به نام خدا

پروژه تحقیقاتی systemC

فاز یک

استاد : دکتر امیر خرسندی

گردآورندگان : سیاوش قاسمی . عرفان بهرامی . محمدحسین احتشامی
. محمد سقلی

در فاز ۱ این پروژه قصد داریم در مورد نحوه ی کار و استفاده از SystemC تحقیقاتی را انجام داده و موارد زیر را به طور مفصل شرح دهیم:

۱- نگاه کلی به System C

۲- تاریخچه

۳- تفاوت اصلی با C/C++

۴- ورژن های SystemC

۵- عناصر و اجزای System C

۶- زبانهای برنامه نویسی پشنیبانی شده در این هسته

۷- توان/انرژی در SystemC

۸- توضیحات فاز ۲ پروژه

نگاه کلی

SystemC مجموعه ای از کلاس های زبان C++ و ماکروهایی(macro) است که یک رابط شبیه سازی رویدادمحور (event-driven simulation interface) را ارائه می دهند و همچنین می توان شبیه سازی رویداد های گسسته را با آن مشاهده کرد. این امکانات، طراح را قادر می سازد تا شبیه سازی فرآیندهای همزمان را با استفاده از syntax ساده C/C++ انجام دهد.
فرآیندهای SystemC می توانند در محیط شبیه سازی شده در زمان واقعی(real-time environment)، با استفاده از سیگنال های تمام انواع داده ها ارائه شده توسط C ++، برخی از امکانات اضافی ارائه شده توسط کتابخانه SystemC، و همچنین تعریف شده توسط کاربر، ارتباط برقرار کنند.
در بعضی موارد، SystemC زبان های توصیف سخت افزاری VHDL و Verilog را تقلید می کند، اما به طور دقیق تر باید آن را به عنوان یک زبان مدل سازی در سطح سیستم توصیف کرد.

SystemC یک زبان طراحی سیستم است که در پاسخ به نیازهای فراگیر برای یک زبان توسعه یافته است که بهره وری کلی برای طراحان سیستم های الکترونیکی را بهبود می بخشد. به طور معمول، سیستم های امروز شامل سخت افزار و نرم افزار خاص برنامه می شوند. علاوه بر این، سخت افزار و نرم افزار معمولا در یک برنامه دقیق توسعه داده می شوند.

SystemC در واقع با بهره گیری از مزایای بهره وری واقعی، اجازه می دهد مهندسان نرم افزار و قطعات سخت افزاری را باهم ترکیب کنند زیرا این اجزا در سیستم نهایی وجود دارد، اما این سطح بالایی از انتزاع، تیم طراح را در درک اولیه ای از فرآیند طراحی پیچیدگی ها و تعاملات کل سیستم که باعث می شود سیستم های بهتر با بهره وری بیشتر از طریق استفاده مجدد از مدل های سیستم اولیه راهنمایی کند.

تاریخچه

در تاریخ 1999-09-27 : ایده SystemC اعلام شد.

در تاریخ 2000-03-01 : SystemC ورژن 0.91 عرضه شد.

در تاریخ 2000-03-28 : SystemC ورژن 1.0 عرضه شد.

در سال 2002 : SystemC (osci) ورژن 1.0.2 عرضه شد.

در تاریخ 2003-06-03 : SystemC LRM (language reference manual) ورژن 2.0.1 عرضه شد.

در سال 2005 : IEEE std 1666-2005 LRM

در تاریخ 2005-06-06 : SystemC در سطح مدل سازی (TLM) ورژن 1.0 وSystemC LRM ورژن 2.1 عرضه شد.

در تاریخ 2007-04-13 : SystemC ورژن 2.2 عرضه شد.

در سال 2009 : systemC TLM 2.0 standard

در سال 2009 : پیش نویس قابل سنتز 1.3 آن عرضه شد.

سال 2011 : IEEE std 1666-2011 LRM

در تاریخ 2011-11-10 : IEEE approves the IEEE 1666–2011 standard for SystemC

سال 2012 : SystemC ورژن 2.3 عرضه شد.

