MathWorks — MATLAB&SIMULINK

Heterogeneous Function Composition to Eliminate a Class of Direct Relationships in Software Components of Dynamic Systems

Pieter J. Mosterman

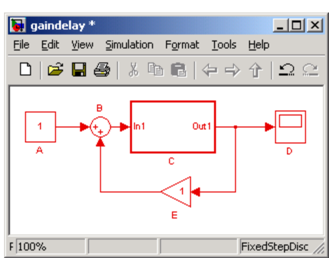
Senior Research Scientist
Design Automation Department
The MathWorks

Adjunct Professor
School of Computer Science
McGill

MathWorks — MATLAB&SIMULINK

The problem!

- Demo … ☺

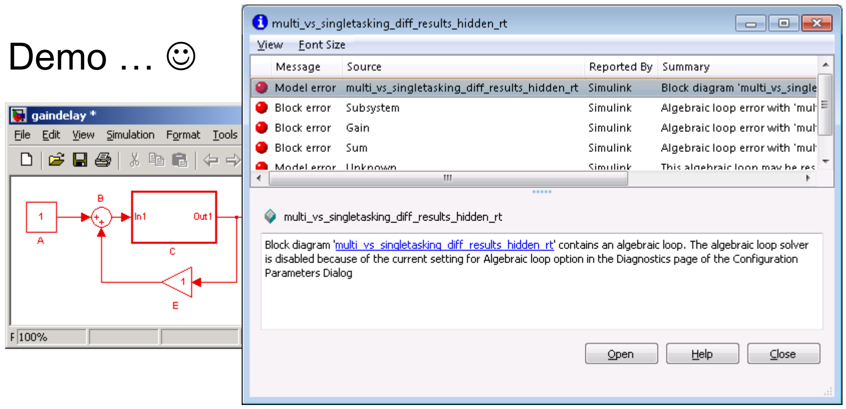


2

MathWorks MATLAB&SIMULINK

The problem!

- Demo … ☺



- What happened?!

3

MathWorks MATLAB&SIMULINK

Agenda

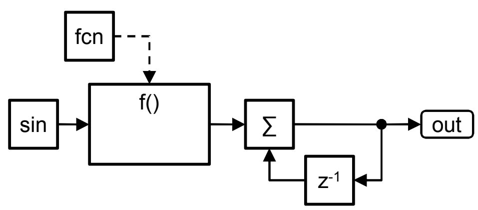
- Simulink preliminaries
- Executing a Simulink model
- Dealing with hierarchy
- Heterogeneous composition
- An implementation in Simulink
- Conclusions

4

MathWorks MATLAB&SIMULINK

Basic Simulink syntax

- An example model …

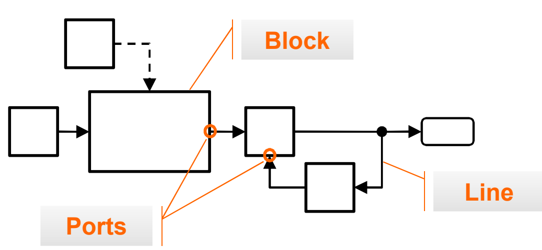


5

MathWorks MATLAB&SIMULINK

Basic Simulink syntax

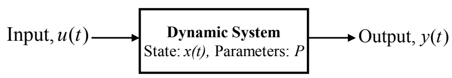
- Network of blocks with directed lines connected between ports



6

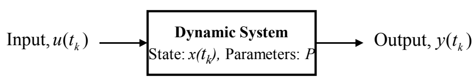
The execution hierarchy

- The lines reflect input/output relations
- The (nonvirtual) blocks are dynamic systems



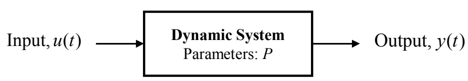
$$\text{Output}: y(t) = f(u(t), x(t), P, t)$$

$$\text{Derivative}: \frac{dx(t)}{dt} = g(u(t), x(t), P, t)$$



$$\text{Output}: y(t_k) = f(u(t_k), x(t_{k-1}), P, t_k)$$

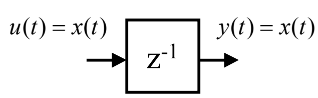
$$\text{Update}: x(t_k) = g(u(t_k), x(t_{k-1}), P, t_k)$$



$$\text{Algebraic}: f(u(t), y(t), P) = 0$$

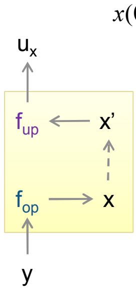
Example block implementation

- A unit delay



State, $x(t)$

$x(0^-) = P_{IC}$



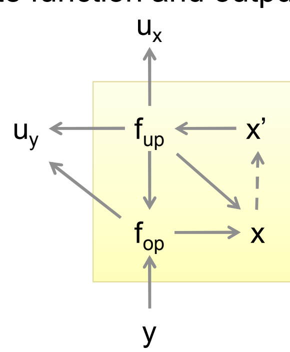

```
class UnitDelayBlock : public Block {
  public:
     ErrorStatus BlockDrawIcon() {
        // Draw '1/z' on the icon
        ..............................
     }
     BlockParameterData BlockGetParameterData() {
        // Return initial_condition as block data
        ..............................
     }
     ErrorStatus BlockOutput(){
        // Implement y(t) = x(t)
        ..............................
     }
     ErrorStatus BlockUpdate(){
        // Implement x(t) = u(t)
        ..............................
     }
  private:
     double initial_condition;

};
```

