COMP 522 – Modeling and Simulation by Professor Hans Vangheluwe

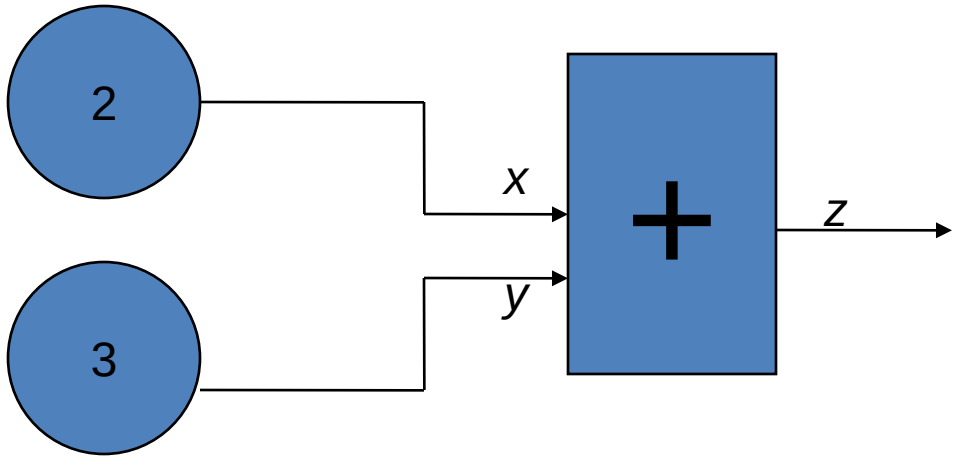
An optimizing compiler for hierarchical Causal Block Diagrams. Application: the PHYSBE benchmark model of the human circulatory system.

Nurudeen Lameed
McGill University
Fall 2008

Contents

- Introduction - CBD
- Optimizing compiler.
- PHYSBE – Physiological simulation benchmark experiment.

Causal Block Diagram (CBD)



≡

$z = x + y;\ \ x = 2,\ y=3;\ z = 5;$
$x, y, z \in \mathbb{R}$
$f: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
(semantic mapping)

- A graph of connected operational blocks.
- Connections denote signals.
- Blocks may be algebraic (Adder, Product) or Time-delay based (Delay, Integrator, Derivatives). Order of blocks is important – cause & effect.

Causal Block Diagram (CBD)

- Concrete syntax (block-diagram) is transformed into abstract syntax.
- Abstract syntax transformed to semantic syntax and semantic mapping – the meaning of the graph (CBD).

Issues

- Non-varying computations are often recomputed adding to performance overheads.
- Non-trivial scientific problems usually involve intensive and large number of computations.
- Optimization must not change the meaning of a model. Still, optimization possible.

Opportunities for optimization: leveraging the dependency graph.

- Constant propagation may improve performance.
- Redundant computations may be eliminated - loop invariants; expressions may be separated into parts: constant, invariant and variable. Only variables need to be recomputed.
- Compiled code typically runs faster than interpreted code.

Examples:

- x, y are constants; z = x + y

  for iteration = 1 to 60000

    z.signal[iteration] = x + y

- Optimized execution:

  z.signal[0] = x + y

  for iteration = 2 to 60000

    z.signal[iteration] = z.signal[0]

Examples: Re-association

- We can exploit algebraic properties; e.g., addition is commutative …
- x, y, z are constants; a, b are variables:

```
for iteration = 1 to 100000
    s.signal[iteration] = x + y + z + a + b
```

- Optimized execution:

```
temp = x + y + z
for iteration = 1 to 100000
    s.signal[iteration] = temp + a + b
```

- Savings: 2 * 100000 – 2 additions

Optimizing compiler for CBD

- Transforms CBDs into some optimized C programs.
- CBDs may have cycles: uses optimized LAPACK library for solving systems of linear equations, where necessary.
- Redundant computations are eliminated without loss of accuracy.