در تاریخ 2016-04-06 : IEEE approves the IEEE 1666.1–2016 standard for

- systemC ورژن ۱,۰:
  ارایه VHDL بعنوان یه قابلیت
  کرنل شبیه سازی
  انواع داده های محاسباتی نقطه ثابت
  سیگنال ها (کانال های ارتباطی)
  ماژول ها
  شکستن طرح ها به قسمت های کوچکتر

- systemC ورژن ۲,۰:
  دوباره نویسی کتابخانه بصورت کامل برای ارتقا به SLDL
  رویدادها بعنوان رفتار اولیه تلقی می شدند
  کانال ها واسطه ها و درگاه ها
  مدل سازی قدرتمند تر

- systemC ورژن ۳,۰ در آینده:
مدل سازی سیستم عامل ها
پشتیبانی از مدل ها S/W

تفاوت اصلی با C/C++

تفاوت اصلی که در c با SystemC داریم استفاده از ترد هاست چرا که باید خاصیت پردازش به صورت سخت افزاری را پیاده سازی کنیم از این ترد ها استفاده میکنیم
چیز مهمی که باید در مورد این ترد ها گفت این است که هیچگاه توسط user فراخوانی نمیشود بلکه به وسیله constructorها فراخوانی میشود که این ها حساس به سیگنال های کلاک و یا در تایم های مشخصی صورت میگیرند که این ترد ها عبارتند از :

1.SC_METHODE()
2.SC_THREAD()
3.SC_CTHREAD()

۱- برای ترد هایی که فقط برای انجام یکبار به کار میرود یعنی در کل مناسب شرایطی که نیاز به یک کلاک دارد است .
از جمله استفاده ها در عبارت های منطقی و عبارت های محاسباتی به کار گرفته میشود و قابل سنتز است.
۲- قابل سنتز نیست که در تست بنچ ها به کار گرفته میشود.
۳- همانند ۱ نیست تفاوت ان این است که قابلیت این را دارد که در clock cycle است و قابلیت سنتز دارد.
در بخش توضیح پورت با ذکر مثال تفتوت این سه نوع آورده شده است .

ورژن ها

نسخه ۱ SystemC شامل ویژگی های زبان های سخت افزاری رایج مانند:

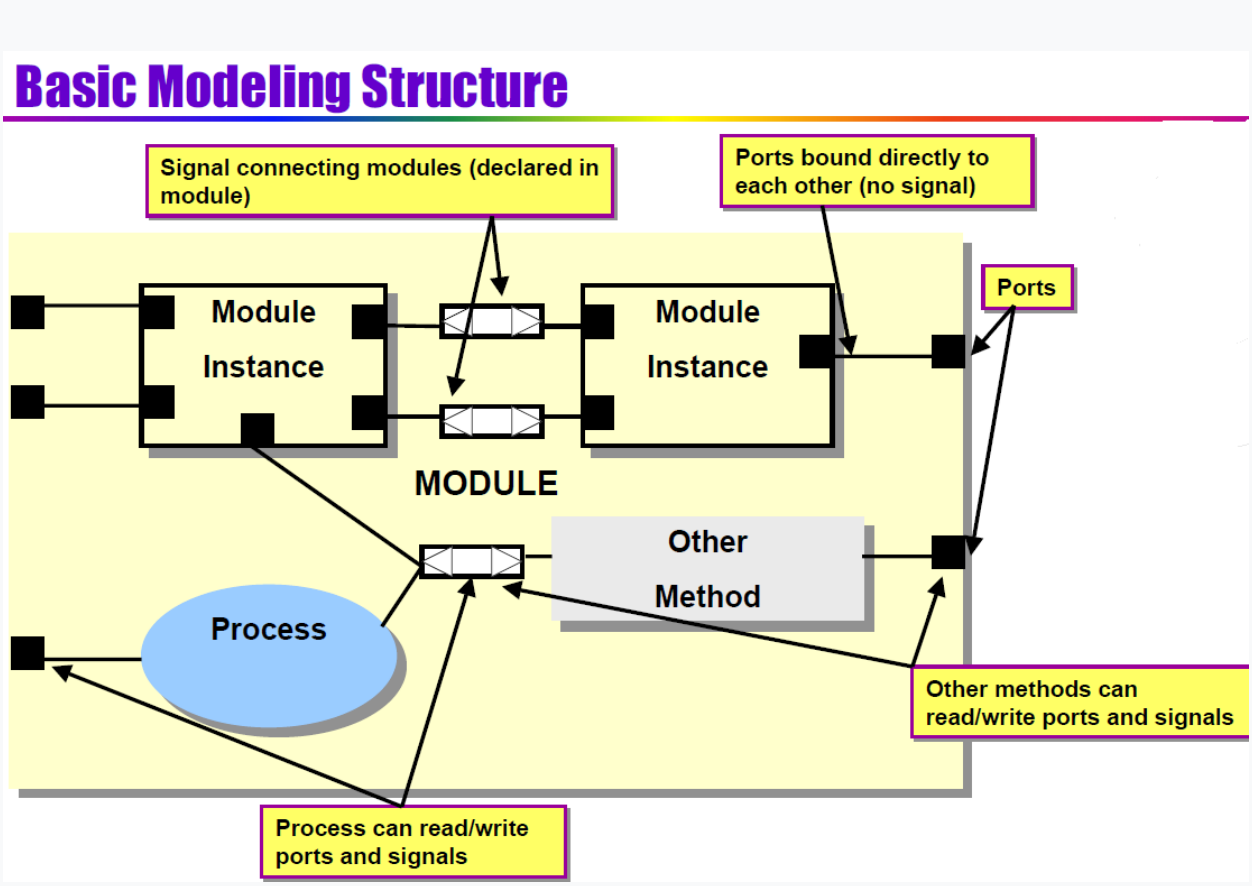
- ساختار سلسله مراتبی (structural hierarchy and connectivity)
- پالس ساعت و تنظیم دقت آن (clock-cycle accuracy)
- چرخه دلتا (delta cycles)
- مقدارهای 0, 1, X, Z
- باس و باس مشترک (bus-resolution functions)

