# Simulink2BIP Tool

G. Tsiligiannis, V. Sfyrla, M. Bozga

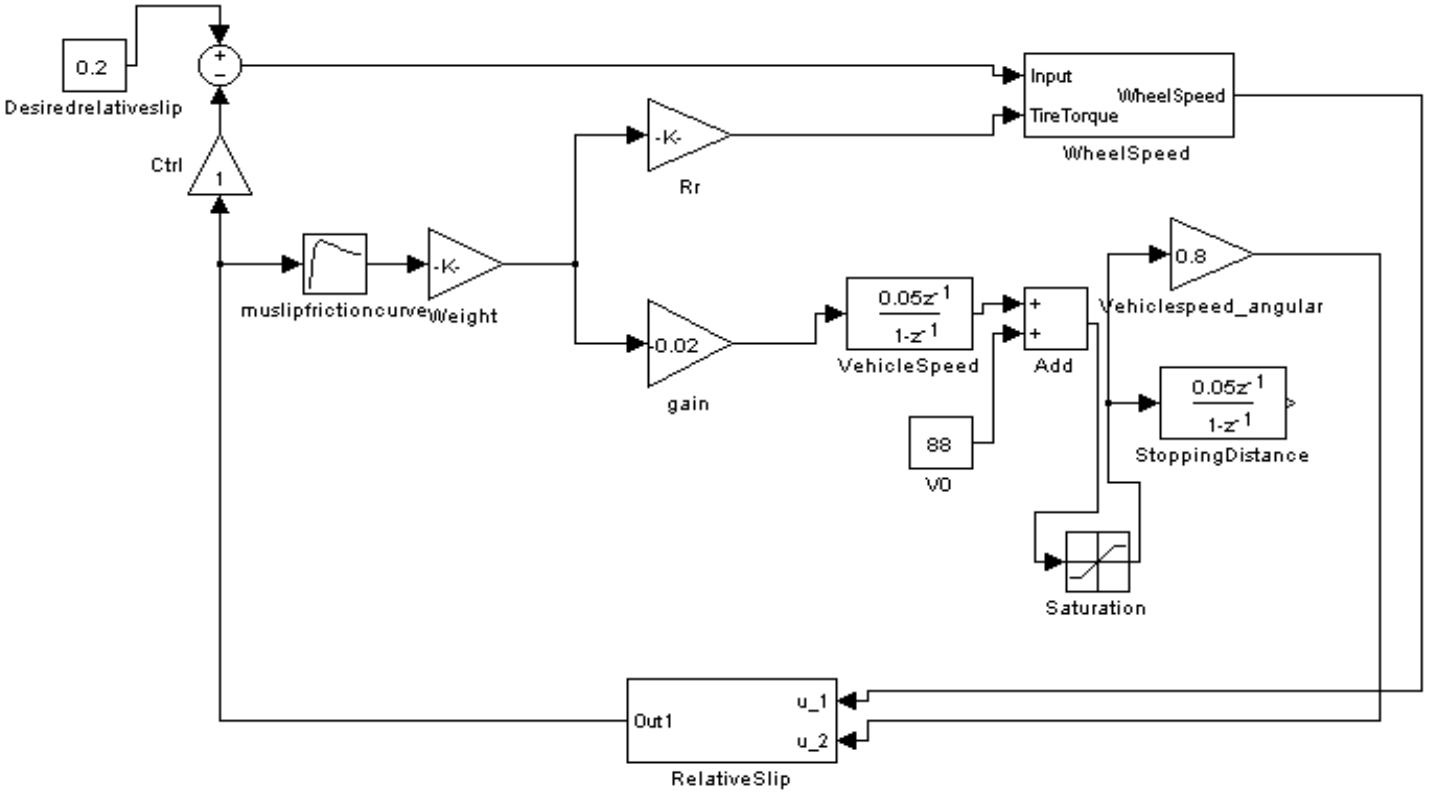
VERIMAG, 26/5/2010

# Outline

- Introduction
- Description of the Simulink2BIP tool
- Simulink library in Synchronous BIP
- An example
- Experimental Results
- Ongoing and Future work

# MATLAB/Simulink

MATLAB/Simulink is a commercial tool for modeling and simulating continuous/discrete systems



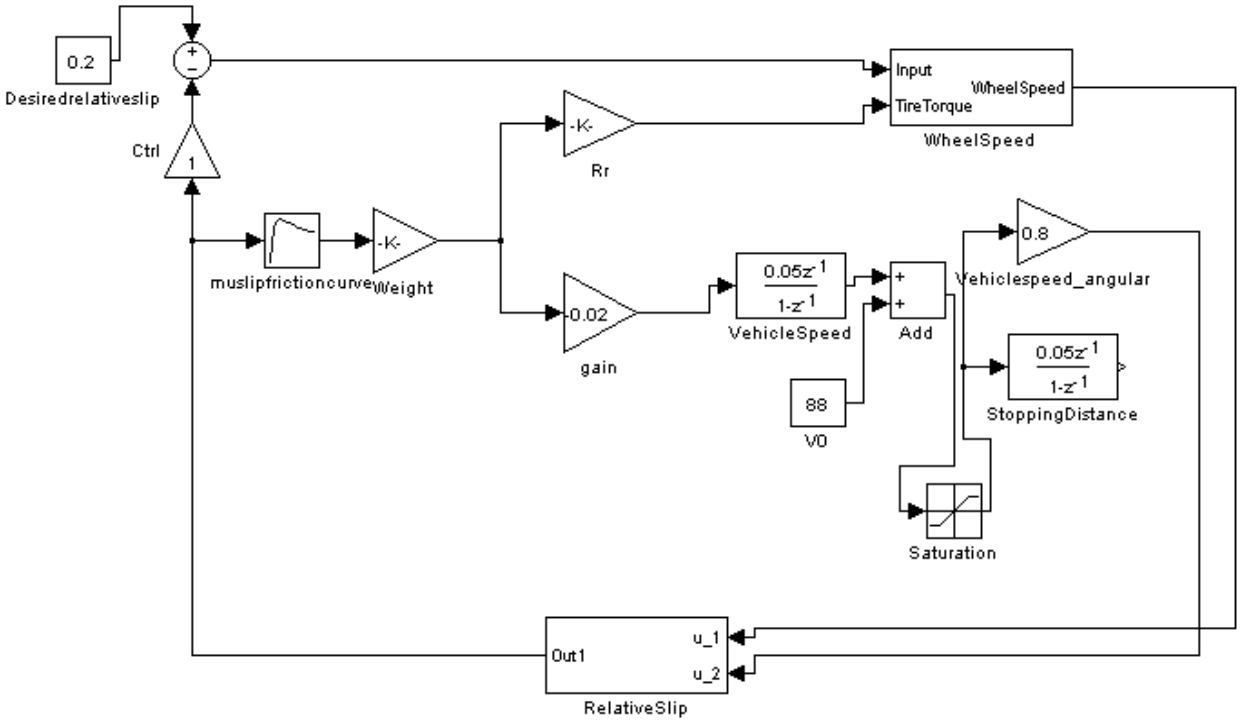

In this work we study only the discrete-time fragment of Simulink.