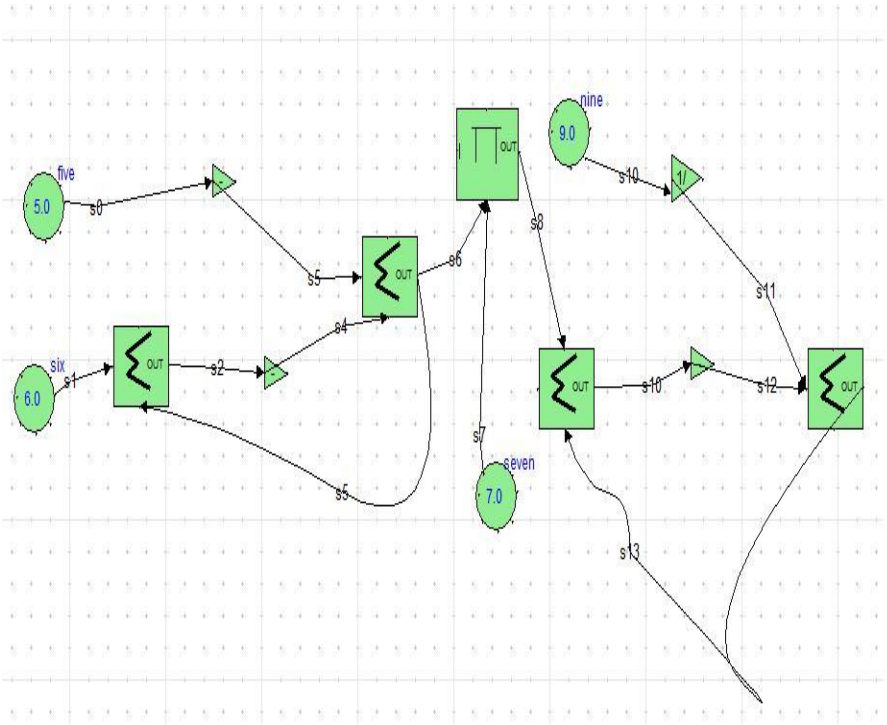
Model with 2 linear loops

```
result[1][0] = -5.0;
result[2][0] = 6.0;

/* solve the system of linear equations */
double _a1[9] = {1,-1,0,0,1,-1,1,0,1};
double _b1[3] = {0+6.0,0+result[1][0],0};
int ipiv_b1[3];
int _b1_res = clapack_dgesv(CblasRowMajor, 3, 1, _a
if (_b1_res != 0 )
{
  printf("Matrix:_b1 Error code:: %d; cannot solve
  exit(1);
}
result[3][0] = _b1[0];
result[4][0] = _b1[1];
result[5][0] = _b1[2];
result[6][0] = 9.0;
result[7][0] = 7.0;
result[8][0] = 1/9.0;
result[9][0] = result[4][0]*7.0;

/* solve the system of linear equations */
double _a2[9] = {1,-1,0,0,1,-1,1,0,1};
double _b2[3] = {0+result[9][0],0+result[8][0],0};
int ipiv_b2[3];
int _b2_res = clapack_dgesv(CblasRowMajor, 3, 1, _a
if (_b2_res != 0 )
{
  printf("Matrix:_b2 Error code:: %d; cannot solve
  exit(1);
}
result[10][0] = _b2[0];
result[11][0] = _b2[1];
result[12][0] = _b2[2];

int i;
for (i = 1; i < N; ++i)
{
  result[0][i] = 5.0;
  result[1][i] = result[1][0];
  result[2][i] = 6.0;
```



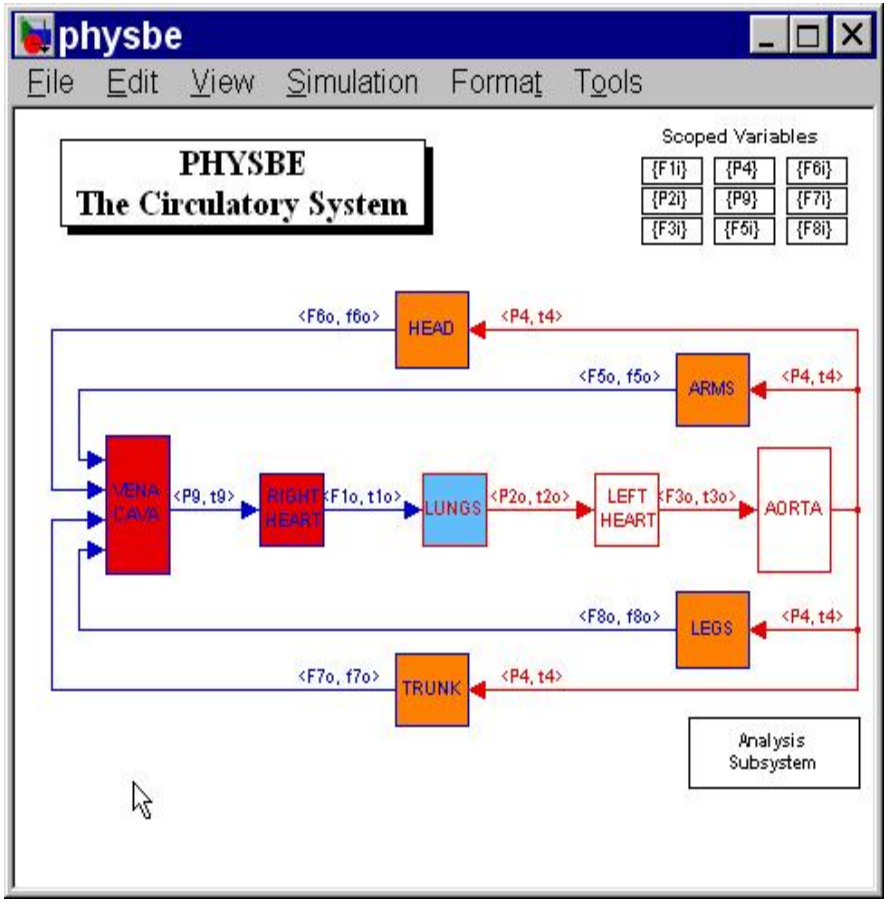
PHYSBE: A Physiological simulation benchmark