MathWorks MATLAB&SIMULINK

The basic structure of a discrete time block

- Data dependencies between
  - Input signals, output signals, current state, new state, update function and output function

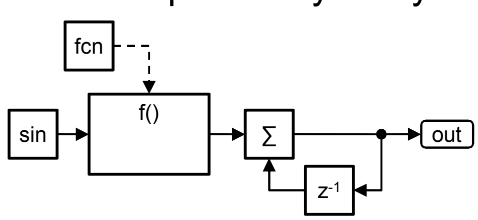


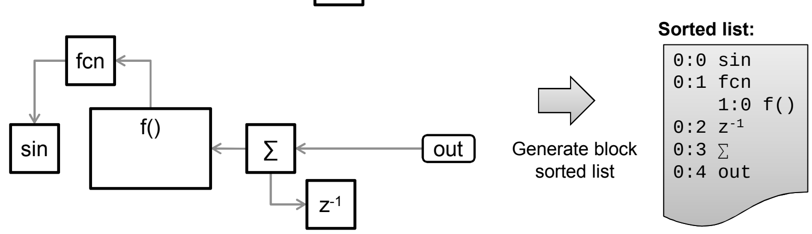
9

MathWorks MATLAB&SIMULINK

Data dependencies to create a sorted list for efficient execution

- Static dependency analysis



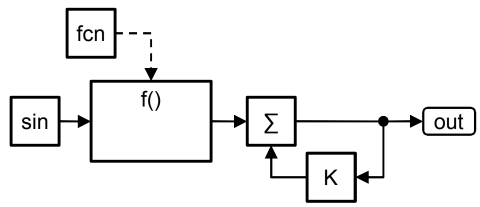


10

MathWorks MATLAB&SIMULINK

Algebraic dependencies

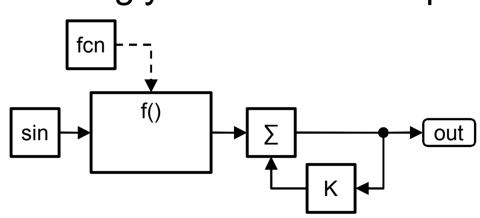
- What if we replace the delay block by a gain?



MathWorks MATLAB&SIMULINK

Algebraic dependencies

- Strongly connected components

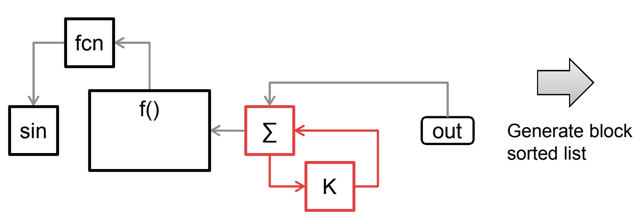


Algebraic loop solution:

$y(t) = u(t) + K\,y(t)$

$\rightarrow$

$y(t) = u(t)/(1-K)$



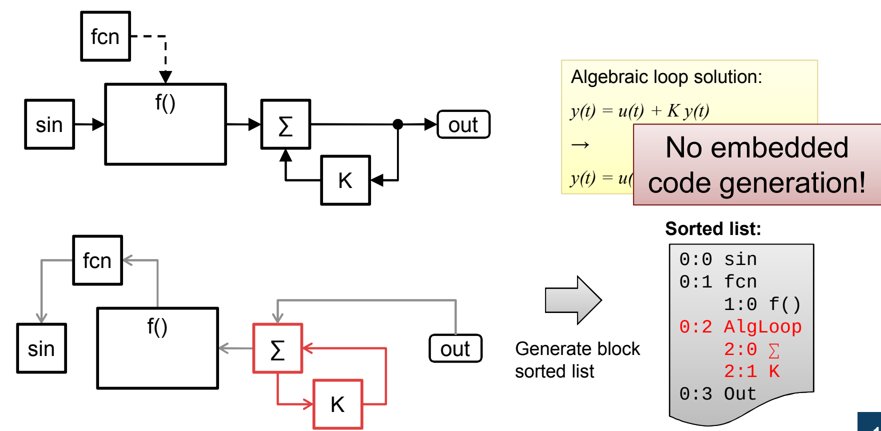
Sorted list:

```
0:0 sin
0:1 fcn
    1:0 f()
0:2 AlgLoop
    2:0 ∑
    2:1 K
0:3 Out
```

MathWorks MATLAB&SIMULINK

Algebraic dependencies

- Strongly connected components



13

MathWorks MATLAB&SIMULINK

Agenda

- Simulink preliminaries
- Executing a Simulink model
- Dealing with hierarchy
- Heterogeneous composition
- An implementation in Simulink
- Conclusions

