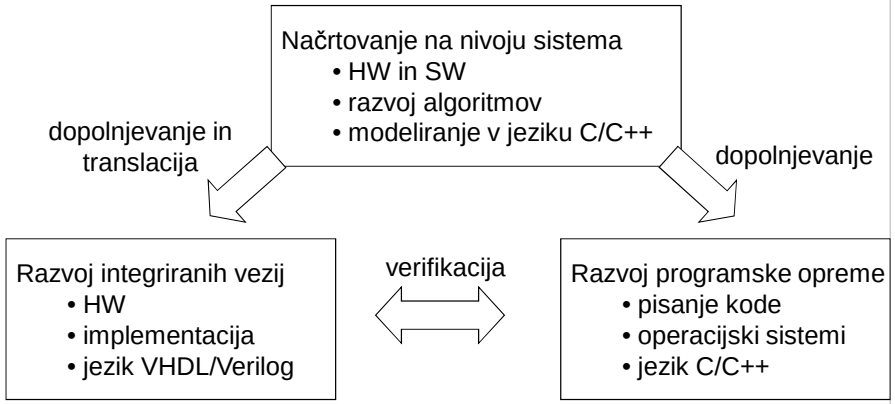
Andrej Trost

Načrtovanje digitalnih el. sistemov

Modeliranje sistemov: SystemC

http://lniv.fe.uni-lj.si/ndes.html

Postopek načrtovanja sistemov

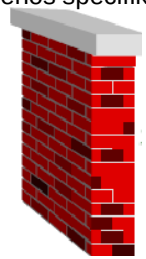


- vrzeli v postopku vplivajo na produktivnost!

Klasičen postopek načrtovanja

Sistemski inženir

1. koncept sistema
2. simulacija v C++
3. pisanje specifikacije
4. prenos specifikacije



HW razvojnik

5. branje specifikacije
6. (re)implementacija
7. (re)verifikacija
8. HDL sinteza

- Težave: pisna specifikacija je nepopolna in nekonsistentna

Ideja načrtovanja v okolju C++

1. koncept sistema
2. simulacija v C++
3. pisanje specifikacije
4. prenos
   - izvršljive specifikacije
   - testne strukture
5. branje specifikacije
6. dopolnitev v okolju C++
7. verifikacija s testnimi strukturami
8. sinteza iz jezika C++

Specifikacija v jeziku C++

- naredimo program, ki naj se obnaša tako kot se bo obnašal sistem
  - specifikacija v obliki programa je nedvoumna
  - izognemo se nedoločenim delom sistema
- specifikacija sistema je v obliki izvršljive datoteke
  - model omogoča zgodnje ocenjevanje zmogljivosti
  - izvršljiva datoteka omogoča testiranje

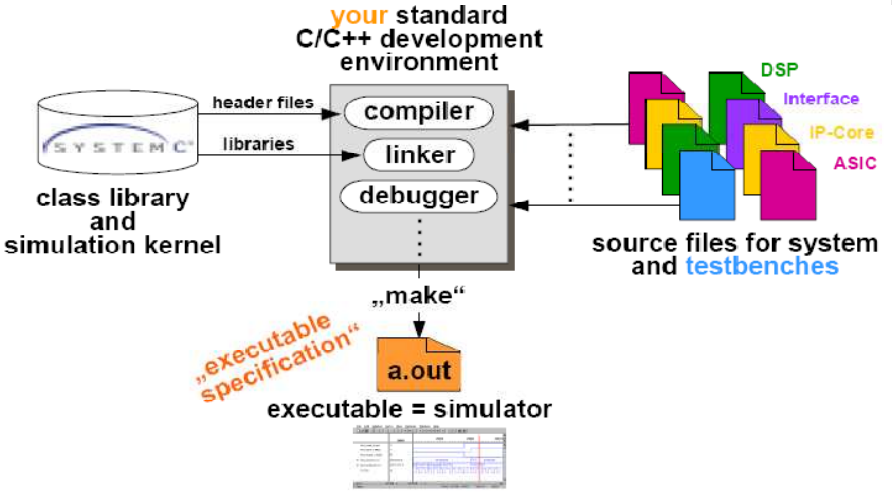
Omejitve jezika C++

- sočasno izvrševanje
  - HW enote se izvajajo sočasno
- modeliranje časa
  - zaporedje dogodkov
- modeliranje HW
  - večvrednostna logika (kot npr. std_logic)
  - modeliranje signalov in priključkov
- reaktivnost sistemov
  - model sistema mora takoj reagirati na spremembe v okolici (spremembe stimlatorjev)