- A classical model of

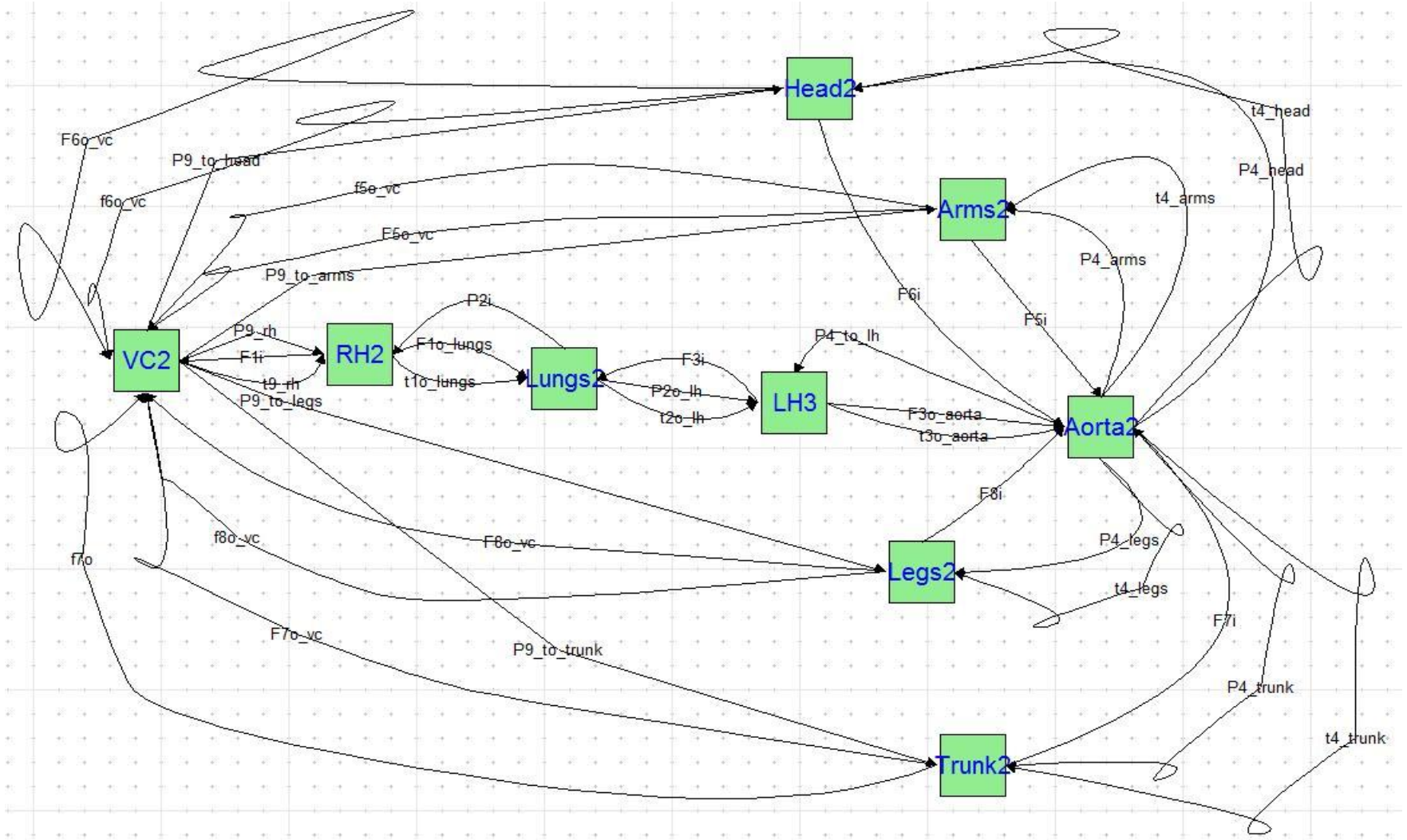
the human circulatory system (John McLeod, 1966).

