بنام خداوند جان و خرد

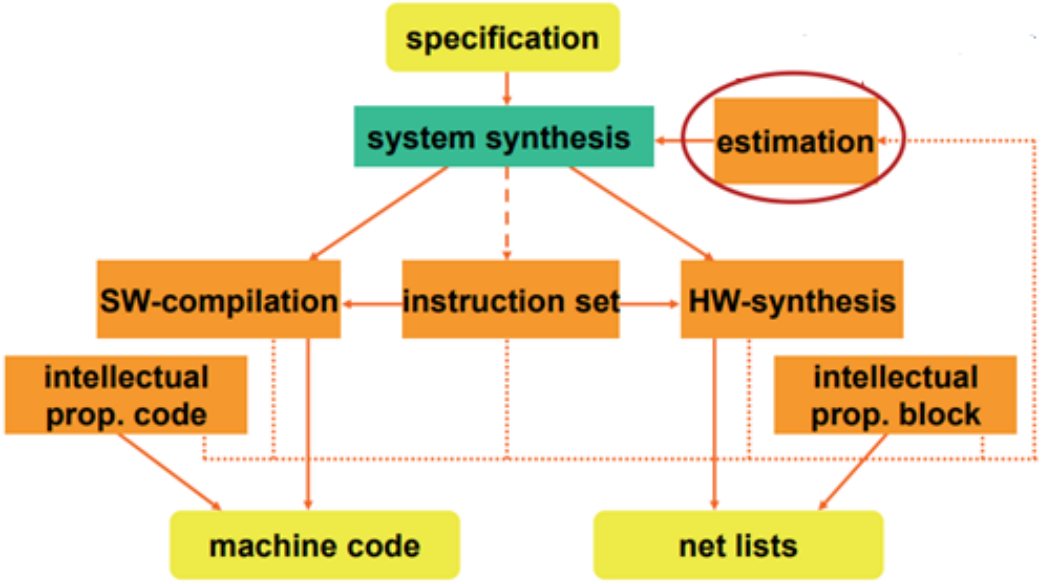
هم‌طراحی سخت‌افزار نرم‌افزار

جلسه بیست‌وسوم: ارزیابی

ارائه‌دهنده: آتنا عبدی
a_abdi@kntu.ac.ir

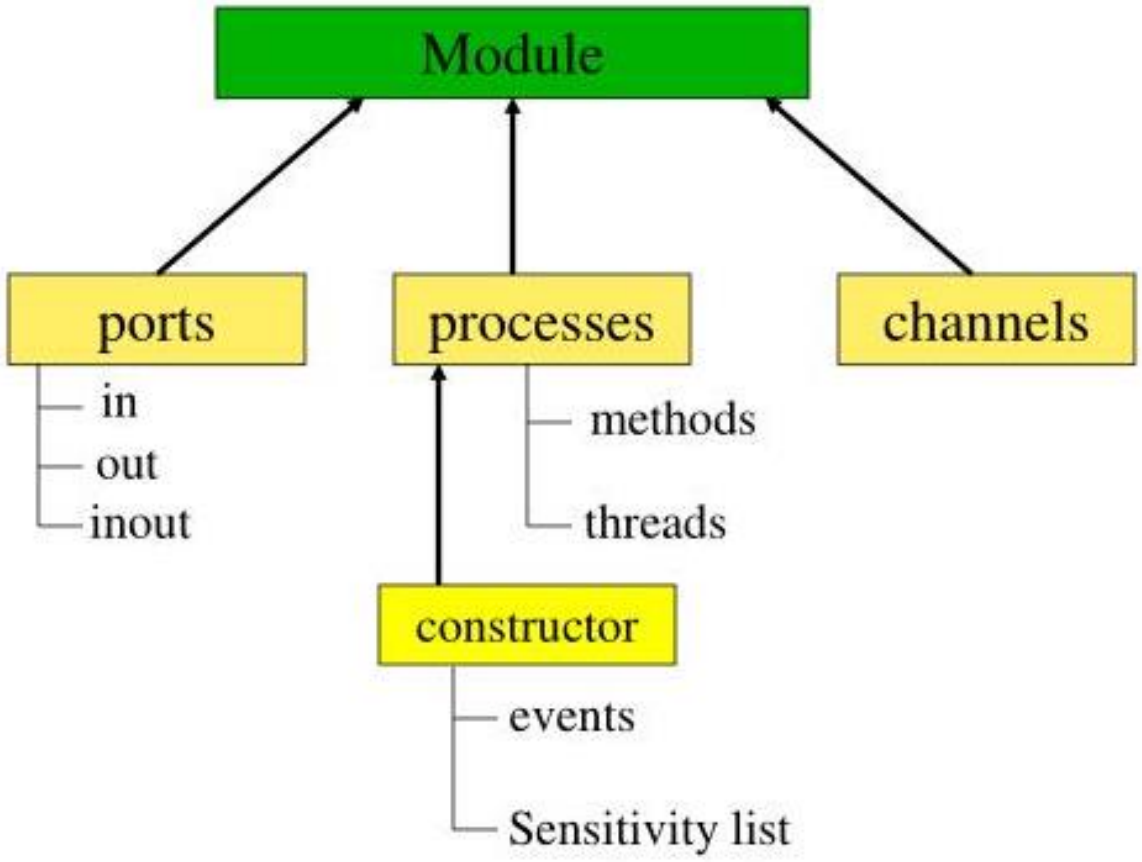
مباحث این جلسه

- ارزیابی فرایند سنتز توأم
  - شبیه‌سازی طراحی
  - اعمال تصمیم‌های فرایند سنتز توأم در پیاده‌سازی



ارزیابی فرایند سنتز توأم

- ساختار برنامه در SystemC:



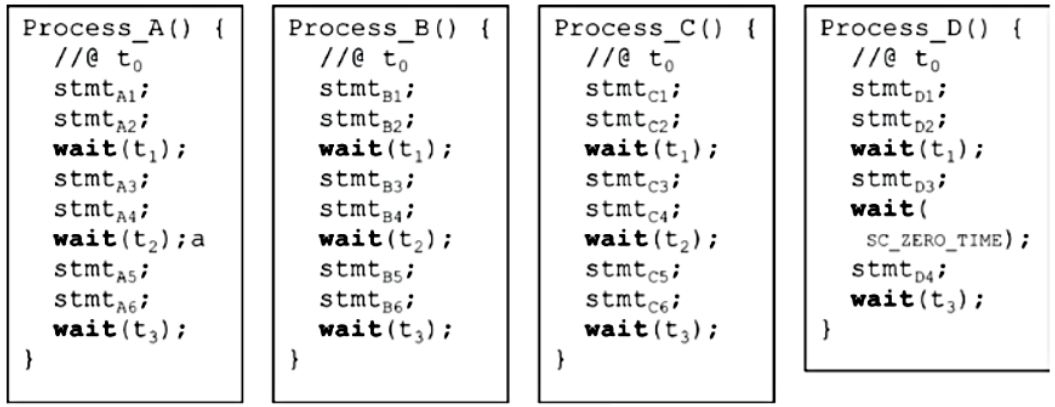
ارزیابی فرایند سنتز توأم

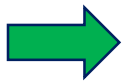
- پیاده‌سازی فرایند سنتز توأم در SystemC:
  - شبیه‌سازی نتیجه تصمیم مرحله افراز توسط تعریف پروسه‌های جداگانه
    - در این مرحله هدف ما طراحی است و به جزئیات پیاده‌سازی در بستر سخت‌افزار نمی‌پردازیم
  - شبیه‌سازی نتیجه تصمیم مرحله زمان‌بندی
    - پیاده‌سازی همروندی در اجرای پروسه‌ها
    - وارد کردن مفهوم time در شبیه‌سازی
    - عملیات سنکرون‌سازی بین پروسه‌ها