می باشد اما از نسخه ۲ به بعد، تمرکز سیستم SystemC روی مدلسازی سطح معامله (transaction-level modeling) ، انتزاع ارتباطات (communication abstraction)و مدلسازی مجازی پلتفرم(virtual-platform modeling) بیشتر شده است. همچنین نسخه ۲ SystemC پورت انتزاعی، فرآیندهای پویا و اعلان های رویداد به موقع را اضافه کرد.

عناصر و اجزای System C

- ماژول ها (modules)
- پورت ها (ports)
- سیگنال ها (signals)
- پردازش ها (processes)
- کانال ها (channels)
- اکسپورت ها (exports)
- انواع داده (data types)

نحوه ی کار این اجزا را با یکریگر در شکل زیر میبینیم.



حال در زیر به شرح هر یک از عناصر می پردازیم:

ماژول ها (modules):

ماژول ها بلوک های اصلی ساختمان یک طراحی در SystemC هستند.می توان آن ها را شبیه به entity در زبان VHDL دانست. ماژول هامی توانند از طریق پورت ها با دیگر ماژول ها ارتباط برقرار می کنند. در شکل زیر syntax ماژول ها در SystemC را با ارائه مثال توضیح داده شده است.

Module Syntax

Syntax:

```
SC_MODULE(module_name){
    // body of module
};
```

EXAMPLE:

```
SC_MODULE(my_module) {
  // body of module
};
```

Code in header file: my_module.h

Note the semicolon at the end of the module definition

هر module دارای یک سازنده (constructor) است که در زمان ساختن یک instance از ماژول صدا زده می شود. تمامی اعضای یک ماژول (اعم از پورت ها و سیگنال ها و ...) در داخل کانستراکتور initialize می شوند. process ها نیز در داخل کانستراکتور register می شوند. همچنین اگر بخواهیم از یک ماژول در یک ماژول دیگر instance بسازیم باید این کار را در کانستراکتور ماژول مورد نظر انجام دهیم.

Example:

Code in header file: module_name.h

```
SC_MODULE (my_module) {
  // Ports, internal signals, processes, other methods
  // Constructor
  SC_CTOR(my_module) {
    // process registration & declarations of sensitivity lists
    // module instantiations & port connection declarations
  }
};
```

پورت ها (ports):

پورتها از طریق کانال ها اجازه ی ارتباطات را از داخل یک ماژول به بیرون (معمولا به سایر ماژول ها) می دهند.

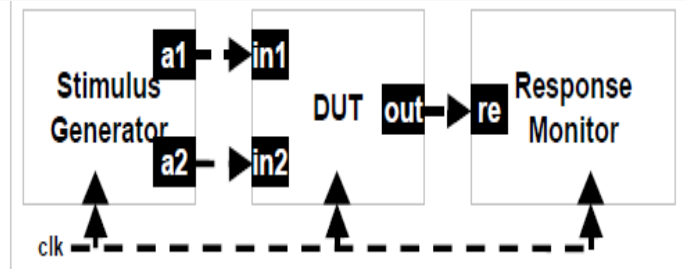
سه نوع پورت داریم :

Input → sc_in<>

Output → sc_out<>

Inout → sc_inout<>

پورت ها زیر مجموعه ی یک ماژول در نظر گرفته می شوند و هر یک دارای data type مستقل هستند.