- An example of a non-linear continuous system. Linearization through approximation.

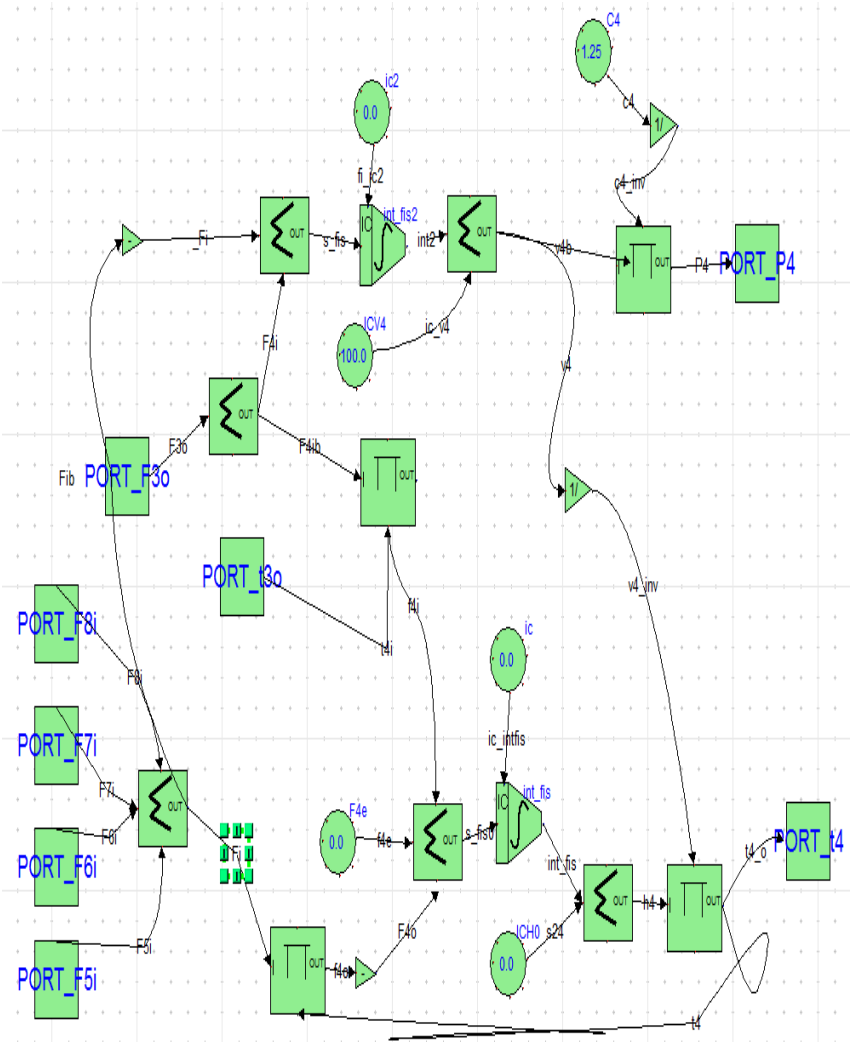
- The mathworks: Simulink

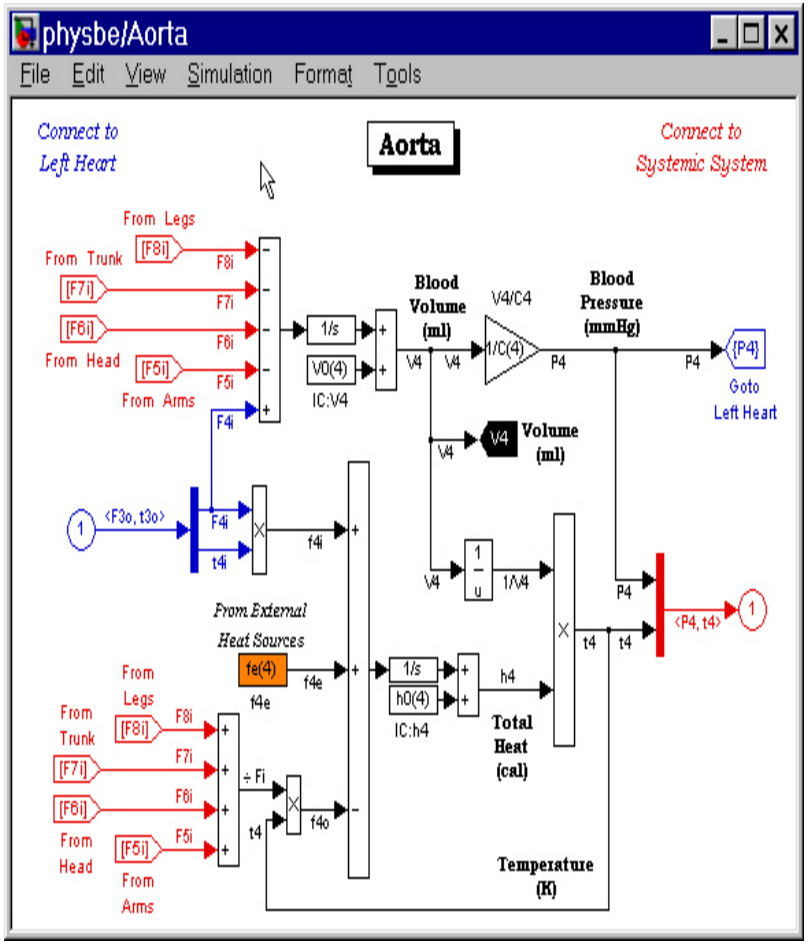


PHYSBE: Equivalent model in AToM^3.