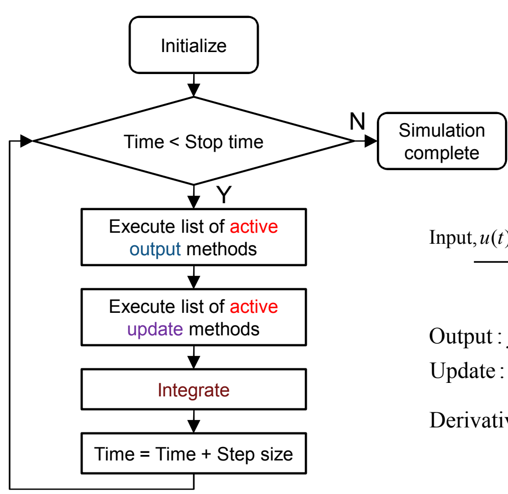
14

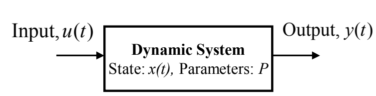
Let's assume single-tasking execution

- Allows multi-rate systems
- All blocks run in a single task
  - Single execution time line
  - Base rate at greatest common denominator
  - Blocks execute when they have a sample hit
  - No data integrity issues

15

A simulation algorithm for networks of dynamic systems





$$\text{Output}: y(t) = f_o(u(t), x(t), P, t)$$

$$\text{Update}: x_d(t) = f_u(u(t), x(t), P, t)$$

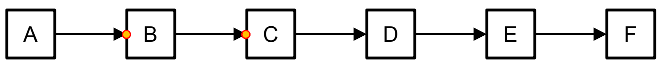
$$\text{Derivative}: \frac{dx_c(t)}{dt} = f_d(u(t), x(t), P, t)$$

16

MathWorks MATLAB&SIMULINK

First generate the sorted list

- Determine data dependencies
  - Based on a direct feedthrough (df) flag

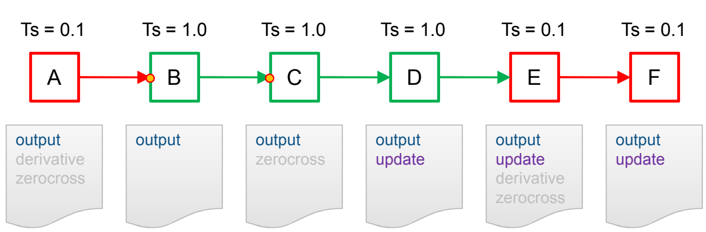



```
0:0 F
0:1 E
0:2 D
0:3 A
0:4 B
0:5 C
```

17

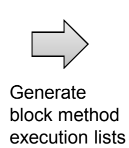
MathWorks MATLAB&SIMULINK

Generate execution lists for all of the block methods



Sorted list:

```
0:0 F
0:1 E
0:2 D
0:3 A
0:4 B
0:5 C
```



Output execution list:

```
0:0 F (0.1)
0:1 E (0.1)
0:2 D (1.0)
0:3 A (0.1)
0:4 B (1.0)
0:5 C (1.0)
```

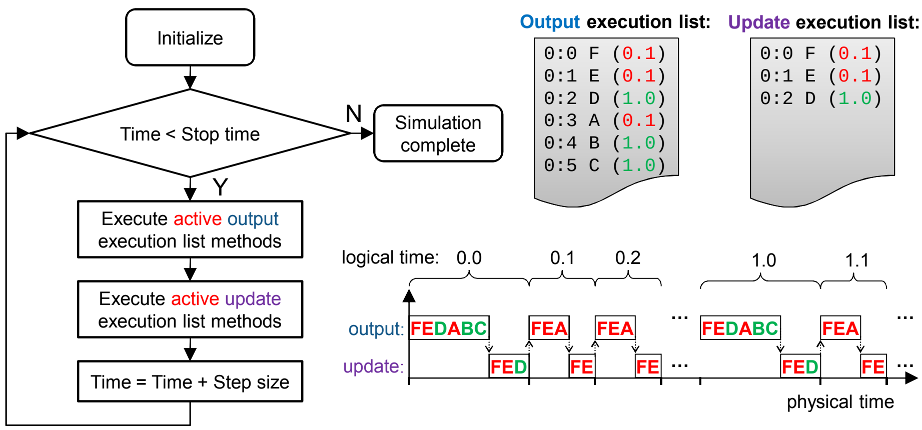
Update execution list:

```
0:0 F (0.1)
0:1 E (0.1)
0:2 D (1.0)
```

18

MathWorks MATLAB&SIMULINK

Repeatedly evaluate the execution lists



19

MathWorks MATLAB&SIMULINK

Agenda

- Simulink preliminaries
- Executing a Simulink model
- Dealing with hierarchy
- Heterogeneous composition
- An implementation in Simulink
- Conclusions

20

MathWorks MATLAB&SIMULINK

Hierarchy

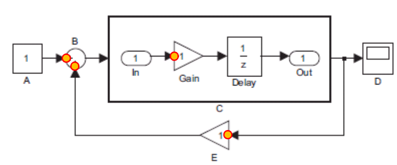
- Virtual blocks to organize graphical hierarchy
  - Referential transparency; no semantic bearing
- Nonvirtual blocks to organize
  - Execution hierarchy
  - Data scope hierarchy

21

MathWorks MATLAB&SIMULINK

A graphical hierarchy

- Group blocks in a virtual subsystem
  - Subsystem C does not appear in the sorted list