SystemC

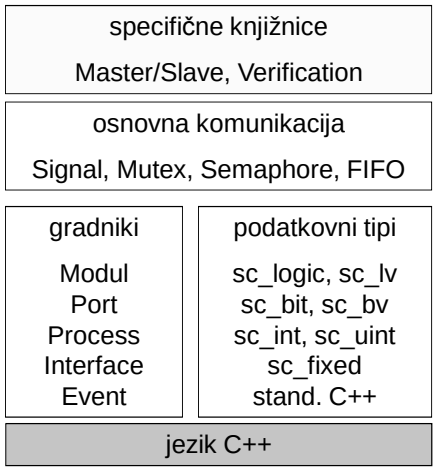
- SystemC je knjižnica v jeziku C++
  - modeliranje sočasnih procesov
  - modeliranje časa in dogodkov
  - HW podatkovni tipi
- SystemC omogoča:
  - abstrakten opis sistema in opis na nivoju ciklov
  - opis HW in SW komponent v enotnem okolju
  - opis HW na nivoju obnašanja in RTL
- potrebujemo standardni C++ prevajalnik
  - gcc, msvcc, bcc..

Uporaba SystemC



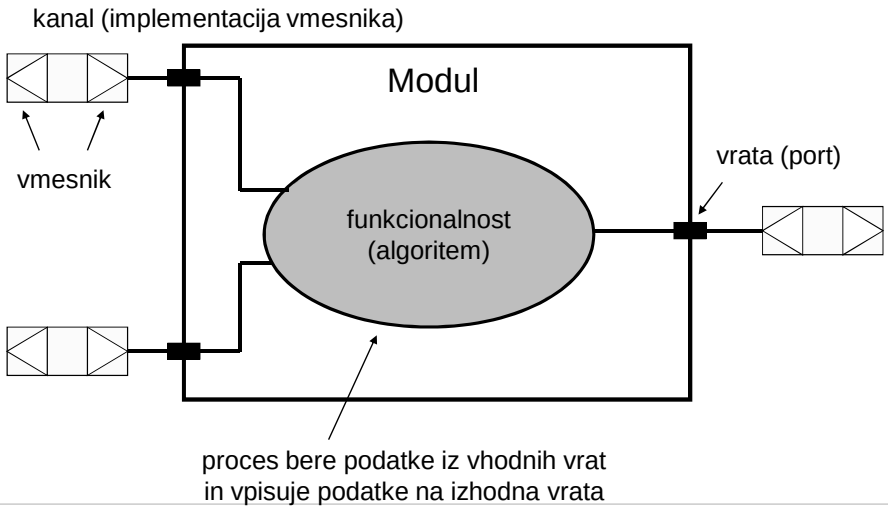
source: KDEC SystemC Overview & Example

Zgradba jezika SystemC



SystemC jedro

Gradniki sistema v SystemC



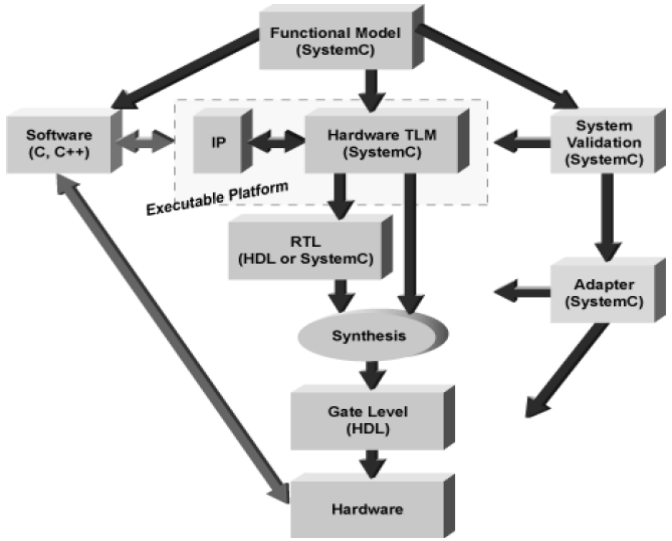
Modeliranje v okolju SystemC

- časovno nedoločen (Untimed)
  - opis vmesnika in funkcionalnosti brez modeliranja časa izvajanja (izvajanje v ničelnem času)
- časovno določen (Timed)
  - modeliranja zakasnitev pri računanju in pri prenosu podatkov oz. komunikaciji
- natančen na nivoju ciklov (Cycle Accurate)
  - delovanje vmesnika je specificirano na nivoju ciklov sistemske ure
- natančen na RT nivoju
  - funkcionalnost je popolnoma časovno določena