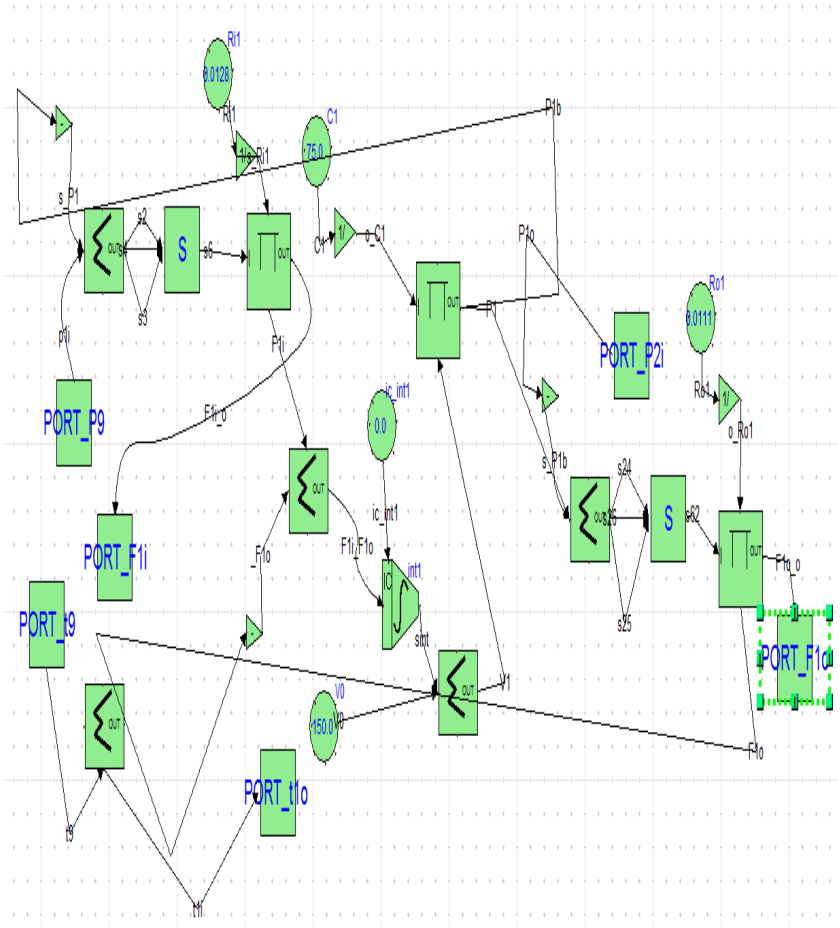


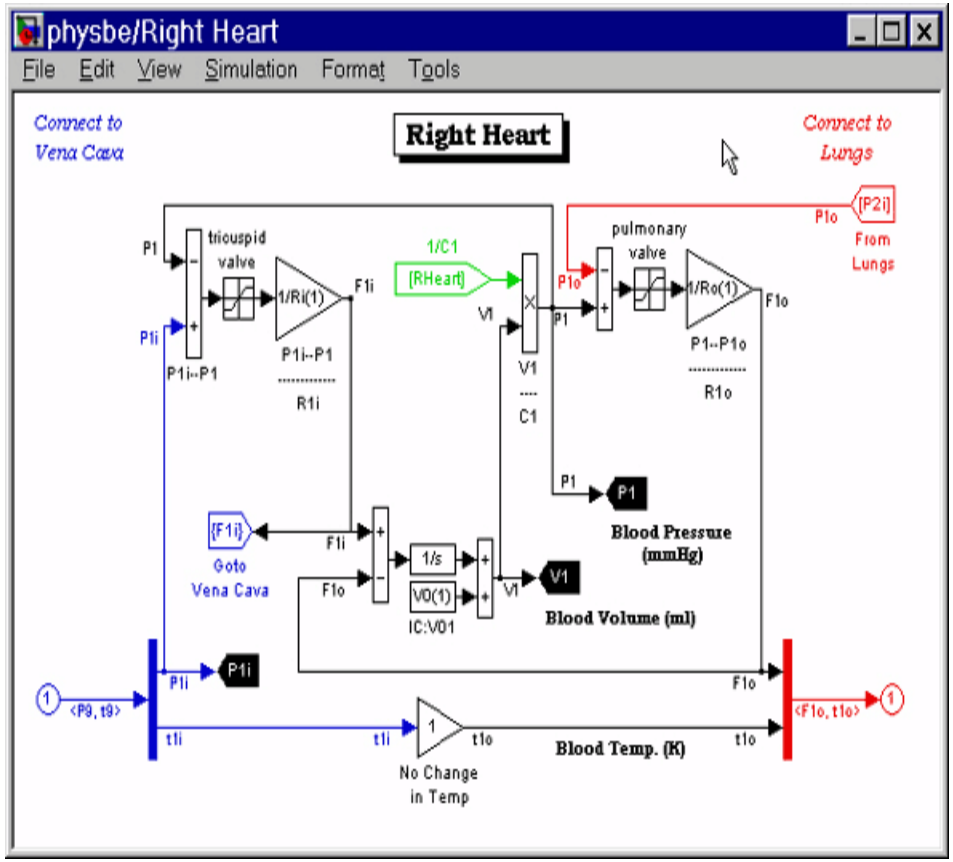
Aorta:


C4
1.25
ic2
0.0
fi_ic2
int_fis2
IC
int2
OUT
s_fis
Fi
P4
PORT_P4
ICV4
100.0
ic_v4
F4i
F3o
F4ib
Fib
PORT_F3o
PORT_t3o
PORT_F8i
PORT_F7i
PORT_F6i
PORT_F5i
F8i
F7i
F6i
F5i
ic
ic_intfis
int_fis
F4e
f4e
s_fis0
t4_o
PORT_t4
ICH0
s24
h4
F4o
f4o
t4
v4
v4_inv
1/
c4_inv