# General overview of MATALB/Simulink

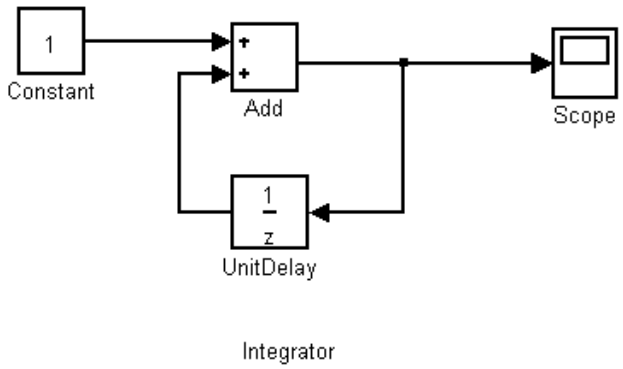
A discrete Simulink model consists of:

- Atomic blocks: e.g. combinatorial, sources, sinks, delays, …
- Subsystems, e.g. triggered, enabled, enabled periodic, ….
  - They contain atomic blocks or other subsystems
  - They are executed under conditions

In Simulink models, the execution period (sample time) of each block can be explicitly provided by the modeler or left unspecified



# The integrator example in Simulink



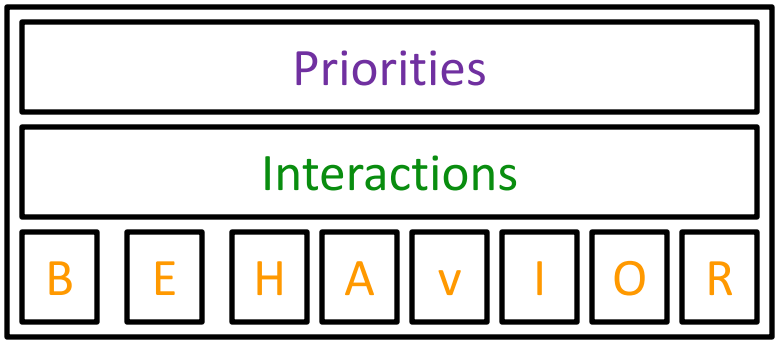

Graphical representation

```
System {
   Name                          ''integrator''
Block {
     BlockType                   Sum
     Name                        "Add"
     Ports                       [2, 1]
     SampleTime                   "8"
   }
   Block {
     BlockType                   Constant
     Name                        "Constant"
     Position                    [105, 65, 135, 95]
     OutDataType                 "sfix(16)"
     OutScaling                  "2^0"
     SampleTime                  "8"
   }

    ...

   Line {
     SrcBlock                    "Constant"
     SrcPort                     1
     DstBlock                    "Add"
     DstPort                     1
   }
   Line {
     SrcBlock                    "Add"
     SrcPort                     1
     Points                      [45, 0]
     Branch {
                 DstBlock        "Scope"
                 DstPort          1
     }
     Branch {
                 DstBlock        "UnitDelay"
                 DstPort          1
     }
   }

   ...
```

Textual representation (.mdl)

# What is BIP?

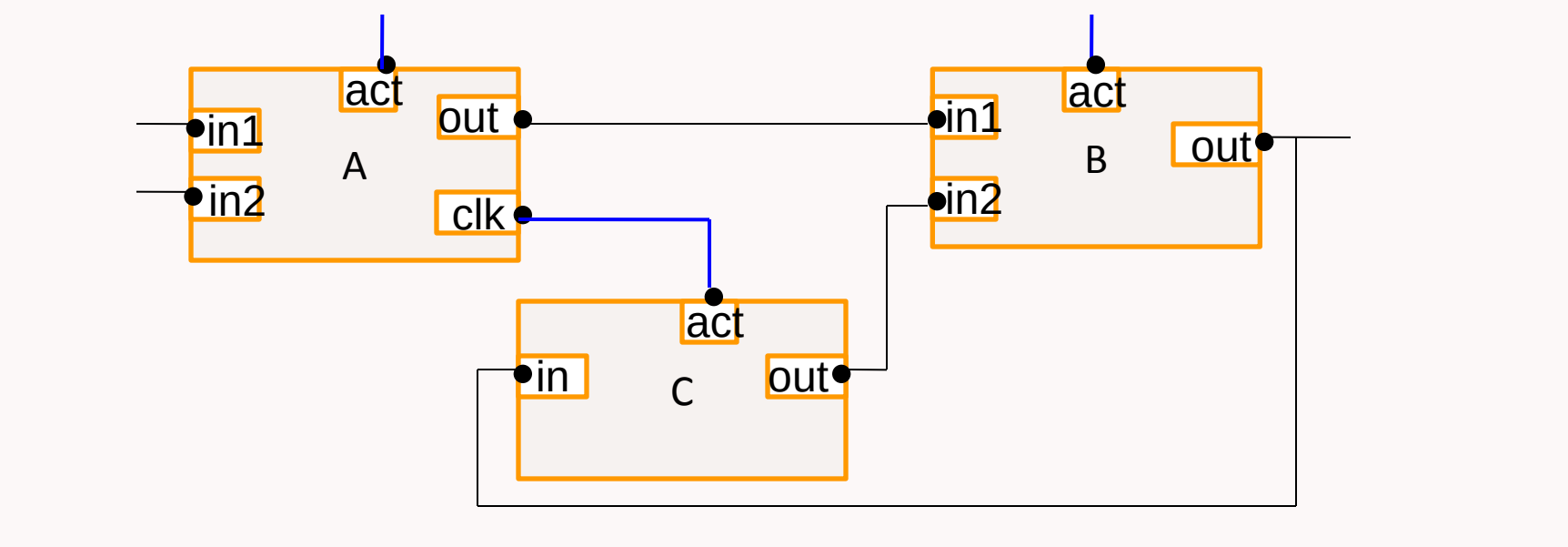
**BIP** is a *layered component based* framework




- *Behavior* - atomic components specified by automata or Petri Nets
- *Interactions* - set of ports of atomic components
- *Priorities* - selection amongst possible interactions

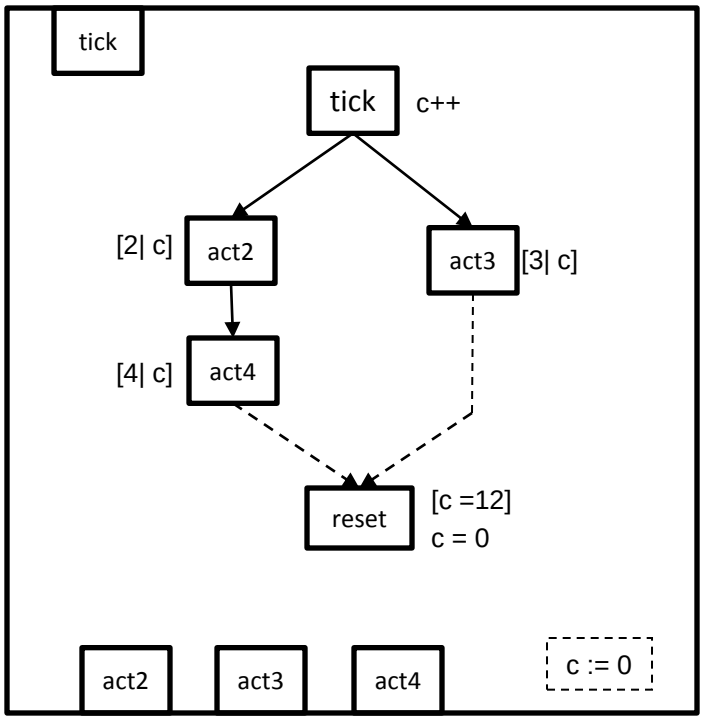
# What is Synchronous BIP

Synchronous BIP (SynBIP) is a subset of BIP for modeling synchronous systems




- The behavior of A, B and C is described by modal flow graphs
  - MFGs consist of ports and causal dependencies between ports
- In previous work we provided translation of Lustre into SynBIP
- We actually have code generator for Synchronous BIP in C (single loop code, no engine needed)