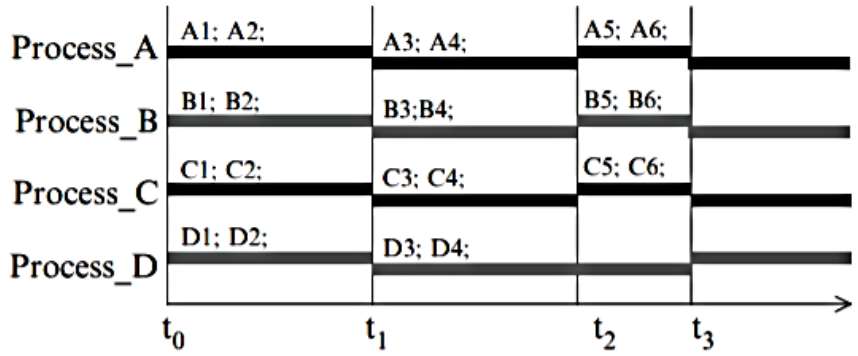
همروندی و زمان‌بندی پروسه‌ها در SystemC

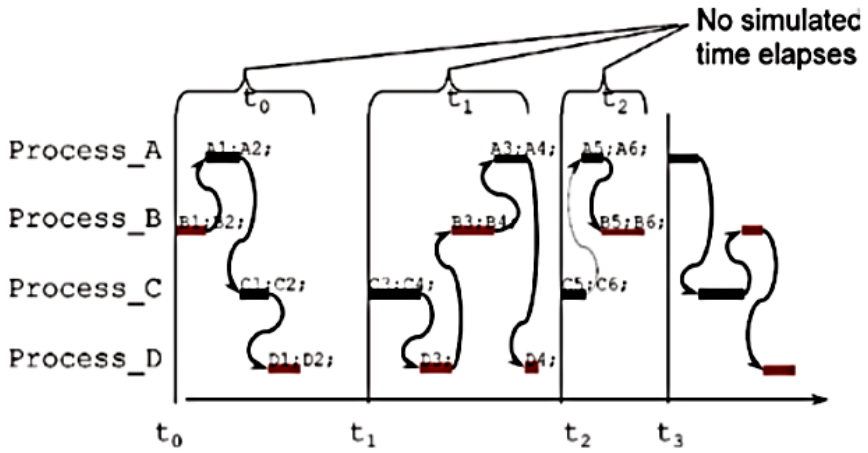
- در فرایند زمان‌بندی لازم است بین اجرای پروسه‌ها سنکرون‌سازی درنظر گرفته شود
  - همروندسازی اجرای پروسه‌ها توسط مفهوم thread
  - طراحی دقیق این فاز اهمیت زیادی دارد
  - مدیریت و ترتیب‌بندی اجرا توسط wait
    - اجرای به‌نوبت پروسه‌ها با فاصله زمانی مشخص در یک حلقه بی‌نهایت
      - در مثال‌های قبلی دیدیم

همروندی و زمان‌بندی پروسه‌ها در SystemC







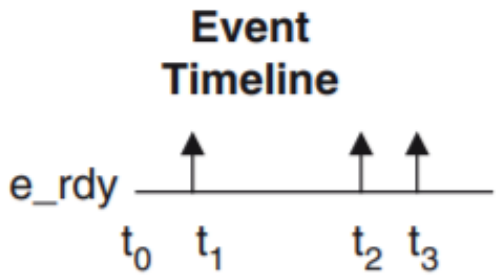


همروندی و زمان‌بندی پروسه‌ها در SystemC

- در پروسه‌های تعریف شده در SystemC لیست حساسیت داشتیم:
  - لیست حساسیت ایستا: توسط "sensitive" در زمان طراحی تعریف می‌شود
    - Sensitive << clock.pos()
  - لیست حساسیت پویا: در طی زمان اجرا ساخته می‌شود
    - شرط‌هایی که در wait ذکر می‌شوند و در طی اجرا معلوم می‌شود چه زمانی محقق می‌گردند
    - wait (e) یا next_trigger()

رخداد در SystemC

- شبیه‌سازی در محیط Event-driven :SystemC
  - Event: رخدادی است که در زمان مشخص رخ می‌دهد و طول زمان و مقدار ندارد
    - تابحال رخداد برای ما زمان بود ولی مفهوم گسترده‌تری دارد
  - شبیه‌سازی رخداد در SystemC توسط کلاس sc_event
  - فعال شدن رخداد نشانه‌ای در سیستم ندارد و تنها بر پروسه‌هایی که منتظر آن بوده‌اند اثر می‌گذارد
    - فعال شدن در لحظه و باید catch شود

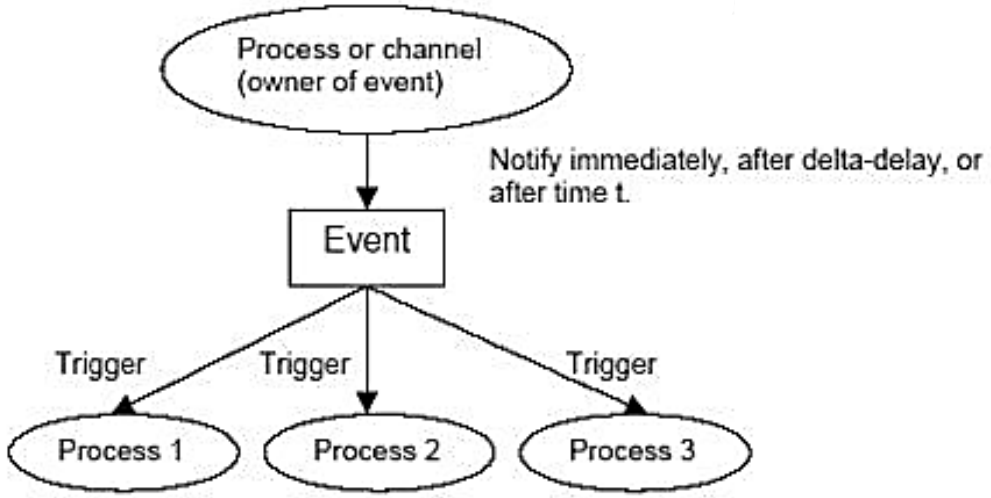


رخداد در SystemC (ادامه)