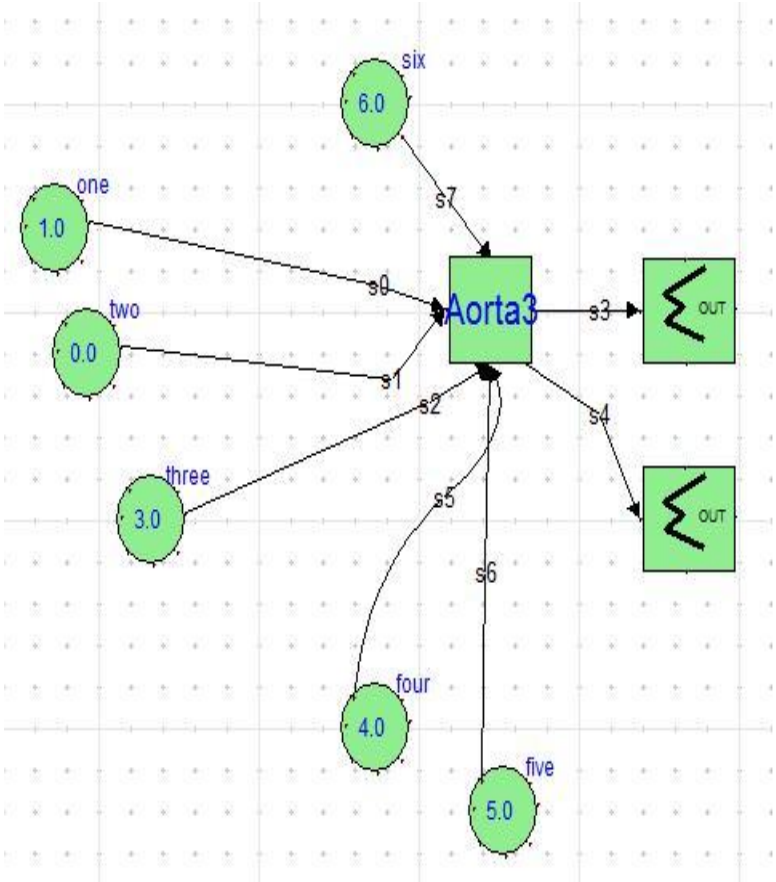

physbe/Aorta
File Edit View Simulation Format Tools
Aorta
Connect to Left Heart
Connect to Systemic System
From Legs
From Trunk
From Head
From Arms
[F8i]
[F7i]
[F6i]
[F5i]
F8i
F7i
F6i
F5i
F4i
1/s
V0(4)
IC:V4
Blood Volume (ml)
V4
V4/C4
1/C(4)
Blood Pressure (mmHg)
P4
{P4}
Goto Left Heart
Volume (ml)
<F3o, t3o>
f4i
t4i
From External Heat Sources
fe(4)
f4e
h0(4)
IC:h4
h4
Total Heat (cal)
÷ Fi
f4o
t4
<P4, t4>
Temperature (K)

Right Heart:


Ri1
0.0128
C1
75.0
Ro1
0.0111
PORT_P9
PORT_P2i
PORT_F1i
PORT_t9
PORT_F1o
PORT_t1o
P1b
P1o
P1i
F1i_0
ic_int1
0.0
F1i,F1o
int1
s_P1b
s24
s25
s62
V0
150.0
V1
F1o_o
t9
t1i
F1o


physbe/Right Heart
File Edit View Simulation Format Tools
Right Heart
Connect to Vena Cava
Connect to Lungs
tricuspid valve
1/Ri(1)
P1i-P1
R1i
1/C1
[RHeart]
V1
C1
pulmonary valve
1/Ro(1)
P1-P1o
R1o
[P2i]
From Lungs
{F1i}
Goto Vena Cava
1/s
V0(1)
IC:V01
Blood Pressure (mmHg)
Blood Volume (ml)
<P9, t9>
<F1o, t1o>
No Change in Temp
Blood Temp. (K)