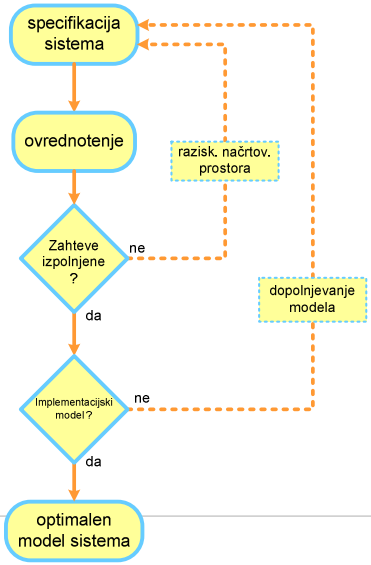
Vrste modelov

- funkcionalni model
  - izvršljiva specifikacija HW in SW za raziskovanje algoritmov (čas. nedoločen model sistema)
  - čas. določena specifikacija za oceno zmogljivosti
- model transakcij (Transaction Level Model)
  - obnašanje in prenos modeliramo s transakcijami
  - časovno določen vmesnik in funkcionalnost
  - na višjem nivoju kot RTL, modeliramo le HW
- model funkc. vodila (Bus Funcitional Model)
  - vmesnik določen na nivoju ciklov
  - primeren za simulacijo procesorjev

Metodologija načrtovanja

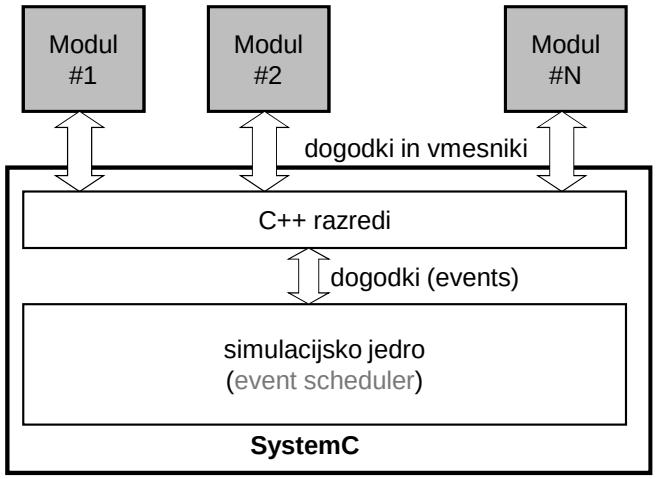


Načrtovalski potek



- začnemo z abstraktnim funkcionalnim modelom
- izvajamo ovrednotenje in raziskovanje variant
- dopolnimo model s podrobnostmi
- implementacijski model

Moduli v okolju SystemC



Prvi koraki v SystemC

- vsak program v C/C++ ima funkcijo **main()**, v SystemC pa je začetna točka **sc_main()**
  - funkcija **main()** je def. v knjižnici in izvede inicializacijo simulacijskega jedra
- ko se izvede **sc_main()** se zgradijo in med seboj povežejo opisani gradniki sistema
- z vključitvijo knjižnice “systemc.h” dobimo dostop do vseh razredov in funkcij

Opis modula

```
SC_MODULE(moj_modul)
{
  // vrata, notranji signali, procesi

  SC_CTOR(moj_modul)     // konstruktor
  {

   // registracija procesov, občutljivostni seznam

  }
};
```

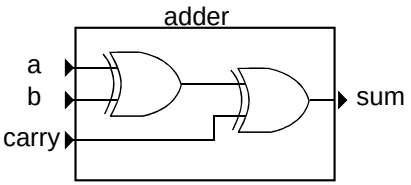
modul definiramo v datoteki “moj_modul.h”

- funkcionalnost je opisana s procesi znotraj modula

opomba: SC_MODULE je makro za:
struct module_name : sc_module

Primer: model logičnega vezja

- model v dveh korakih
  1. model vrat XOR
  2. model seštevalnika
- naredimo module, ki vsebujejo procese
- osnovna oblika procesa: SC_METHOD
  - funkcija, ki jo simulator pokliče ob spremembi signalov iz občutljivostnega seznama



Model logičnih vrat XOR

```
SC_MODULE(xor)
{
  sc_in<bool> A, B;
  sc_out<bool> C;
  void izvedi() { C = A^B; }  // C++ funkcija

  SC_CTOR(xor)     // konstruktor
  {
    SC_METHOD(izvedi);   // registracija funkcije
    sensitive << A << B;      // občutljivostni seznam
  }
};
```

opomba: funkcijo izvedi bi lahko napisali tudi takole:
C.write(A.read() ^ B.read());