مثال:

Declaration as data members of the module:

```
SC_MODULE(my_mod) {
    sc_in<t_data>          port_name;
    sc_out<t_data>         port_name;
    sc_inout<t_data>       port_name;
};
```

t_data:

- **data type of the port**

انواع داده (data types):

SystemC چندین نوع داده را ارائه می دهد که از مدل سازی سخت افزار پشتیبانی می کنند . از جمله :

Extended standard types:

- sc_int<*n*> *n*-bit signed integer
- sc_uint<*n*> *n*-bit unsigned integer
- sc_bigint<*n*> *n*-bit signed integer for *n* > 64
- sc_biguint<*n*> *n*-bit unsigned integer for *n* > 64

Logic types:

- sc_bit 2-valued single bit
- sc_logic 4-valued single bit
- sc_bv<*n*> vector of length *n* of sc_bit
- sc_lv<*n*> vector of length *n* of sc_logic

Fixed point types:

- sc_fixed<> templated signed fixed point
- sc_ufixed<> templated unsigned fixed point
- sc_fix untemplated signed fixed point
- sc_ufix untemplated unsigned fixed point

Syntax:

```
sc_int<length>          variable_name ;
sc_uint<length>         variable_name ;
sc_bigint<length>       variable_name ;
sc_biguint<length>      variable_name ;
```

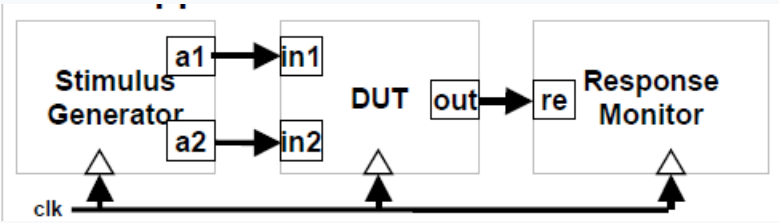
length:

- Specifies the number of elements in the array
- Must be greater than 0
- Must be compile time constant

- Use `[]` to bit select and `range()` to part select.
- Rightmost is LSB(0), Leftmost is MSB (n-1)

سیگنال ها (signals):

سیگنال ها برای مبادله داده ها در اجرای همزمان بین ماژول ها یا بین پردازش ها (processes) ها مورد استفاده قرار میگیرند.بر خلاف پورت ها فقط یک نوع سیگنال داریم. سیگنال ها همیشه یک مقدار را حمل میکنند.چون ممکن است دو سیگنال بین دو پردازش باشد پس لزوما سیگنال ها به پورت ها محدود نمی شوند.یک سیگنال ممکن است عضوی از یک ماژول باشد (برای ارتباطات داخلی) یا اینکه در سطح بالاتر برای اتصال ماژول ها استفاده شوند.



مثال:

Declaration as data members of the module:

```
SC_MODULE(my_mod) {
    sc_signal<t_data>      signal_name;
};
```

t_data:

- data type of the signal

پردازش ها (processes):

عملکرد Module (ماژول ها) توسط پردازش ها (processes) تعریف می شوند. پردازش ها (processes) توابع عضو یک ماژول هستند. SystemC سه نوع process مختلف برای استفاده طراحان سخت افزار و نرم افزار فراهم می کند که تمامی آن ها همزمان هستند.در واقع process ها عناصر اصلی عملیات ها و محاسبات هستند.

```
SC_METHOD ( sc_async_fprocess )
SC_THREAD   ( sc_async_tprocess )
SC_CTHREAD ( sc_sync_tprocess )
```

برای درک تفاوت این ۳ نوع به مثال های زیر توجه کنید.

۱- در اینجا مقدار متغیر incهر بار که process اجرا می شود یک واحد افزایش می یابد.

```
void my_module::proc1()
{
        int inc=5;
         inc++;
        sig2=in1.read() + sig1 + inc;
}
SC_METHOD(proc1);
```

۲- در اینجا مقدار inc هر بار که process فعال شود .(به اصطلاح reactiveشود) به اندازه ی ۱ واحد افزایش می یابد.

```
void my_module::proc2()
{
int inc=1;
while(1){
        out1.write(in2.read() + sig2 + inc);
         inc++;
        wait();
```