# A Multi-period Clock Generator in Synchronous BIP




The **possible executions** are:

tick
tick act2
tick act2 act4
tick act3
tick act2 act4 act3
tick act2 act4 act3 reset
tick act3 act2 act4 reset

| | | | |
|---|---|---|---|
| p → q | strong | q must follow p | pq |
| p → q | weak | q may follow p | p,pq |
| p ⇢ q | conditional | q never precedes p | p,q,pq |

# Simulink2BIP Tool

- Introduction
- Description of the Simulink2BIP tool
- Simulink library in Synchronous BIP
- An example
- Experimental Results
- Ongoing and Future work

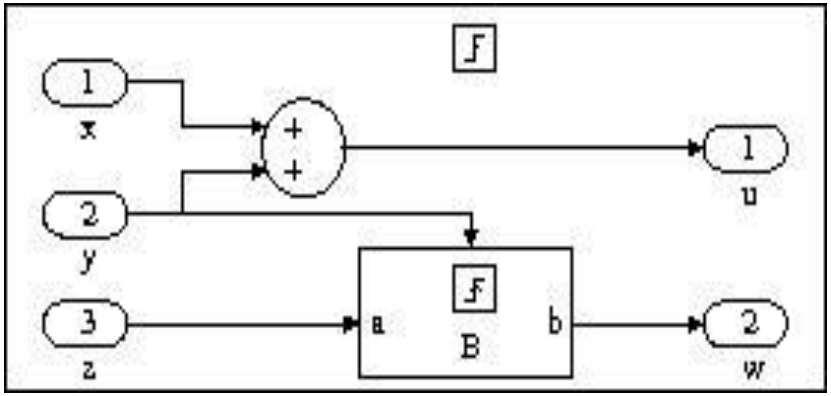
# Translation from Simulink to Synchronous BIP

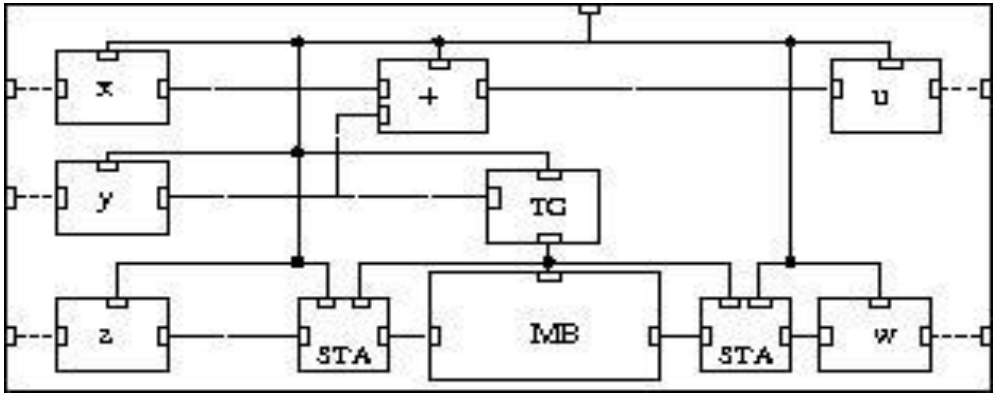
Each Simulink block is associated to a unique synchronous component.

- Basic Simulink blocks are translated to elementary components
- Simulink subsystems are translated recursively as compositions of the components associated to their contained blocks

The translation is structural:

- Dataflow and activation links used within the subsystem are translated to connectors