Sorted list:

```
0:0 Delay
0:1 D
0:2 A
0:3 E
0:4 B
0:5 Gain
```

Generate block method execution lists

Output execution list:

```
0:0 Delay
0:1 D
0:2 A
0:3 E
0:4 B
0:5 Gain
```

Update execution list:

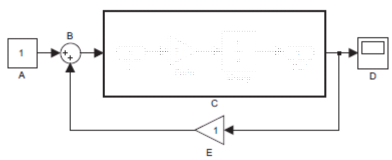
```
0:0 Delay
```

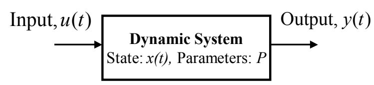
22

MathWorks MATLAB&SIMULINK

Let's create a component …

- Make the virtual subsystem nonvirtual
  - Becomes a dynamic system in its own right





$$\text{Output}: y(t) = f_o(u(t), x(t), P, t)$$

$$\text{Update}: x_d(t) = f_u(u(t), x(t), P, t)$$

$$\text{Derivative}: \frac{dx_c(t)}{dt} = f_d(u(t), x(t), P, t)$$

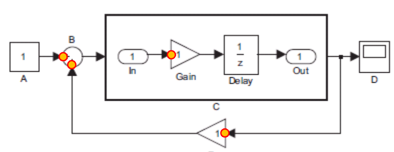
  - Does that affect sorting?

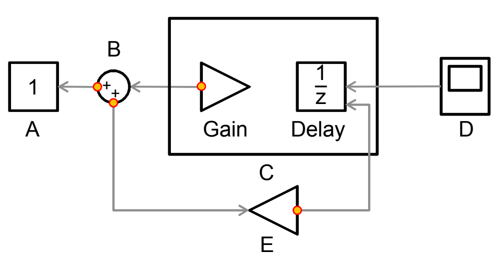
23

MathWorks MATLAB&SIMULINK

Let's start with a virtual subsystem …

- Dependencies derived from df flag



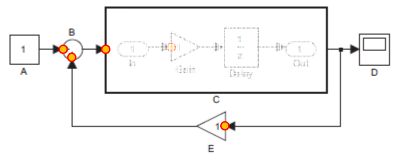


24

MathWorks MATLAB&SIMULINK

Make subsystem an atomic component

- Direct feedthrough moves to component level!

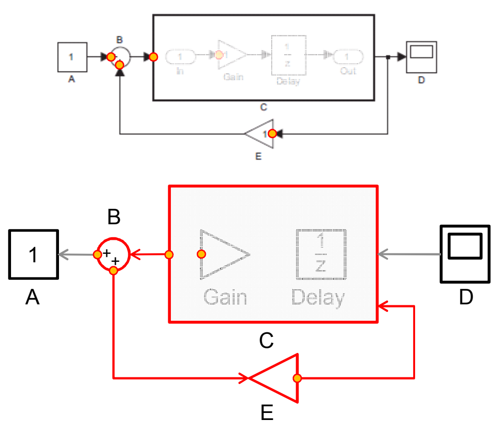


25

MathWorks MATLAB&SIMULINK

Make subsystem an atomic component

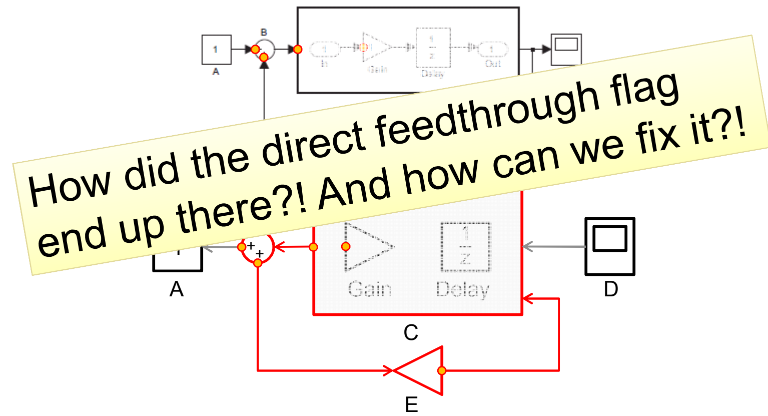
- Now we have a dependency cycle!



26

Make subsystem an atomic component

- Now we have a dependency cycle!



27

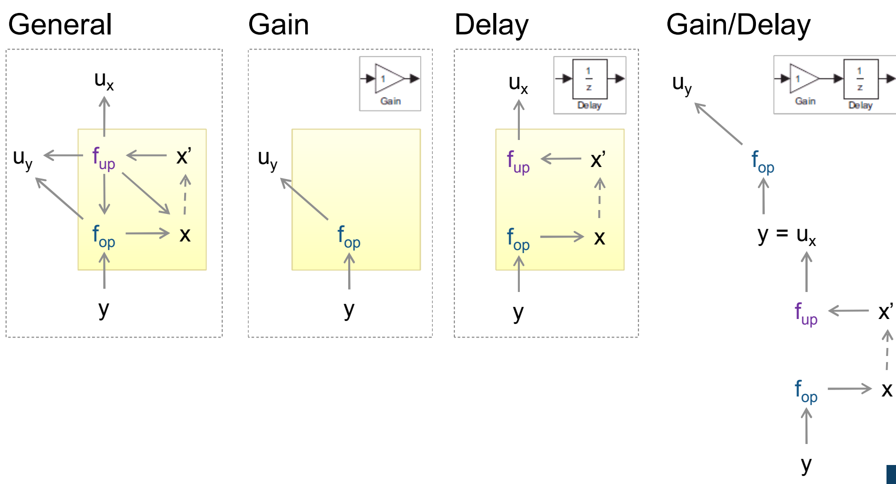
Agenda

- Simulink preliminaries
- Executing a Simulink model
- Dealing with hierarchy
- Heterogeneous composition
- An implementation in Simulink
- Conclusions