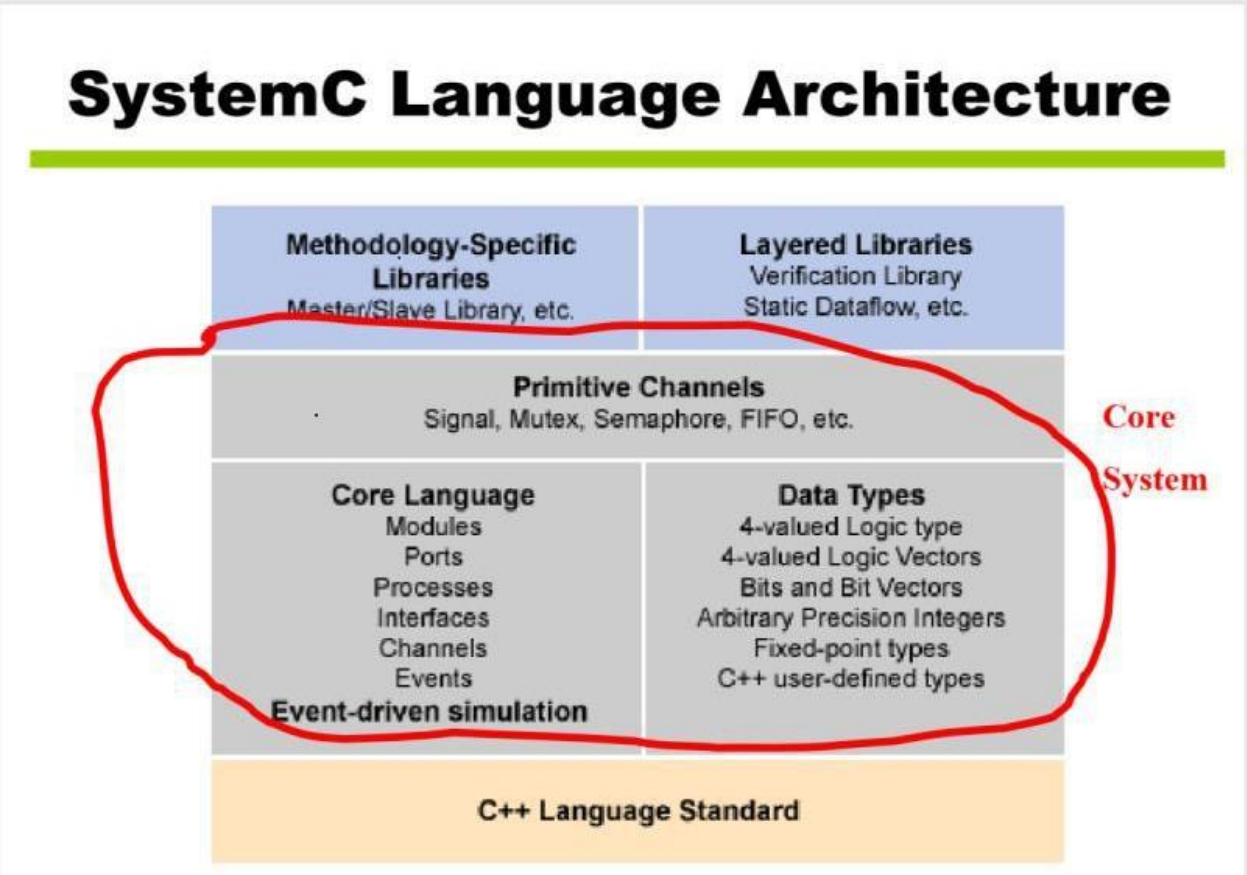
```
        }
}
SC_THREAD(proc2);
```

۳- اینجا مقدار inc در هر لبه ی بالا رونده ی پالس ساعت (clock) به اندازه ی ۱ واحد افزایش می یابد.

```
void my_module::proc3 ()
{
      int inc=1;
       while(1)
       {
              out2.write(inc);
             sig2 = sig2 + inc;
             inc++;
             wait();
       }
}
SC_CTHREAD( proc3 , clk.pos() );
```