Razlaga modula xor

- gradnike opisujemo z razredom sc_module
  - naredili smo razredni objekt z imenom xor
- vrata povezujejo modul z ostalimi gradniki
  - uporabili smo v knjižnici definirana vrata sc_in<> in sc_out<>
- napisali smo funkcijo, ki opisuje delovanje
  - prebere vrednosti A in B, izračuna xor (^) ter vpiše rezultat na vrata C
- v konstruktorju registriramo funkcijo
  - funkcijo povežemo s simulacijskim jedrom

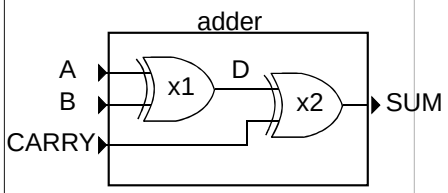
Model seštevalnika

```
#include “systemc.h”
#include “xor.h”

SC_MODULE(adder)
{
  sc_in<bool> A, B, CARRY;
  sc_out<bool> SUM;

  xor x1, x2;
  sc_signal<bool> D;

  SC_CTOR(adder): x1(“X1”), x2(“X2”)
  {
    x1.A(A);  x1.B(B);  x1.C(D);
    x2 << D << CARRY << SUM;
  }
};
```



Priprava simulacije: stimulatorji

```
SC_MODULE(stim)
{
 sc_out<bool> A, B, CARRY;
 sc_in_clk Clk;

 void StimGen()
 {
   A=false;   B=false;  CARRY=false;   wait();
   A=true;    B=true;   CARRY=false;   wait();
   sc_stop();
 }
 SC_CTOR(stim)
 {
   SC_CTHREAD(StimGen, Clk.pos());
 }
};
```

Glavni program sc_main()

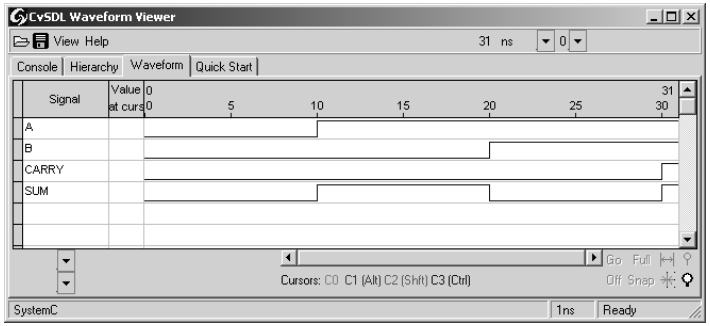
```
int sc_main(int argc, char* argv[])
{
 sc_signal<bool> A, B, CY, SUM;
 sc_clock TestClk("TestClock", 10, SC_NS, 0.5);

 stim Stim1("Stimulus");                    // stimulatorji
 Stim1 << A << B << CY << TestClk;

 adder Add("exor2");                        // testirani modul
 Add << A << B << CY << SUM;

 sc_trace_file *Tf;                         // sledenje signalom (VCD)
 Tf = sc_create_vcd_trace_file("wave");
 sc_trace(Tf, A, "A");
 sc_trace(Tf, B, “B");
 ...
```

Rezultati izvedbe simulacije



- izpisi stanj in statistike v konzoli
- signali v obliki VCD (Value Change Dump)

Povzetek

- sistemski jeziki so nastali zaradi potrebe po hkratnem načrtovanju SW in HW
- knjižnica SystemC dodaja k jeziku C++ konstrukte za opis HW
- SystemC je namenjen opisu sistemov na višjih nivojih abstrakcije
  - ni primeren za podrobno modeliranje logike
- obstajajo programi za sintezo (behavioral) iz jezika SystemC (Agility Compiler, CoCentric, Cynthesizer
  - uporabljati smemo le del konstruktov jezika

Povzetek (II)

- SystemC je prosto dostopen (www.systemc.org) in standardiziran IEEE 1666-2005
- uporablja standardna C++ razvojna orodja
  - lahko povežemo z drugimi orodji (npr. Modelsim)
- razvoj
  - podpora modeliranju operacijskih sistemov
  - podpora različnim računskim modelom
- večina orodij za modeliranje sistemov temelji na jeziku C/C++
  - SpecC (dodatni konstrukti jezika C)
  - Bluespec (nadgradnja SystemC za avto. sintezo)
  - Metropolis (splošni računski modeli in prev. za SystemC)