28

MathWorks MATLAB&SIMULINK

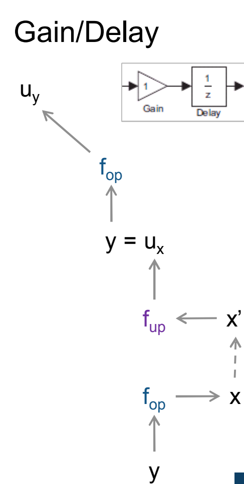
Gain followed by delay



29

MathWorks MATLAB&SIMULINK

Gain followed by delay

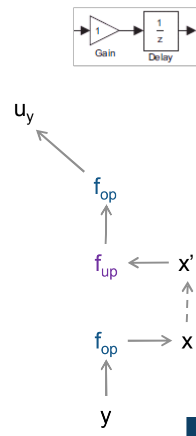


30

MathWorks MATLAB&SIMULINK

Gain followed by delay

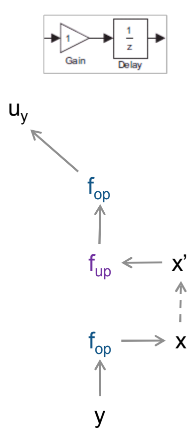
Gain/Delay



31

MathWorks MATLAB&SIMULINK

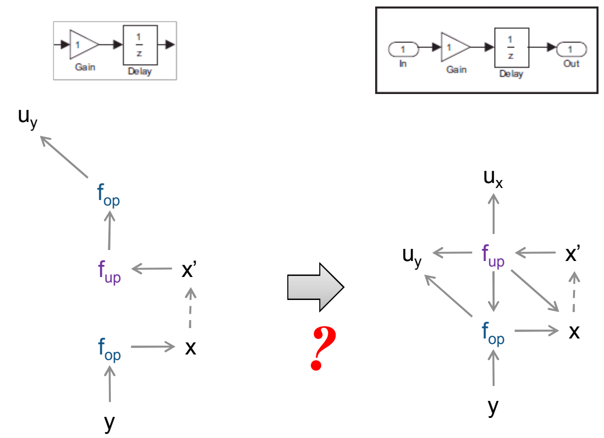
Gain followed by delay



32

MathWorks MATLAB&SIMULINK

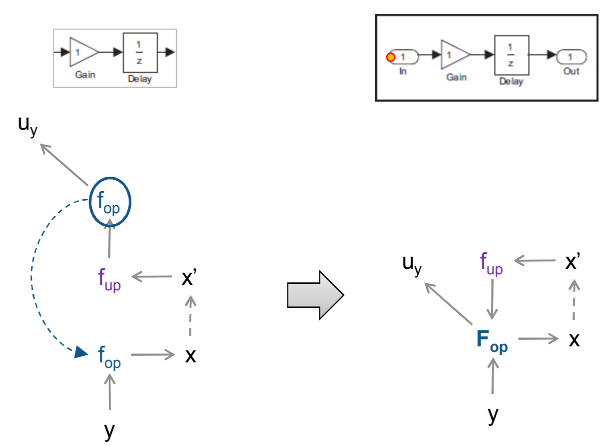
How do we create a component?