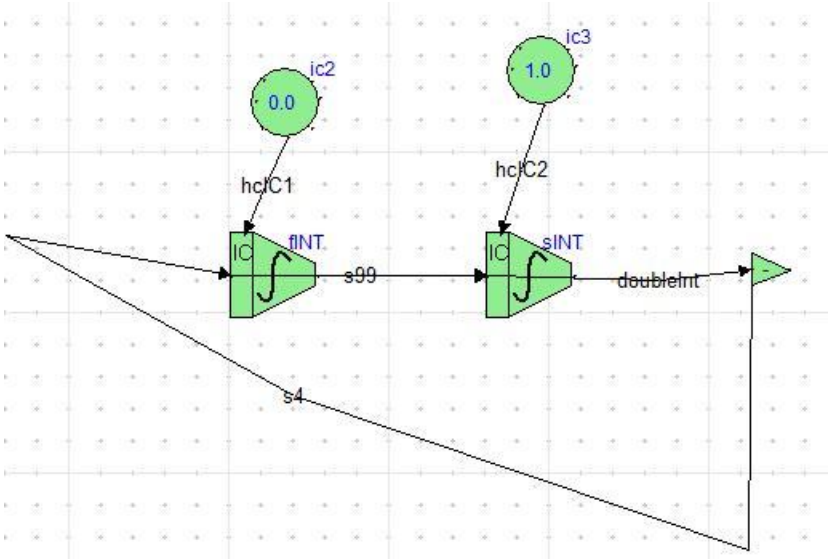
Generated C – code for Aorta



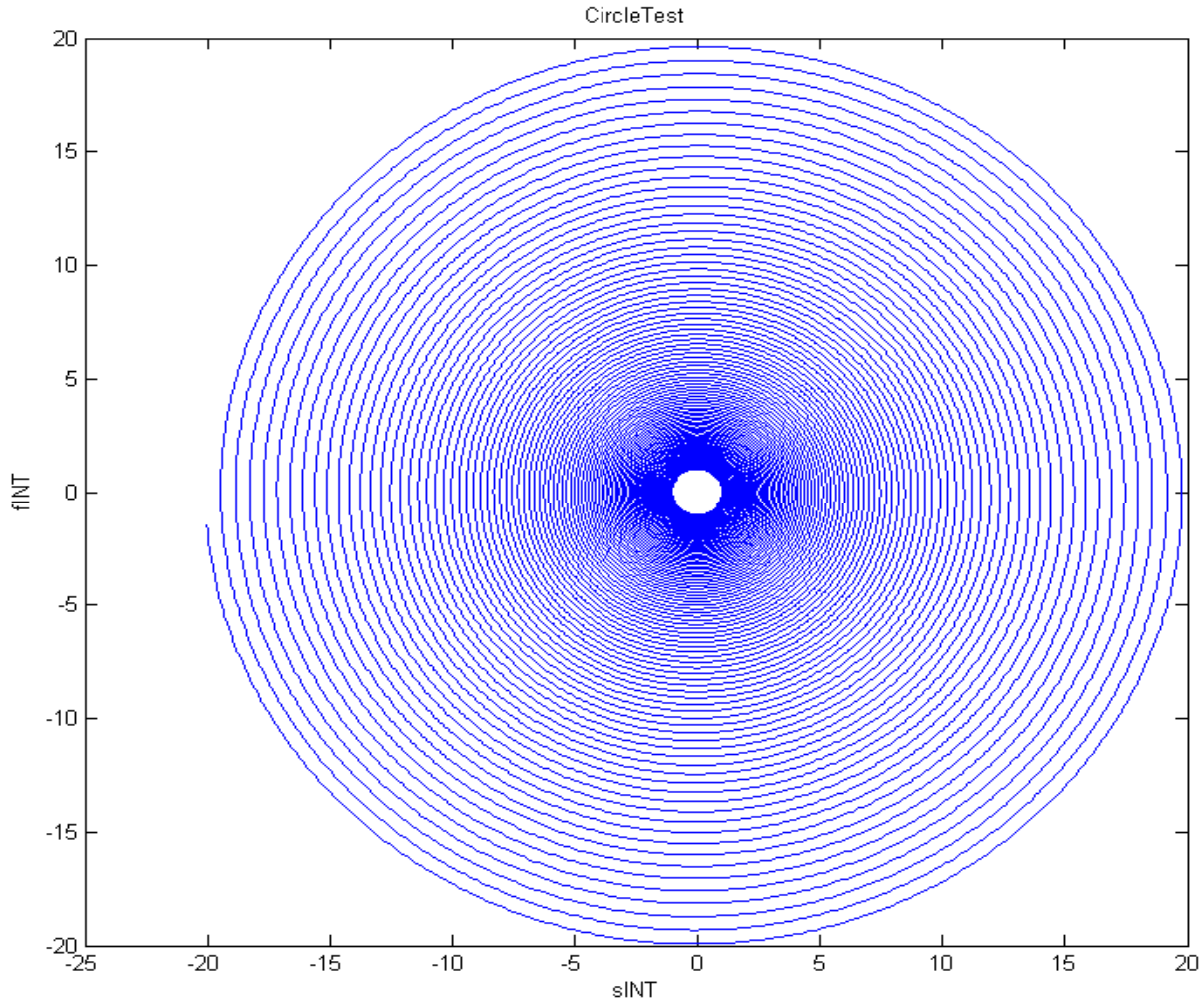

```
for (i = 1; i < N; ++i)
{
  result[0][i] = 1.0;
  result[1][i] = 0.0;
  result[2][i] = 3.0;
  result[3][i] = 4.0;
  result[4][i] = result[4][0];
  result[5][i] = 0.0;
  result[6][i] = 0.0;
  result[7][i] = result[7][i-1] + result[22][i-1] *0.01;
  result[8][i] = result[7][i]+0.0;
  result[9][i] = 0.0;
  result[10][i] = result[10][i-1] + result[24][i-1] *0.01;
  result[11][i] = 100.0;
  result[12][i] = result[10][i]+100.0;
  result[13][i] = 1/result[12][i];
  result[14][i] = result[8][i]*result[13][i];
  result[15][i] = result[4][0]*result[14][i];
  result[16][i] = -result[15][i];
  result[17][i] = 0.0;
  result[18][i] = 5.0;
  result[19][i] = result[19][0];
  result[20][i] = 6.0;
  result[21][i] = result[21][0];
  result[22][i] = result[16][i]+result[21][0]+0.0;
  result[23][i] = result[23][0];
  result[24][i] = result[24][0];
  result[25][i] = result[14][i];
  result[26][i] = 1.25;
  result[27][i] = result[27][0];
  result[28][i] = result[12][i]*result[27][0];
  result[29][i] = result[28][i];
}
```

More … Circle test

- The Next slide shows the result for circle test using 60,000 iterations.



Circle test result: 60000 iterations.



Conclusion

- Hierarchical ordering in CBD provides opportunities for optimizations.
- Elimination of redundant computations might result in performance gains.

Questions?