- sc_event شبیه‌سازی رخدادی که در زمان مشخصی رخ می‌دهد
  - قابلیت فعال و متوقف شدن دارد
  - رخدادها مدت زمان ندارند و اگر فعال شوند و پروسه‌ای متنظرشان نباشد از سیستم حذف می‌شوند
    - لازم است توسط پروسه‌ای catch شوند و از قبل در انتظارشان بود
  - تعریف به‌صورت: `sc_event e1;`
  - دو عملیات روی این کلاس تعریف می‌شود:
    - wait (e1)
    - notify (e1)

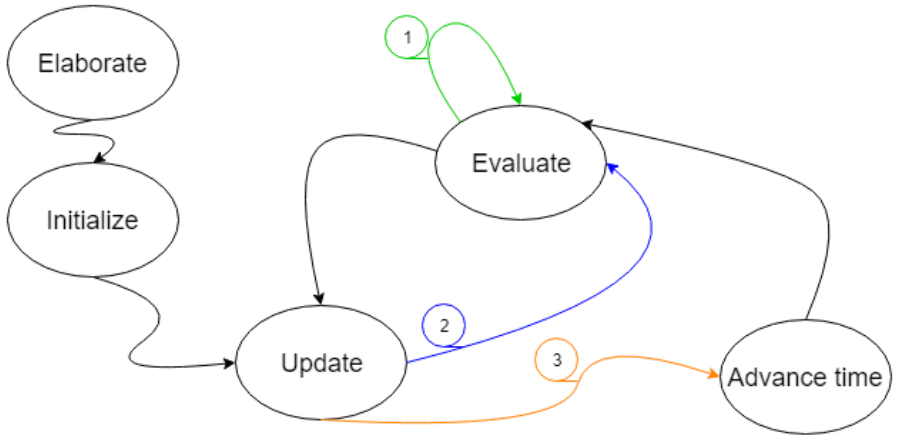
رخداد در SystemC (ادامه)

- استفاده از event در پروسه‌های از نوع thread برای اعمال زمان‌بندی اجرا
  - مشخص کردن شروع یا ادامه به کار یک پروسه: فعال شدن یک رخداد (notify)
  - با فعال شدن رخداد، تمامی پروسه‌های حساس به آن قابل اجرا می‌شوند



فعال شدن رخداد در SystemC

- انواع Notify:
  - نوع Immediate: notify()
    - فعال شدن رخداد در سیکل جاری اجرا و ارزیابی فعلی
  - نوع delta notification: notify(SC_ZERO_TIME)
    - فعال شدن رخداد در اجرای بعدی با به‌روز شدن شبیه‌سازی
  - نوع زمان‌دار: notify(1, SC_NS)
    - فعال شدن رخداد در زمان مشخص شده در آینده



فعال شدن رخداد در SystemC (ادامه)

- انتخاب نوع فعال شدن رخداد برحسب کاربرد

```
sc_event    a, b, c ;              // event declaration
sc_time     t (10, SC_NS)          // declaration of a 10ns time interval

a.notify();                        // immediate notification, current delta-cycle
notify(SC_ZERO_TIME, b);           // delta-delay notification, next delta-cycle
notify(t, c);                      // 10 ns delay

//Cancel an event notification
a.cancel();                        // Error! Can't cancel immediate notification
b.cancel();                        // cancel notification on event b
c.cancel();                        // cancel notification on event c
```

فعال شدن رخداد در SystemC (ادامه)

- در حالتی که یک رخداد چندین بار فعال شود، نزدیک‌ترین دستور پذیرفته می‌شود

```
…
sc_event action;
sc_time now(sc_time_stamp()); //observe current time
//immediately cause action to fire
action.notify();
//schedule new action for 20 ms from now
action.notify(20,SC_MS);
//reschedule action for 1.5 ns from now
action.notify(1.5,SC_NS);
//useless, redundant
action.notify(1.5,SC_NS);
//useless preempted by event at 1.5 ns
action.notify(3.0,SC_NS);
//reschedule action for evaluate cycle
action.notify(SC_ZERO_TIME);
//useless, preempted by action event at SC_ZERO_TIME
action.notify(1,SC_SEC);
//cancel action entirely
action.cancel();
//schedule new action for 1 femto sec from now
action.notify(20,SC_FS);
…
```

انتظار برای رخداد در SystemC

- توسط تابع wait
  - در این وضعیت، کنترل برای اجرای پروسه دیگر به کرنل داده می‌شود
  - وضعیت thread فعلی ذخیره می‌شود
  - با فعال شدن مجدد thread، آخرین اطلاعات ذخیره شده بازیابی می‌شود

```
wait(time);                    // timeout is the event
wait(double,time_unit);        // convenience
wait(event);                   // single event
wait(event1 | eventn...);      // any of these
wait(event1 & eventn...);      // all of these
wait(time,event);              // event or timeout
wait(time,event1 | eventn...);// any event or timeout
wait(time,event1 & eventn...);// all events or timeout
wait(); // static sensitivity
```