33

MathWorks MATLAB&SIMULINK

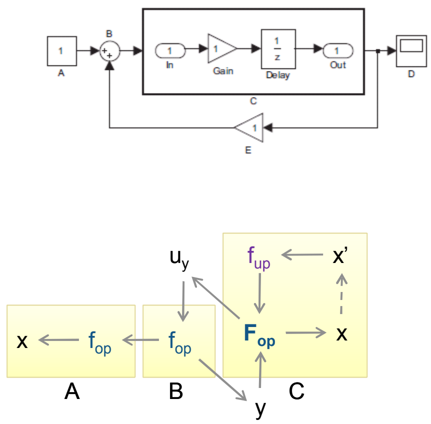
Homogeneous composition



34

MathWorks MATLAB&SIMULINK

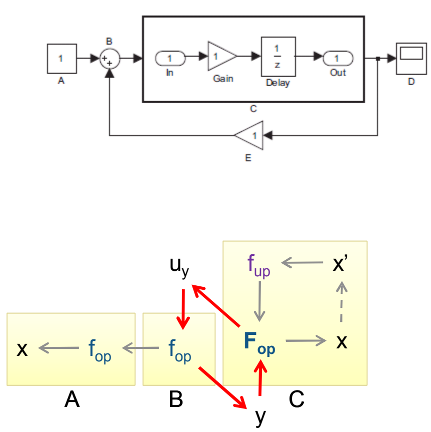
Put the component in context



35

MathWorks MATLAB&SIMULINK

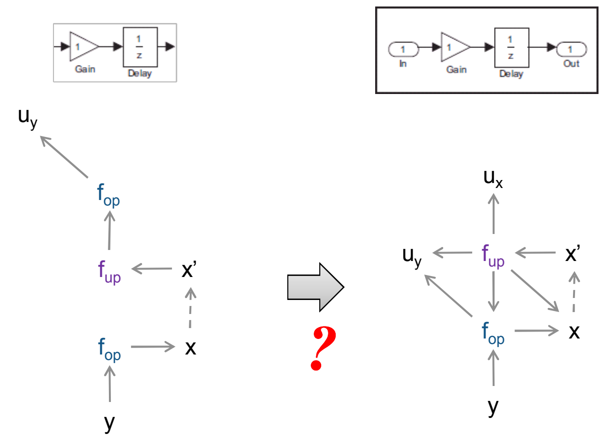
We have a dependency cycle!



36

MathWorks MATLAB&SIMULINK

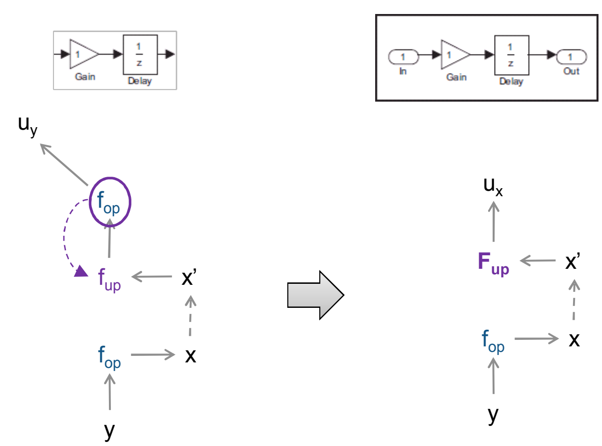
Is there another way … ?



37

MathWorks MATLAB&SIMULINK

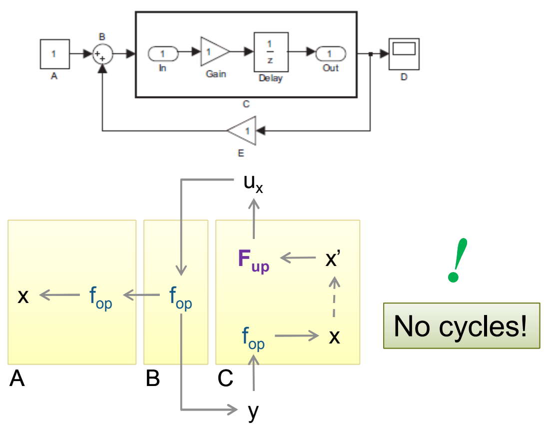
Heterogeneous composition!



38

MathWorks MATLAB&SIMULINK

Putting it in context again



39

MathWorks MATLAB&SIMULINK

Agenda

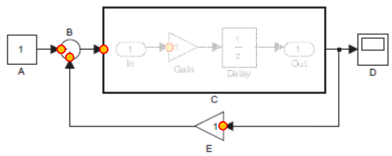
- Simulink preliminaries
- Executing a Simulink model
- Dealing with hierarchy
- Heterogeneous composition
- An implementation in Simulink
- Conclusions

40

MathWorks MATLAB&SIMULINK

How do we fit this into the model compilation machinery?

- Create sorted list after clearing df flag

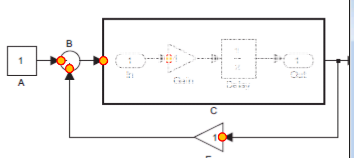


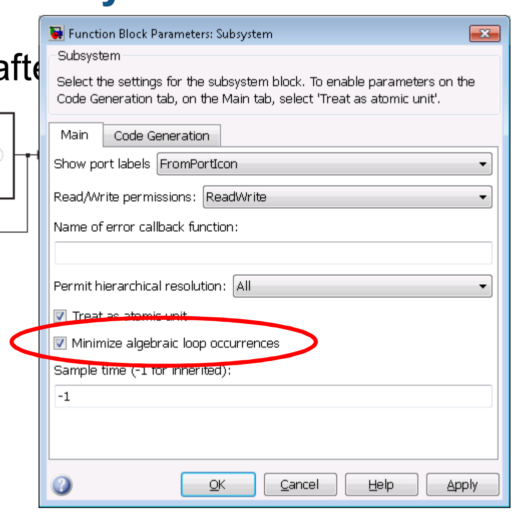
41

MathWorks MATLAB&SIMULINK

How do we fit this into the model compilation machinery?

- Create sorted list afte

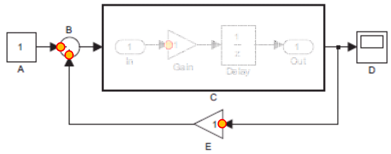




42

How do we fit this into the model compilation machinery?