زبانهای برنامه نویسی پشتیبانی شده در این هسته

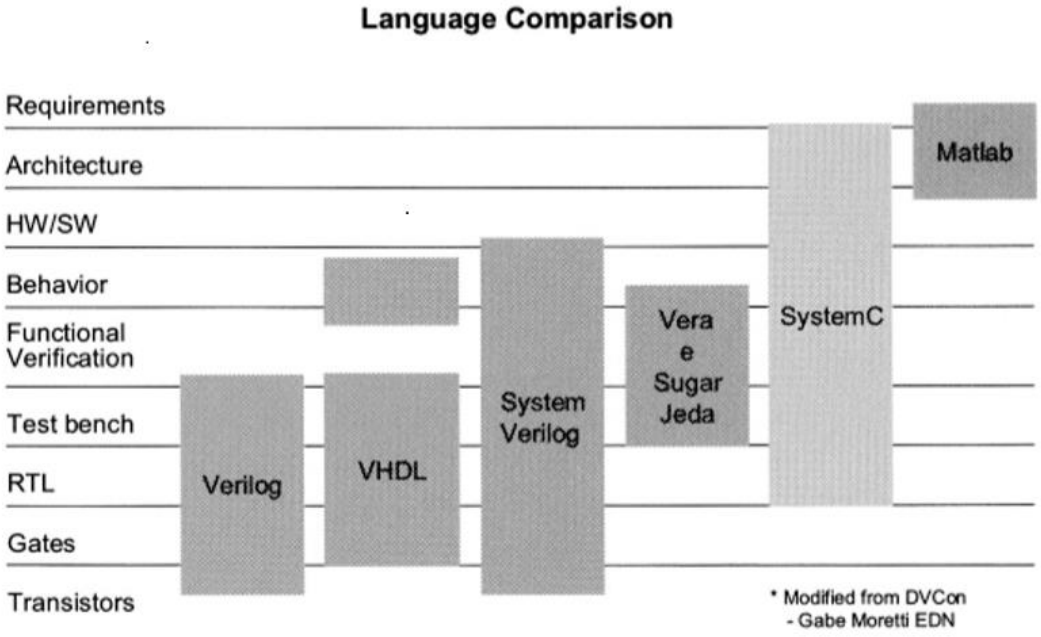
دو زبان اصلی برنامه نویسی که پشتیبانی میشود c , c++ میباشد و معماری زبانهای آن نیز به شکل زیر است.



به طور دقیق، SystemC یک زبان نیست، بلکه یک کتابخانه کلاس است که در زبان C ++ قرار دارد. SystemC یک زبانی نیست که هر مسئله بهره وری طراحی را حل کند. با این حال، هنگامی که SystemC همراه با کتابخانه خود است، شامل بسیاری از ویژگی های مربوط به طراحی سیستم و کارهای مدل سازی است که در میان دیگر زبان ها گم شده و یا پراکنده است. علاوه بر این، SystemC یک زبان مشترک برای نرم افزار و سخت افزار، C ++ را فراهم می کند.

برای رفع ابعاد مختلف طراحی سیستم، زبان های متعددی ظاهر شده اند. گرچه آدا و جاوا ارزش خود را ثابت کرده اند، اما امروزه C / C ++ برای نرم افزار سیستم جاسازی شده استفاده می شود. زبانهای توصیف سخت افزاری (HDLs)، VHDL و Verilog برای شبیه سازی و تلفیق مدارهای دیجیتال استفاده می شود. Vera و e زبان های انتخابی برای تایید عملکرد مدارهای مجتمع پیچیده کاربردی (ASICs) هستند. SystemVerilog یک زبان جدید است که زبان Verilog را برای رفع بسیاری از مشکلات طراحی سیستم سخت افزاری طراحی می کند. Matlab و چندین ابزار و زبان دیگر مانند SPW و System Studio به طور گسترده ای برای گرفتن الزامات سیستم و توسعه الگوریتم پردازش سیگنال استفاده می شود.

شکل زیر کاربرد این و دیگر زبان های طراحی سیستم را برجسته می کند.



SystemC contrasted with other design languages

توان/انرژی در SystemC

برآورد توان در SystemC با استفاده از شبیه سازی انجام می شود. Powersim یک کلاس کتابخانه SystemC است که برای محاسبه مصرف توان و انرژی سخت افزار در سطح سیستم توصیف شده است. برای این منظور، اپراتورهای C++ نظارت می شوند و مدل های مختلف انرژی برای هر نوع داده (SystemC data types)مورد استفاده قرار می گیرند. شبیه سازی با Powersim نیازی به تغییر در کد برنامه ندارد.