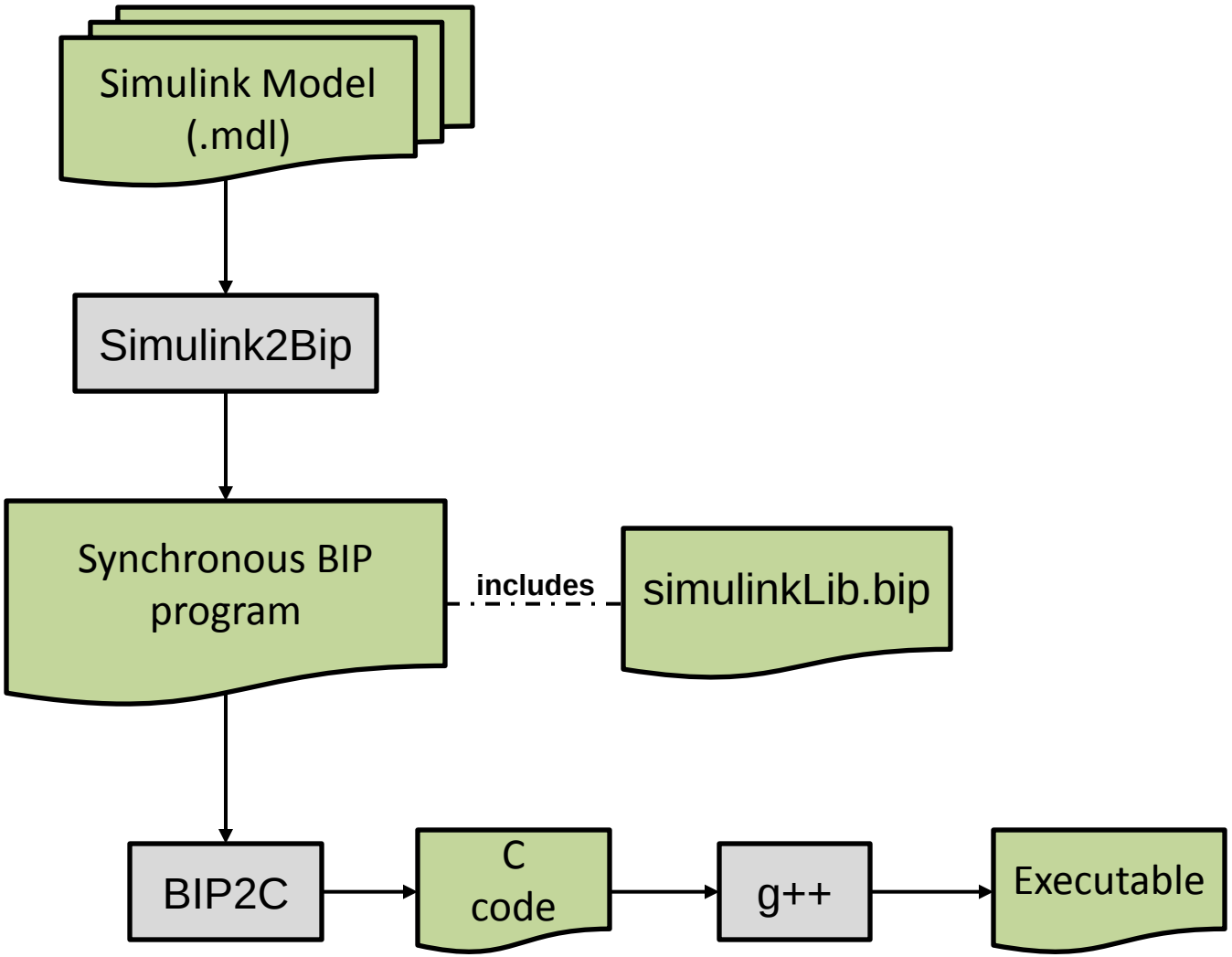




A Simulink model translated to Synchronous BIP

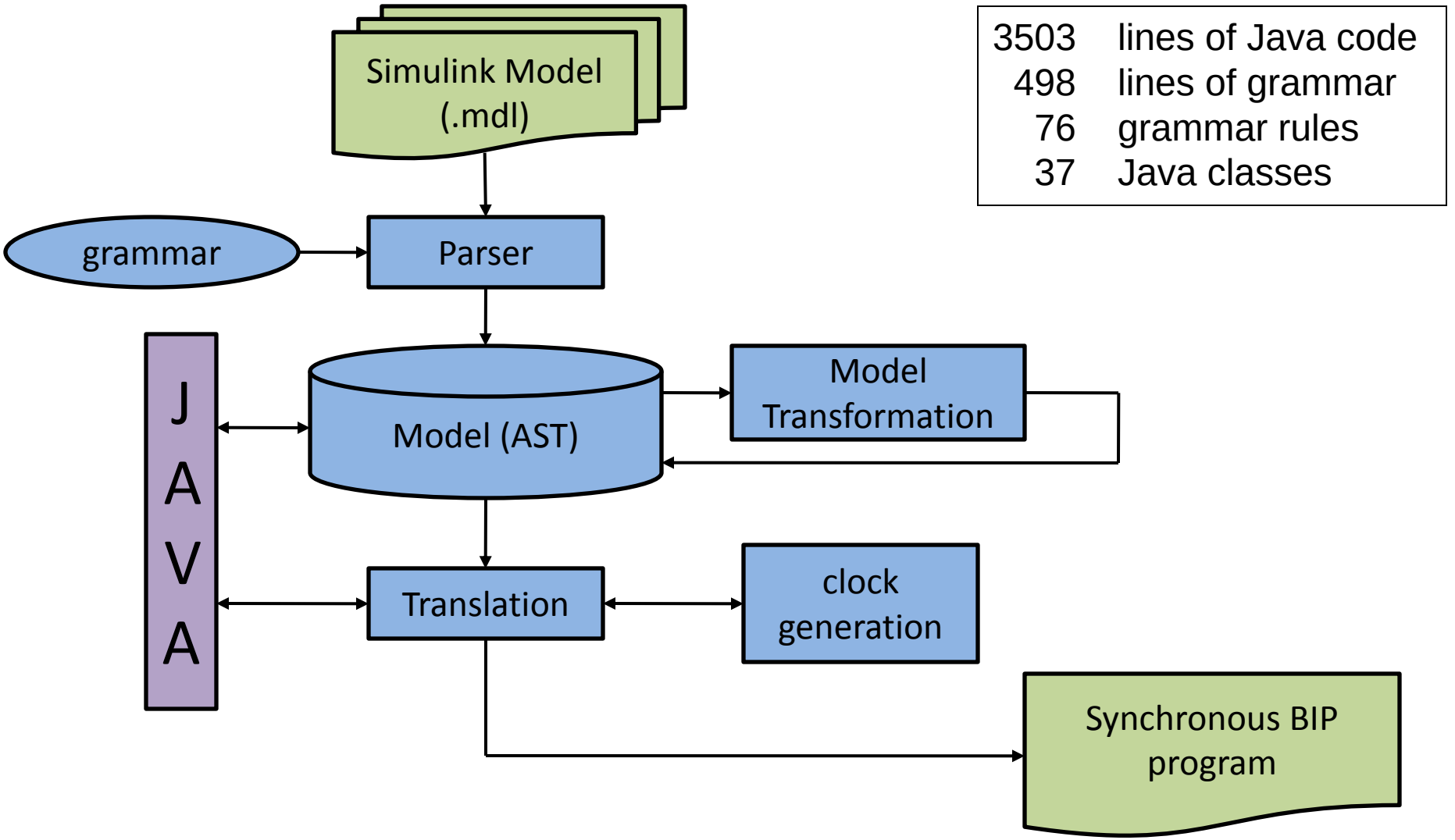
Synchronous components obtained by translation enjoy several structural properties

- well-triggered, confluent and deadlock-free, ….

