به نام خدا

تمرین ۳ درس سیستم های نهفته و بی درنگ، ترم بهار سال تحصیلی ۱۳۹۷-۱۳۹۸

دانشکده کامپیوتر، دانشگاه شهید بهشتی

تاریخ تعریف تمرین: ۹۷-۱۲-۲۶

مهلت آپلود در سامانه درس افزار : پنج شنبه ۹۸-۰۱-۲۲ ساعت ۲۳:۵۹

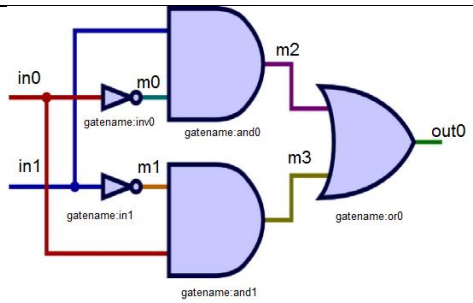
زمان تحویل حضوری: بعدا اعلام می شود.

موضوع: پیاده سازی شبیه ساز مبتنی بر رخداد (Event-driven Simulator) برای مدارهای با گیت های منطقی پایه

هدف: آشنایی با system specification و system simulation برای سامانه های مبتنی بر رخداد

توضیحات:

۱. پشتیبانی از گیت های منطقی not ،and (۲ ورودی) و or (۲ ورودی) مد نظر است.
۲. نرم افزار شبیه ساز را به دلخواه با هر کدام از زبانهای سطح بالا مانند C++، C#، Java ، Python و یا Matlab طراحی نمایید.
۳. این نرم افزار تنها یک فایل ورودی با فرمت text را دریافت می کند. در این فایل ساختار مدار مبتنی بر گیت های فوق توصیف شده است. همچنین زمان انجام شبیه سازی، نام سیگنالهای ورودی و خروجی و میانی و همچنین مقدار ورودی ها در زمانهای مختلف نیز مطابق فایل ورودی نمونه در جدول زیر توضیح داده شده است. به دلخواه می توانید موارد دیگر را نیز در این فایل اضافه کنید.



Input file format:

```
simtime:  500;                  // Max Simulation time =500 ns
input:    in0,in1;              // input signal names
output:   out0;                 // output signal names
signal:   m0,m1,m2,m3;          // internal signal names
not:      not0,in0,m0,5;        // not gate ,name=not0,inp=in0,outp=m0,delay=5ns
not:      not1,in1,m1,0;        // not gate ,name=not0,inp=in0,outp=m0,delay=0ns → causes delta time in simulation
and:      and0,in0,m0,m2,6;     // and gate, name=and0, inp0=in0,inp1=m0,outp=m2,delay=6ns
and:      and1,m1,in0,m3,6;     //…
or:       or0,m2,m3,out0,7;     //…
value:    in0,0,0,1,5,0,10,1,20;  // define input signal value as in0=0 after 0ns,  1 after 5ns, 0 after 10ns,1 after 20ns
value:    in1,0,0,1,7,0,12,0,35;  // …
```

جدول ۱: نحوه نوشتن فایل ورودی برای یک مثال ساده

۴- کامپایلر ساده: برای پیاده سازی شبیه ساز باید ابتدا یک برنامه (کامپایلر) ساده که خط به خط اطلاعات را از فایل ورودی می خواند بنویسید. بعد از خواندن هر خط محتویات آن خط را پردازش نموده و مطابق توضیحات زیر آنها را استفاده نمایید.

فرمت خط ها به این ترتیب است:

subject: element0,element1,element1, … ; //comment

هشت نوع subject وجود دارد:

simtime برای اعمال زمان حداکثر شبیه سازی

Input لیست سیگنال های ورودی

Output لیست سیگنال های خروجی

Signal لیست سیگنال های میانی

Not تعریف یگ گیت not در مدار و نام و اتصالات و تاخیر آن

And تعریف یک گیت and در مدار و نام و اتصالات و تاخیر آن

Or تعریف یک گیت or در مدار و نام و اتصالات و تاخیر آن

Value اعمال مقادیر مختلف به ورودیها در زمان های مختلف

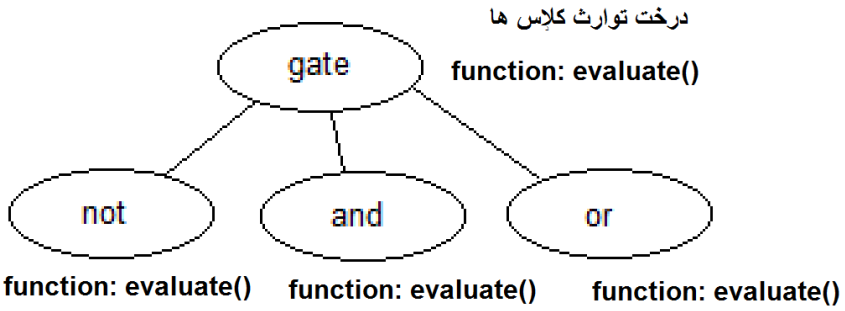
۵- (ساخت مدل مدار درون حافظه memory model) در زبان برنامه نویسی مورد نظر خود می توانید (ترجیحا) از object oriented programming استفاده کنید و کلاس هایی با نام ها و به صورت درخت توارث به صورت زیر (شکل ۱) تعریف کنید. همچنین از مفهوم polymorphism می توانید استفاده نمایید. بدین معنی که در کلاس بالایی تابع ()evaluate تعریف شود و همه کلاس های پایین هم این تابع را override نمایند. همچنین کلاسی از نوع signal تعریف کنید و برای همه انواع سیگنال استفاده کنید. در آن یک تایپ در نظر بگیرید که مقادیر ورودی/خروجی/میانی بودن هر سیگنال در current-sim-time مشخص شود. بدین ترتیب کامپایلر بعد از خواندن خطوط مربوط به input/output/signal ، از کلاس مربوط به سیگنال ها instance های لازم را درست کند. این instanceها باید در یک لیست ذخیره شوند. همچنین کامپایلر بعد از خواندن خطوط مربوط به گیت ها and/or/not ، از کلاس های مربوط به گیت ها مطابق شکل۱ ، instance درست کند. لازم به ذکر است در تعریف کلاس gate لازم است که pointerی به گیتی که خروجی به آن متصل است وجود داشته باشد و بعد از پردازش خط pointer به درستی تنظیم شود. همچنین دو pointer به دو گیتی که ورودی به آنها متصل است قرار دهید (برای گیت not یکی از pointerهای ورودی null می ماند، همچنین برای گیتهایی که به سیگنال خروجی وصل هستند pointer خروجی null می ماند). دقت کنید که به مرور که خط ها خوانده می شود لازم است pointerهای مربوط به instanceهای ساخته شده در خطوط قبلی نیز با توجه به اطلاعات خطوط جدید به روزرسانی شوند. همچنین توجه شود که لازم است درون کلاس سیگنال نیز لیستی از گیت ها وجود داشته باشد که مشخص کند که هر سیگنال (ورودی و یا میانی) به ورودی کدام گیت ها متصل هستند. با استفاده از این لیست می توانید پس از تغییر مقدار هر سیگنال در لیست schedule تعیین کنید که کدام گیتها باید evaluate شوند. توضیح لیست schedule در ادامه خواهد آمد.

بدین ترتیب بعد از خوانده شدن فایل ورودی توسط کامپایلر مدل حافظه نیز در درون حافظه موقت درست شده است و همه pointerها باید به درستی تنظیم شده باشند.

نکته ۱: استفاده از polymorphism به این صورت است که اگر چه pointerها به gateها از نوع gate است ولی instanceها از نوع گیت های مشتق شده از گیت (and/or/not) می باشد.و در صورت call شدن تابع g->evaluate() تابع مربوط به هر object در زمان runtime به درستی اجرا می شود، بدین معنی که برای and محاسبه عملیات and منطقی و برای not محاسبه not انجام می شود.

نکته۲: جهت در نظر گرفتن مقادیر پیش بینی شده برای مقادیر سیگنال ها در خطوط value لازم است یک لیست در برنامه شبیه ساز در نظر گرفته شود. در این لیست هر عنصر آن شامل pointer به object از نوع signal و مقدار و زمان اعمال مقدار به سیگنال در نظر گرفته می شود. بعد از رسیدن کامپایلر به خطوط value لازم است که (به عنوان مقدار دهی اولیه همه مقادیر مورد نظر در زمان صفر شبیه سازی و یا زمان های آینده برای سیگنال های ورودی در این لیست اصطلاحا Schedule شود. لازم به ذکر است که تا اینجا هنوز شبیه سازی شروع نشده و در زمان صفر شبیه سازی هستیم (current-sim-time=0).

در این حالت در لیست schedule صرفا در نظر گرفته می شود که مثلا در ازای خط "value: in0,0,0,1,5" برای سیگنال با نام in0 در زمان 0ns مقدار 0 و در زمان 5ns مقدار 1 در لیست event ها اصطلاحا schedule شود.



شکل ۱: درخت توارث کلاس ها برای انواع گیت

۶- (شبیه سازی) پس از ساخته شدن مدل درون حافظه و تنظیم شدن همه pointerها به درستی، مدلی در اختیار داریم که از طریق آن می توانیم شبیه سازی انجام دهیم. برای این کار لازم است درون برنامه متغیری به نام current-sim-time داشته باشیم که در ابتدای کار مقدار آن برابر با صفر باشد. همچنین دقت کنید که در این لحظه (current-sim-time=0) لیستی از event های schedule شده روی ورودی ها برای زمان های آینده در لیست مورد نظر داریم. توجه کنید که این لیست باید بر اساس اندیس زمان (event) در هر لحظه قابل دسترسی باشد (مثلا می تواند بر اساس زمان مرتب و یا به اصطلاح sort شده باشد). همچنین در این لیست باید امکان اضافه کردن eventهای جدید برای سیگنال های میانی و یا سیگنال های خروجی وجود داشته باشد. همچنین امکان حذف کردن event هایی که اجرا شده اند نیز در لیست وجود داشته باشد.