توضیحات فاز ۲ پروژه

بر روی سیستم عامل لینوکس اقدامی که صورت گرفت اضافه کردن systemC در ابتدای کار برای اجرای کتابخانه مورد نظر است که با دستور زیر در لینوکس قابلیت اضاف کردن را دارد

```
wget
http://www.accellera.org/images/downloads/standards/systemc
/systemc-2.3.3.gz
```

پس از دریافت بسته بالا آن را از حالت فشرده خارج میکنیم

```
tar -xzf systemc-2.3.3.gz
sudo mkdir /usr/local/systemc-2.3.3/
cd systemc-2.3.3 && mkdir objdir && cd objdir
```

سپس آن را نصب میکنیم

```
sudo ../configure --prefix=/usr/local/systemc-2.3.3/
sudo make -j$(nproc)
sudo make install
```

چگونگی اجرای فایل مورد نظر به صورت زیر است systemC پس از اقدام به نصب:

```
g++ -I. -I /usr/local/systemc-2.3.3/include -L. -
L/usr/local/systemc-2.3.3/lib-linux64 -Wl,-
rpath=/usr/local/systemc-2.3.3/lib-linux64 -lsystemc -lm -o
first first.cpp
```

باید فایل اجرایی first.cpp که در صورتی که دستور بالا متوجه خطایی نشده باشد در فایل اجرای برنامه تولید شده باشد که در این صورت با اجرای کد زیر میتوانیم پاسخ خود را در ترمینال ببینیم first

```
sudo ./first
```

بنویسیم که systemC برای اجرای این پروژه ما باید در دو مرحله اقدام کنیم اول اینکه باید یک برنامه تحت طراحی اصلی این پروژه است یعنی محاسبه ی عبارت جبری مورد نظرمان مرحله دومی برنامه همانند تاپ ماژول که تا که این خروجی برنامه را vcd. مقدار اولیه به ورودیمان میدهیم و همچنین تولید فایلی در این برنامه با فرمت بتوانیم با شکل موج نمایش دهیم برای این که بتوانیم شکل موج برنامه رو نمایش دهیم این اقدام را صورت میدهیم که میدهیم و برنامه به ما شکل خروجی موج ها را در زمان های متفاوت GTKwave را به برنامه vcd. فایل نمایش خواهد داد

پس ابتدا به شرح طراحی اصلی این برنامه یعنی محاسبه عبارت جبری میپردازیم
در این برنامه ما به چهار ورودی و همچنین به یک خروجی برای نمایش جواب نیاز داریم
همینطور از یک کلاک و قابلیت ریست هم استفاده کرده ایم که سنکرون و از نوع بالارونده میباشد

و در یک تابع عبارت لازم را محاسبه میکنیم و پاسخ را نمایش میدهیم که در این تابع از یک حلقه استفاده شده است که پس از محاسبه و نمایش هر پاسخ یک کلاک صبر میکند و مجددا حلقه را اجرا میکند

در مرحله دوم ما مقدار دهی رو به ورودی هایمان میکنیم
را به متغییر های فایل طراحی شده اصلی مان متصل میکنیم testbench و متغییر های ایجاد شده در این فایل را ایجاد میکنیم و متغییر های ایجاد شده را به این فایل اضافه vcd. در مرحله بعد اقدامی که صورت میدهیم فایل میکنیم تا تغییرات ایجاد شده در طول عمل محاسبه ثبت کند تا بتوانیم موج های آن را در طول زمان ثبت کنیم و تغییرات را متوجه شویم

```
(a^2 + b^2)^3 - (c^2 + d^2)^2
```

محاسبه عبارت بالا را ما با ۳ تست کیس متفاوت انجام دادیم که هر کلاک هم دو نانو ثانیه به طول می انجامد
که عبارتند از

```
1)a = 1
  b = 1
  c = 1
```

```
        d = 2
result = -17

2)a = 2
  b = 2
  c = 1
  d = 2
result = 487

3)a = 1
  b = 1
  c = 1
  d = 1
result = 4
```

که البته ما به مدت ۱۰ نانو ثانیه ریست را فعال میکنیم که در این مدت پاسخ ما صفر خواهد شد