# General tool architecture

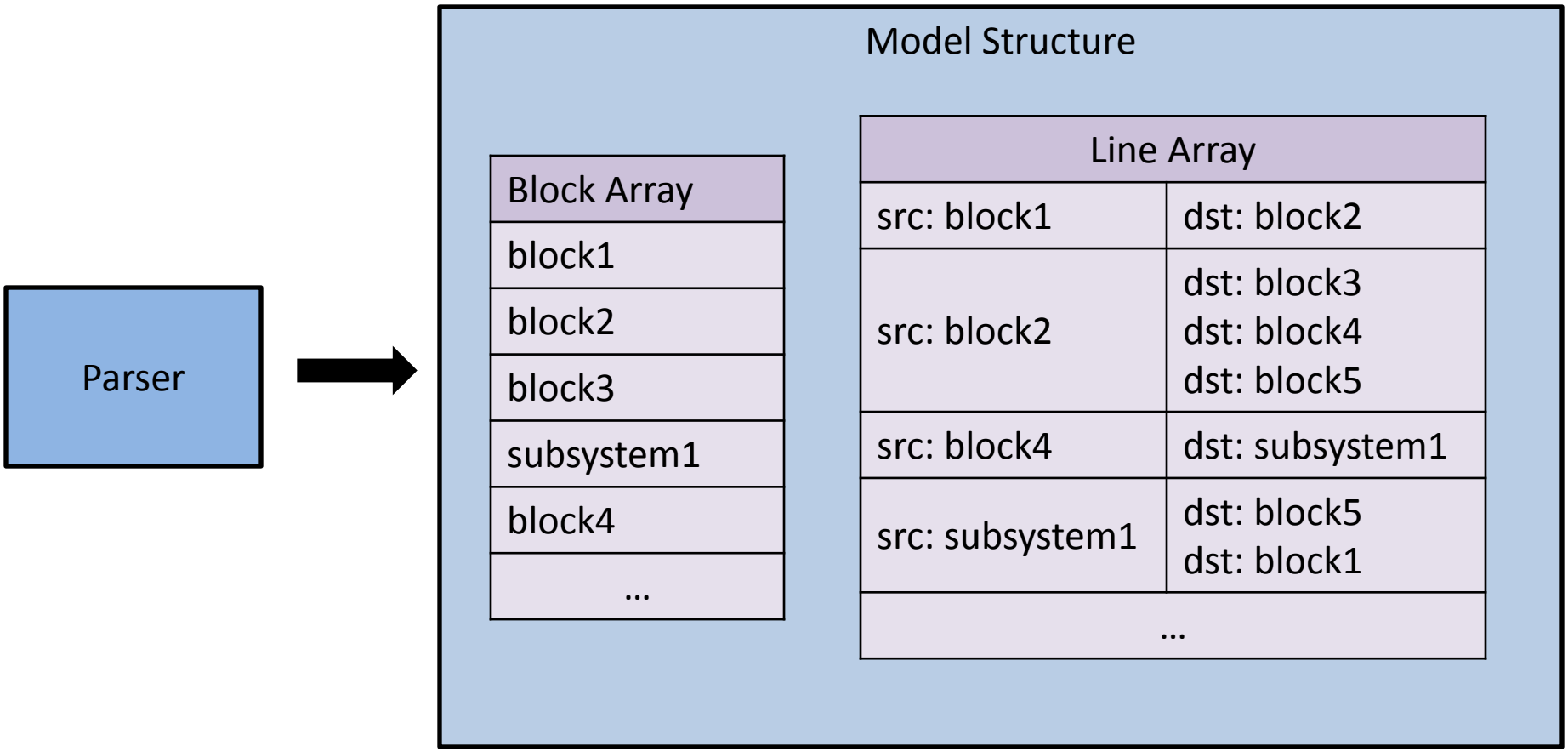


# Simulink2bip translation
# Tool schema




| | |
|---|---|
| 3503 | lines of Java code |
| 498 | lines of grammar |
| 76 | grammar rules |
| 37 | Java classes |

# Parser


Parser
Model Structure
Block Array
block1
block2
block3
subsystem1
block4
…
Line Array
src: block1
dst: block2
src: block2
dst: block3
dst: block4
dst: block5
src: block4
dst: subsystem1
src: subsystem1
dst: block5
dst: block1
…

# Model Transformations

Model transformation is an intermediate operation between the Parser and the Translator.

• It transforms some of the structures of the model produced by the parser into structures than can be handled by the translator.

• Transformations are performed according to the translation theory

These transformations concern:

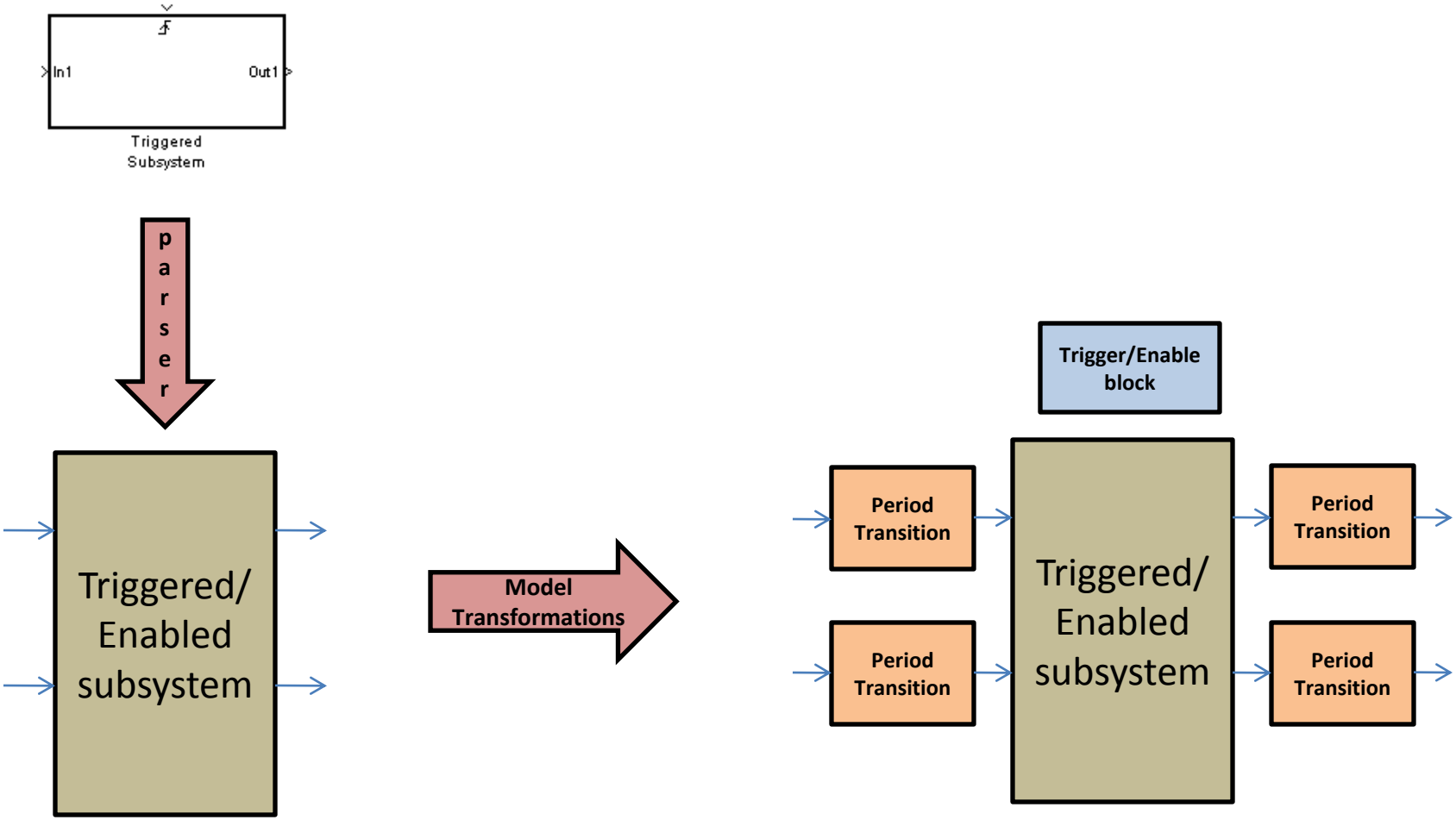
• several atomic blocks, e.g unit delay, zero order hold, etc, and the specifying of their functionality in synchronous BIP.

• subsystems, i.e. apply rules of the translation

• activation events of the blocks in order to be synchronized.

