output observability, *max* {0-controllability of *b*, 1-controllability of *b*})
	b: ∏ (output observability, *max* {0-controllability of *a*, 1-controllability of *a*})
Stem	*max* {branch observabilities}

Note: *a* and *b* are inputs of an XOR or XNOR gate.

مطابق جدول بالا، کد محاسبه observability به طور مثال گیت NAND به ترتیب زیر در بدنه تابع evl اش نوشته شد:

```
i1->observability = o1->observability * i2->controllability;
i2->observability = o1->observability * i1->controllability;
```

با اعمال تغییراتی در تابع تعیین کننده Level گیت ها و با صدا زدن evl گیت ها، از خروجی به ورودی (برای محاسبه observability) هر دو مقادیر controllability و observability مدار در نقاط مختلف بدست آمد:

```
void recognizeLevel(map<string,gates*> GATES){
    int maxLevel=0;
    cout<<"\n\n";
    for(int i=0; i<GATES.size(); i++)
        for(auto it=GATES.begin(); it!=GATES.end(); it++){
            it->second->evl();
            maxLevel=MAX(maxLevel, it->second->Level);
        }

    for(auto it=GATES.begin(); it!=GATES.end(); it++)
        cout<<"Level of "<<it->first<<" is: "<<it->second->Level<<endl;

    int gatesEncountered=0;
    while(gatesEncountered!=GATES.size()){
        for(auto it=GATES.begin(); it!=GATES.end(); ++it){
            if(it->second->Level == maxLevel){
                gatesEncountered++;
                it->second->evl();
            }
        }
        maxLevel--;
    }
}
```