جواب های ما در یک کلاک اماده میشود که حداقل مقدار ممکن میباشد

در این شبیه سازی کلاک در حالت بالارونده قرار دارد

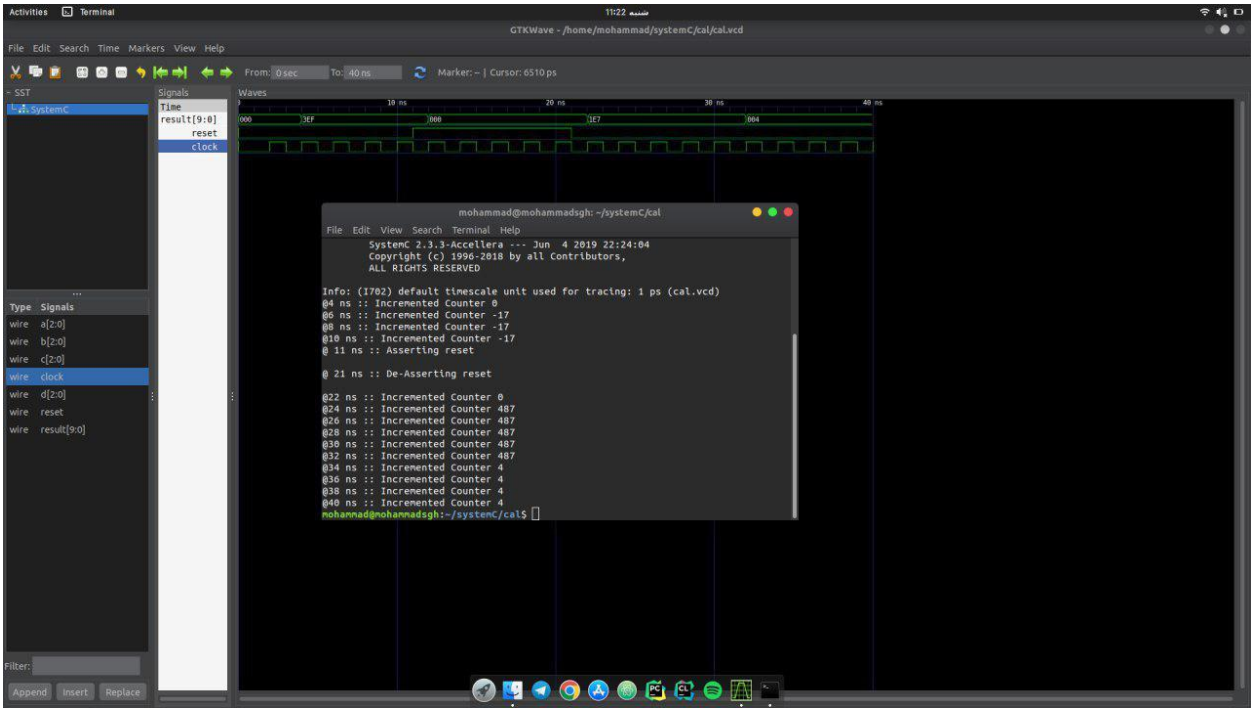
a,b,c,d را فعال کرده ایم که در این بازه زمانی دیگر تغییرات reset در این شبیه سازی ما یک بار تاثیری نخواهد داشت و جواب ما صفر نمایش داده خواهد شد

را صفر نکنیم این شرایط ادامه خواهد داشت reset تا زمانی که

را صفر کنیم result در شروع کار ما ابتدا یک کلاک را صرف این موضوع میکنیم که

و بعد از کلاک اول با هر کلاک مقداری که ما مقدار دهی کرده ایم محاسبه میشود و پاسخ محاسبه میشود

```
g++ -I. -I /usr/local/systemc-2.3.3/include -L. -
L/usr/local/systemc-2.3.3/lib-linux64 -Wl,-
rpath=/usr/local/systemc-2.3.3/lib-linux64 -lsystemc -lm -o
cal cal.cpp cal_tb.cpp
```


GTKWave - /home/mohammad/systemC/cal/cal.vcd
mohammad@mohammadsgh: ~/systemC/cal
SystemC 2.3.3-Accellera --- Jun 4 2019 22:24:04
Copyright (c) 1996-2018 by all Contributors,
ALL RIGHTS RESERVED
Info: (1702) default timescale unit used for tracing: 1 ps (cal.vcd)
@4 ns :: Incremented Counter 0
@6 ns :: Incremented Counter -17
@8 ns :: Incremented Counter -17
@10 ns :: Incremented Counter -17
@ 11 ns :: Asserting reset
@ 21 ns :: De-Asserting reset
@22 ns :: Incremented Counter 0
@24 ns :: Incremented Counter 487
@26 ns :: Incremented Counter 487
@28 ns :: Incremented Counter 487
@30 ns :: Incremented Counter 487
@32 ns :: Incremented Counter 487
@34 ns :: Incremented Counter 4
@36 ns :: Incremented Counter 4
@38 ns :: Incremented Counter 4
@40 ns :: Incremented Counter 4
mohammad@mohammadsgh:~/systemC/cal$