# Model Transformations

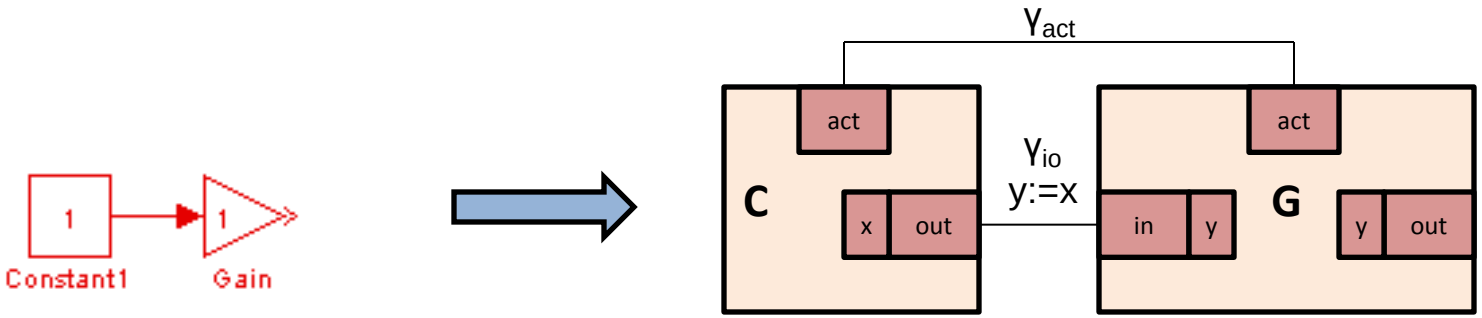
This is an example of the transformations, showing the subsystems initialization



# Translation

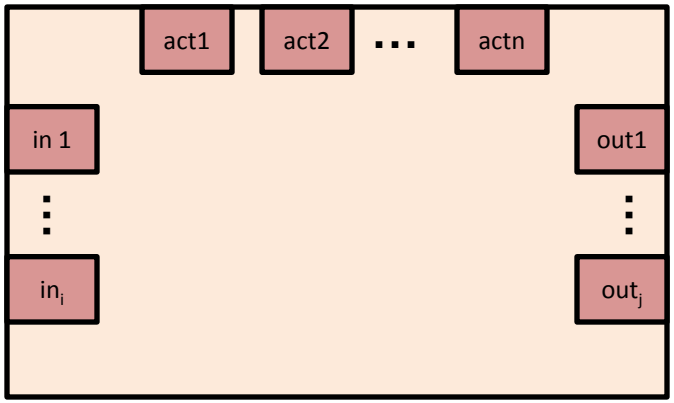
The translation of the transformed model is done as following:

- Blocks are translated into synchronous atomic components
- Lines are translated into data flow connections, e.g. $\gamma_{io}=\{C.out,G.in\}$
- Blocks are synchronized with control event connections, e.g. $\gamma_{act}=\{C.act,G.act\}$



# Translation

The top level of the Simulink model is represented by the following compound component in Synchronous BIP.




The compound component has:

- control events {$act_1$,$act_2$, …, $act_n$} and
- data events {$in_1$, …$in_i$, $out_1$, …, $out_j$ }.

The produced control events, which correspond to running time periods, need to be triggered by a clock generator

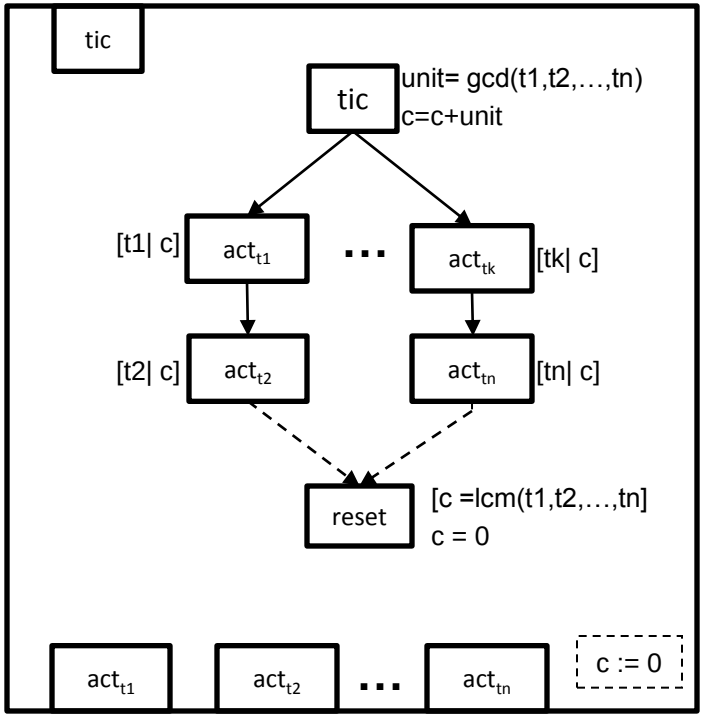
# Clock Generation

The clock generator produces control events corresponding to different sample times.

• *tic* is the global clock of the model. This port is triggered every synchronous step, and increases the value of c by one time unit.

•Each clock event $act_{ti}$ is produced when the counter value c is equal to the clock period ti.

• When the value of the global clock is equal to the least common multiple of the periods ti of all the clocks, it resets c.



# Simulink2BIP Tool

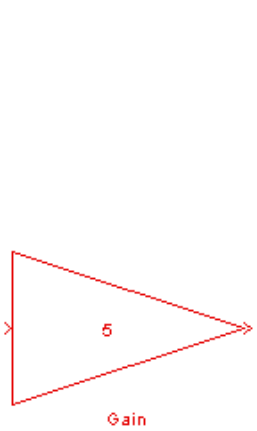
- Introduction
- Description of the Simulink2BIP tool
- Simulink library in Synchronous BIP
- An example
- Experimental Results
- Ongoing and Future work

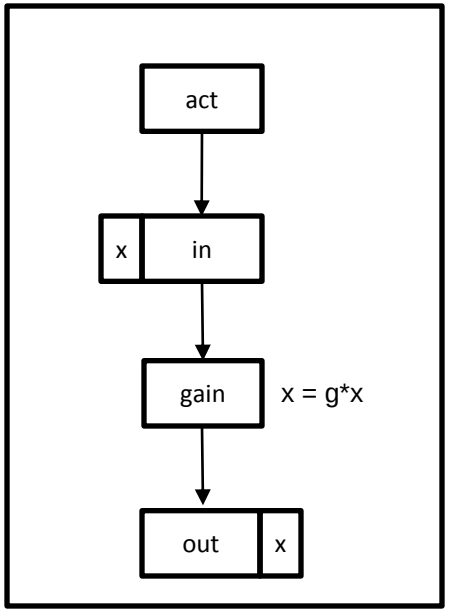
# Simulink Library in Synchronous BIP

The library contains

- 54 atomic components and
- 5 connectors
- 1739 lines of code

Each Simulink block corresponds to an atomic component



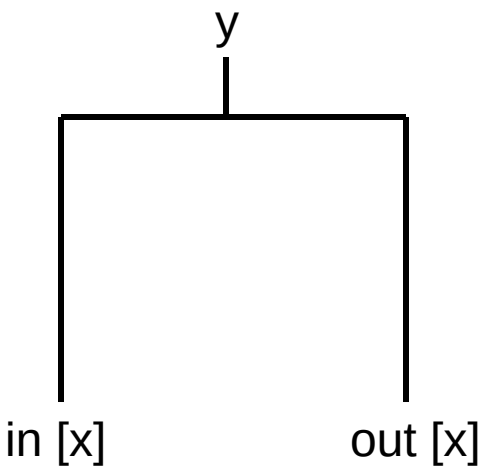

```
modal type Gain(float g)
  data float x1
  data float y
  data float g1=g

  export port ActPort act=act
  export port FloatPort in1(x1)=in1
  export port FloatPort out(y)=out

port ActPort gain

on act

on in1<=act

on gain<=in1
do { y = g1*x1; }

on out<=gain

end
```