برای پیاده سازی تابع شبیه سازی مثلا تابعی به نام Simulate(simtime) پیاده سازی نمایید. در سیکل ابتدایی شبیه سازی (current-sim-time=0) همه eventها در زمان صفر را بر اساس لیست event ها به ترتیب پردازش می کنیم. اجرای هر event روی هر سیگنال باعث اجرای تابع ()evaluate در گیت (های) مورد نظر می شود. این گیت ها آنهایی هستند که آن سیگنال روی آنها اثر می گذارد و یا به تعبیری به آنها وصل است. در صورتی که خروجی گیت عوض شود لازم است مقدار جدید روی سیگنال خروجی آن گیت درون لیست event ها با توجه به تاخیر گیت دوباره schedule شود. چنانچه تاخیر صفر برای گیت ها وجود نداشته باشد حتما زمان schedule شدن برای آینده است و در current-sim-time جاری، اصطلاحا event های همین زمان expire نمی شوند (expire شدن یعنی اجرا شدن event). در صورتی که بخواهیم تاخیر صفر را برای گیت ها در شبیه ساز پشتیبانی نماییم علاوه بر Current-sim-time متغیر دیگری به نام delta داریم (مطابق توضیحات و اسلایدهای سر کلاس درس) که delta=0,1,2... به ازای هر Current-sim-time می تواند باشد. متغیر delta برای پیاده سازی event های زمان جاری که ناشی از تاخیر صفر روی سیگنال ها است استفاده می شود. بدین ترتیب تا زمانی که سیگنال گیت ها در هر زمان جاری stable نشود متغیر delta افزایش یک واحدی در یک حلقه خواهد داشت تا همه eventها در زمان

جاری اجرایی شوند. برای سادگی میتوانید در شبیه ساز ابتدا تاخیر صفر گیتها (delta) را پشتیبانی نکنید و پس از اتمام آن در مرحله بعد این قابلیت را هم به شبیه سازتان اضافه نمایید.

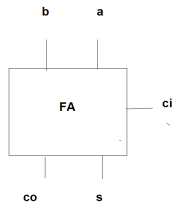
پس از اتمام event ها در زمان current-sim-time=0 برنامه شبیه ساز لیست event ها را بررسی کرده و اولین زمان که در آن یک event وجود دارد را پیدا کرده (مثلا زمان 5ns). سپس متغیر زمان جاری شبیه سازی را به آن تغییر می دهد. به عنوان مثال current-sim-time=5.

روند انجام شده در زمان صفر شبیه سازی دوباره در زمان جدید تکرار می شود و همه event ها در زمان جاری expire می شوند. ترتیب اجرای eventها در زمان جاری را برای سادگی می توانید به دلخواه انتخاب کنید. (دقت کنید این می تواند بالاقوه در حالت های پیچیده باعث اشتباه در شبیه سازی شود. در حالت کلی این ترتیب با توجه به ساختار Inertial و یا Transport سیگنال ها و نحوه schedule شدن اعمال می شود و در اینجا ما به آن نخواهیم پرداخت). همچنین در صورتی که روی یک سیگنال در یک زمان بیش از یک event بخواهد schedule شود موقع schedule کردن بقیه event ها روی آن سیگنال بقیه eventها در آن زمان را باید حذف کنیم. به عبارتی آخرین event را فقط در لیست می گنجانیم و مورد استفاده قرار می دهیم.

روند افزودن current-sim-time در حلقه ای اجرا می شود تا اینکه زمان مورد نظر شبیه سازی تمام شود و یا اینکه حتی قبل از پایان زمان شبیه سازی دیگر هیچ event ای برای expire شدن وجود نداشته باشد. که در هر دو صورت روند شبیه سازی متوقف می شود.

۷- برای این شبیه ساز در نظر بگیرید که به ازای هر کدام از سیگنال های خروجی یک فایل text مجزا درست شود و مقادیر تغییر خروجی در آن فایل به ترتیب به (همراه زمان تغییر) گنجانده شود.

۸- پس از اتمام پیاده سازی شبیه ساز، مدار یک full-adder را در سه حالت زیر با استفاده از شبیه ساز خودتان، شبیه سازی نمایید. (simtime=500ns در نظر بگیرید)



الف: همه گیتها دارای تاخیر 5 ns باشند و ورودی ها در ابتدا به صورت زیر schedule شوند.

ci=0 after 0ns, 1 after 50 ns, 0 after 100ns, 1 after 150ns , 0 after 200ns, 1 after 250 ns, 0 after 300ns, 1 after 350ns , 0 after 400ns
a=0 after 0ns, 1 after 100 ns, 0 after 200ns, 1 after 300ns , 0 after 400ns
b=0 after 0ns, 1 after 200ns, 0 after 400ns

ب: گیتهای and دارای تاخیر 5ns و گیتهای or دارای تاخیر 7ns و گیت های not دارای تاخیر 0ns باشند.
مقادیر ورودی مانند حالت الف باشد.

ج: همه گیتها دارای تاخیر صفر باشند
مقادیر ورودی مانند حالت الف باشد.

موفق باشید
رحمتی