- Create sorted list after clearing df flag



Sorted list:

```
0:0 C
    0:0 Gain
    0:1 Delay
0:1 A
0:2 E
0:3 B
```

- Mark where output for update should be used

Sorted list:

```
0:0 C
    0:0 Gain•
    0:1 Delay
0:1 A
0:2 E
0:3 B
```

Generate block method execution lists

Output execution list:

```
0:0 C(op)
    0:0 Delay(op)
0:1 A(op)
0:2 E(op)
0:3 B(op)
```

Update execution list:

```
0:0 C(up)
    0:0 Gain(op)
    0:1 Delay(up)
```


What does the generated code look like?

```
/* Model output function */
void gaindelay_output(void)
{
  real_T rtb_E;

  /* Outputs for atomic SubSystem: '<Root>/C' */
  gaindelay_C();

  /* end of Outputs for SubSystem: '<Root>/C' */

  /* Gain: '<Root>/E' */
  rtb_E = gaindelay_P.E_Gain * gaindelay_B.Delay;

  /* Sum: '<Root>/B' incorporates:
   * Constant: '<Root>/A'
   */
  gaindelay_B.B = gaindelay_P.A_Value + rtb_E;
}
```

Output execution list:

```
0:0 C(op)
    0:0 Delay(op)
0:1 A(op)
0:2 E(op)
0:3 B(op)
```

Update execution list:

```
0:0 C(up)
    0:0 Gain(op)
    0:1 Delay(up)
```

MathWorks MATLAB&SIMULINK

What does the generated code look like?

```
/* Model output function */
void gaindelay_output(void)
{
  real_T rtb_E;

  /* Outputs for atomic SubSystem: '<Root>/C' */
  gaindelay_C();

  /* end of Outpu
  
  /* Gain: '<Root
  rtb_E = gaindel

  /* Sum: '<Root>
   * Constant: '<Root>/A'
   */
  gaindelay_B.B = gaindelay_P.A_Value + rtb_E;
}
```

```
/* Outputs for atomic system: '<Root>/C' */
void gaindelay_C(void)
{
  /* UnitDelay: '<S1>/Delay' */
  gaindelay_B.Delay = gaindelay_DWork.Delay_DSTATE;
}
```

Output execution list:

```
0:0 C(op)
    0:0 Delay(op)
0:1 A(op)
0:2 E(op)
0:3 B(op)
```

Update execution list:

```
0:0 C(up)
    0:0 Gain(op)
    0:1 Delay(up)
```

45

MathWorks MATLAB&SIMULINK

What does the generated code look like?

```
/* Update for atomic system: '<Root>/C' */
void gaindelay_C_Update(void)
{
  real_T rtb_Gain;

  /* Gain: '<S1>/Gain' */
  rtb_Gain = gaindelay_P.Gain_Gain * gaindelay_B.B;

  /* Update for UnitDelay: '<S1>/Delay' */
  gaindelay_DWork.Delay_DSTATE = rtb_Gain;
}
```

Output execution list:

```
0:0 C(op)
    0:0 Delay(op)
0:1 A(op)
0:2 E(op)
0:3 B(op)
```

Update execution list:

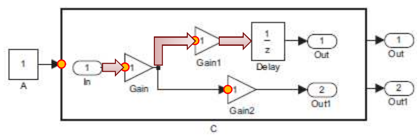
```
0:0 C(up)
    0:0 Gain(op)
    0:1 Delay(up)
```

46

MathWorks MATLAB&SIMULINK

The general analysis

- For an inport block
  - Depth-first search for a df path to output
  - If a port without df is reached, mark visited nodes for potential move of output method
- If output found
  - Clear all blocks visited from the initial input port

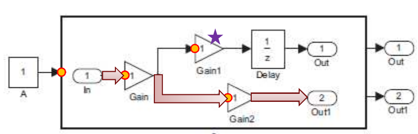


47

MathWorks MATLAB&SIMULINK

The general analysis

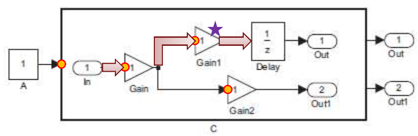
- For an inport block
  - Depth-first search for a df path to output
  - If a port without df is reached, mark visited nodes for potential move of output method
- If output found
  - Clear all blocks visited from the initial input port



48

The general analysis

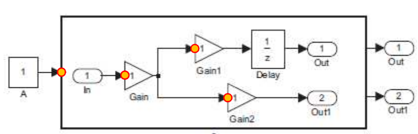
- For an inport block
  - Depth-first search for a df path to output
  - If a port without df is reached, mark visited nodes for potential move of output method
- If output found
  - Clear all blocks visited from the initial input port



MathWorks MATLAB&SIMULINK

The general analysis

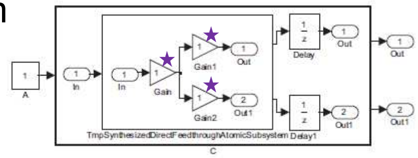
- For an inport block
  - Depth-first search for a df path to output
  - If a port without df is reached, mark visited nodes for potential move of output method
- If output found
  - Clear all blocks visited from the initial input port



MathWorks MATLAB&SIMULINK

The synthesis part of the algorithm

- Move marked df blocks into an atomic subsystem



Sorted list:

```
0:0 A
0:1 C
    1:0 TmpSyn…*
        0:0 Gain
        0:1 Gain1
        0:2 Gain2
    1:1 Delay
    1:2 Delay1
```

Output execution list:

```
0:0 A(op)
0:1 C(op)
    1:0 Delay(op)
    1:1 Delay1(op)
```

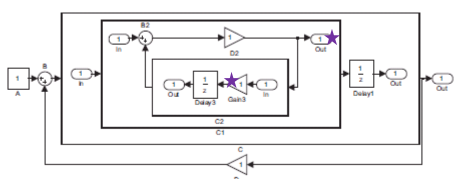
Update execution list:

```
0:0 C(up)
    0:0 TmpSyn…(op)
        0:0 Gain(op)
        0:1 Gain1(op)
        0:2 Gain2(op)
    0:1 Delay(up)
    0:2 Delay1(up)
```

51

MathWorks MATLAB&SIMULINK

Nested cyclic dependencies



Sorted list:

```
0:0 A
0:1 C
    1:0 C1*
        0:0 C2
            0:0 Gain3*
            0:1 Delay3
        0:1 B2
        0:2 D2
    1:1 Delay1
0:2 D
0:3 B
```

Output execution list:

```
0:0 A(op)
0:1 C(op)
    1:0 Delay1(op)
0:2 D(op)
0:3 B(op)
```

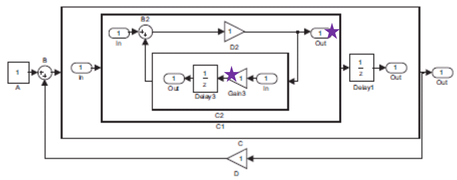
Update execution list:

```
0:0 C(up)
    0:0 C1(op)
        0:0 C2(op)
            0:0 Delay3(op)
        0:1 B2(op)
        0:2 D2(op)
    0:1 C1(up)
        0:0 C2(up)
            0:0 Gain3(op)
            0:1 Delay3(up)
    0:2 Delay1(up)
```

52

MathWorks MATLAB&SIMULINK

Nested cyclic dependencies



Sorted list:

```
0:0 A
0:1 C
    1:0 C1*
        0:0 C2
            0:0 Gain3*
            0:1 Delay3
        0:1 B2
        0:2 D2
    1:1 Delay1
0:2 D
0:3 B
```

Output execution list:

```
0:0 A(op)
0:1 C(op)
    1:0 Delay1(op)
0:2 D(op)
0:3 B(op)
```

Update execution list:

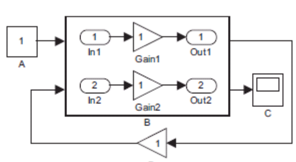
```
0:0 C(up)
    0:0 C1(op)
        0:0 C2(op)
            0:0 Delay3(op)
        0:1 B2(op)
        0:2 D2(op)
    0:1 C1(up)
        0:0 C2(up)
            0:0 Gain3(op)
            0:1 Delay3(up)
    0:2 Delay1(up)
```

53

MathWorks MATLAB&SIMULINK

Scope of applicability

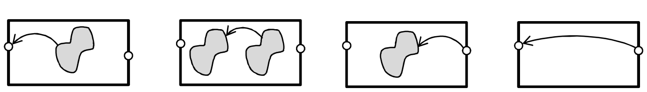
- Applies to derivative and zerocrossing as well
- But, the df path has to be breakable



Sorted list:

```
0:0 A
0:1 AlgLoop
    1:0 B
        1:0 Gain1
        1:1 Gain2
    1:1 D
0:4 C
```

- In general, four basic data dependency classes



54

MathWorks MATLAB&SIMULINK

Agenda

- Simulink preliminaries
- Executing a Simulink model
- Dealing with hierarchy
- Heterogeneous composition
- An implementation in Simulink
- Conclusions

55

MathWorks MATLAB&SIMULINK

Conclusions

- Simulink as a network of dynamic systems
  - Method set to generate behavior (output, update, …)
- Lists for the execution phases
  - df to sort according to data dependencies
  - Block methods and sample times to create lists
- Atomic subsystems for componentization
  - Heterogeneous composition to avoid dependencies
  - Heterogeneous execution lists (flat and hierarchical)
  - Scoped the class of systems where this applies

56

References

Pieter J. Mosterman and John E. Ciolfi, "**Using Interleaved Execution to Resolve Cyclic Dependencies in Time-Based Block Diagrams**," *in Proceedings of 43rd IEEE Conference on Decision and Control* (CDC'04), pp. 4057-4062, December 14 - 17, Atlantis, Paradise Island, Bahamas, 2004.

Ben Denckla and Pieter J. Mosterman, "**An Intermediate Representation and Its Application to the Analysis of Block Diagram Execution**," in Proceedings of the 2004 Summer Computer Simulation Conference (SCSC'04), pp. 167-172, July 25 - 29, San Jose, CA, 2004.

Acknowledgments

John E. Ciolfi
MathWorks

Ben Denckla
Independent Thinker

Many thanks for their continuing collaboration!

MathWorks

MATLAB&SIMULINK

MathWorks™

Accelerating the pace of engineering and science

59