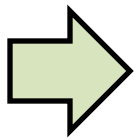
Example of the gain atomic component

# Simulink Library in Synchronous BIP

Connectors are of two types:

- data connectors, γ={ in, out }
- control connectors, γ= {acti }, i=1,..,n



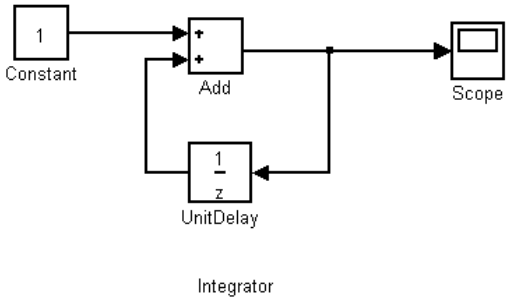
```
connector type conn(FloatPort out,FloatPort in1)
    define [ out in1 ]
    data float y
    on out in1
    up {y=out.x;}
    down {in1.x=out.x;}
    export port FloatPort cout(y)
end
```

Example of a data connector

# Simulink2BIP Tool

- Introduction
- Description of the Simulink2BIP tool
- Simulink library in Synchronous BIP
- An example
- Experimental Results
- Ongoing and Future work

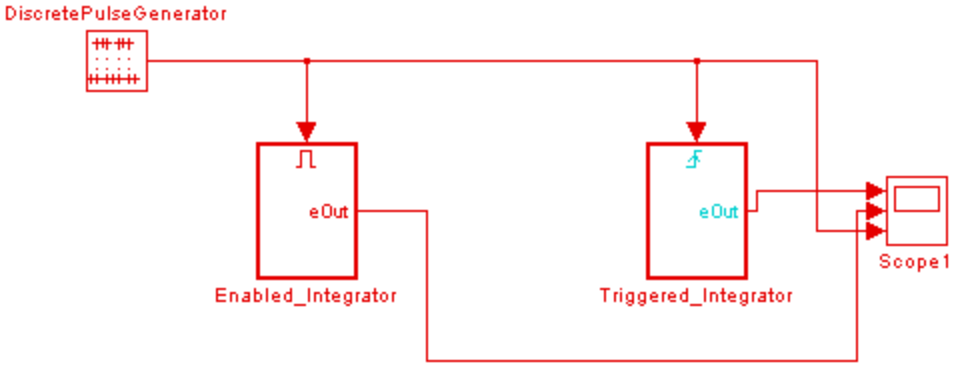
# Integrator example in Synchronous BIP

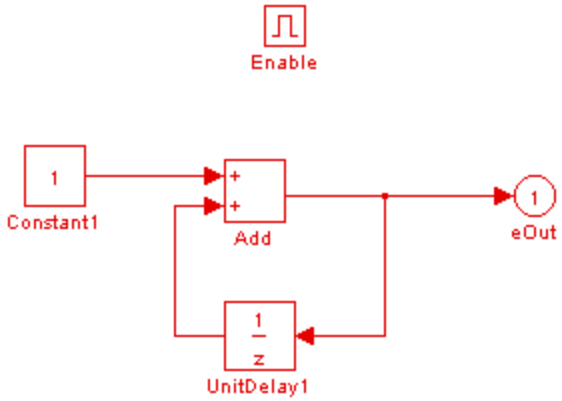



```
compound type Integrator

component simulinkLib.FloatSumSum Add
component simulinkLib.FloatConstant Constant1(1)
component simulinkLib.UnitDelaySimpleRes UnitDelay1(0.0)
component simulinkLib.OutPort eOut(6.0)

connector simulinkLib.conn Constant1_Add(Constant1.out,Add.in1)
connector simulinkLib.conn
Add_UnitDelay1(Add.out,UnitDelay1.in1)
connector simulinkLib.conn
Add_eOut(Add_UnitDelay1.cout,eOut.in1)
connector simulinkLib.conn
UnitDelay1_Add(UnitDelay1.out,Add.in2)

connector simulinkLib.act4 actConn1_1(Constant1.act, Add.act,
UnitDelay1.act, eOut.act)

export port simulinkLib.FloatPort out1 is eOut.out

export port simulinkLib.ActPort act0 is actConn1_1.act
export port simulinkLib.ActPort act1 is UnitDelay1.res

end
```

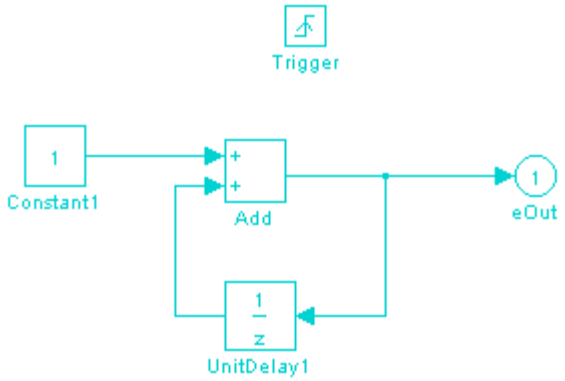
# The enabled subsystem example by Mathworks

**sample time of the model : 0.01**




Enable Integrator



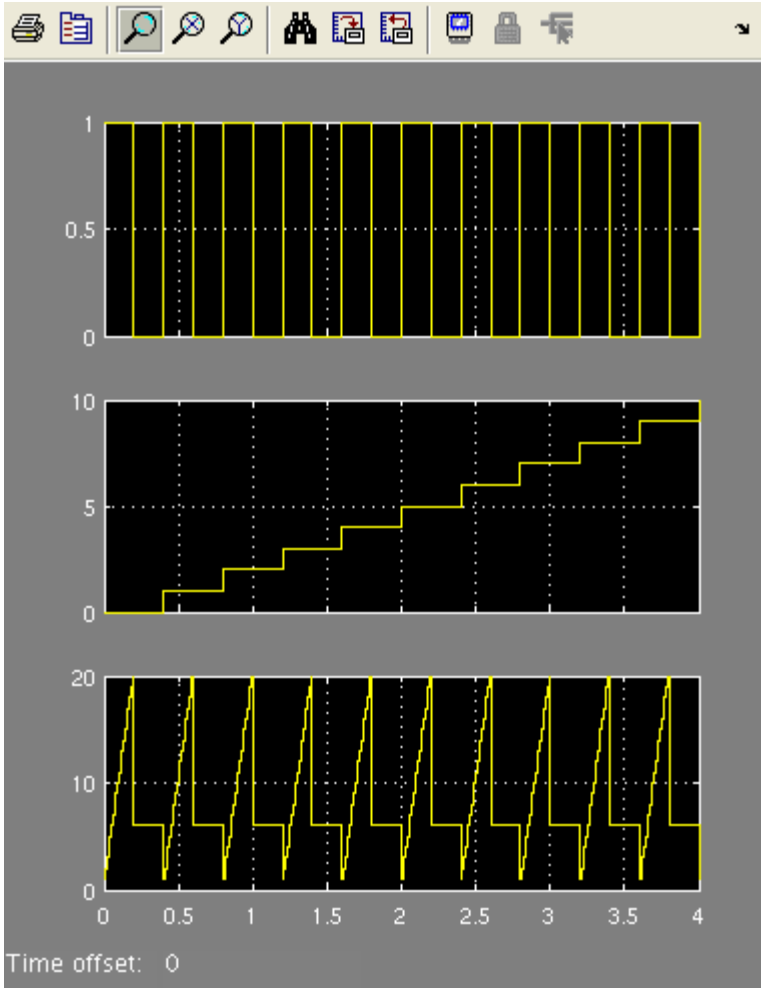

Trigger Integrator

# The enabled subsystem example by Mathworks

Pulse Generator

Triggered subsystem output

Enabled subsystem output



# The enabled subsystem example by Mathworks

In the top level system there are:

- Subsystems
  - one enabled integrator
  - one triggered integrator
- Atomic blocks
  - a pulse generator and a scope

The two subsystems contain a discrete integrator which is executed according to some condition.

# Translating the Top level system

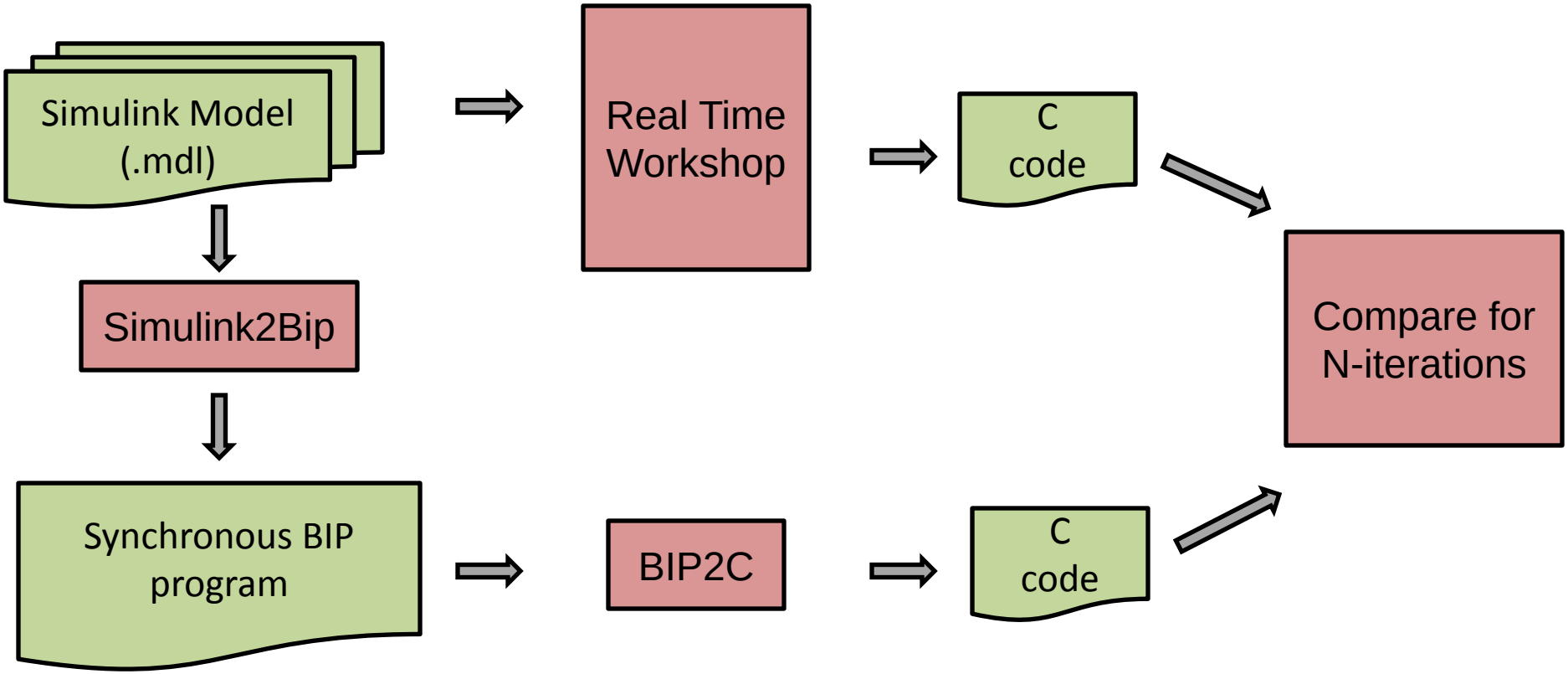
```
compound type enable_substste m

component simulinkLib.PulseGenerator DiscretePulseGenerator(1.0,20.0,40.0)

component Enabled_Integrator_sub Enabled_Integrator
component simulinkLib.FloatSink3 Scope1

component Triggered_Integrator_sub Triggered_Integrator
component simulinkLib.PeriodTransitionRes Enabled_Integrator_STA_out1_Scope1(6.0)
component simulinkLib.PeriodTransitionRes Triggered_Integrator_STA_out1_Scope1(0.0)
component simulinkLib.TriggerRise Trigger
component simulinkLib.EnablePortRes Enable

connector simulinkLib.conn DiscretePulseGenerator_Enable(DiscretePulseGenerator.out,Enable.in1)
connector simulinkLib.conn DiscretePulseGenerator_Scope1(DiscretePulseGenerator_Enable.cout,Scope1.in3)
connector simulinkLib.conn DiscretePulseGenerator_Trigger(DiscretePulseGenerator_Scope1.cout,Trigger.in1)
connector simulinkLib.conn Enabled_Integrator_STA_out1_Scope1_Scope1(Enabled_Integrator_STA_out1_Scope1.out,Scope1.in2)
connector simulinkLib.conn Enabled_Integrator_out1_Enabled_Integrator_STA_out1_Scope1(Enabled_Integrator.out1,Enabled_Integrator_STA_out1_Scope1.in1)
connector simulinkLib.conn Triggered_Integrator_STA_out1_Scope1_Scope1(Triggered_Integrator_STA_out1_Scope1.out,Scope1.in1)
connector simulinkLib.conn Triggered_Integrator_out1_Triggered_Integrator_STA_out1_Scope1(Triggered_Integrator.out1,Triggered_Integrator_STA_out1_Scope1.in1)

connector simulinkLib.act2 actConn1_1(Enable.enab0, Enabled_Integrator.act0)
connector simulinkLib.act2 actConn1_2(actConn1_1, Enabled_Integrator_STA_out1_Scope1.acti)
connector simulinkLib.act2 actConn3_1(Enable.res, Enabled_Integrator.act1)
connector simulinkLib.act2 actConn3_2(actConn3_1, Enabled_Integrator_STA_out1_Scope1.res)
connector simulinkLib.act2 actConn4_1(Trigger.trig, Triggered_Integrator.act0)
connector simulinkLib.act2 actConn4_2(actConn4_1, Triggered_Integrator_STA_out1_Scope1.acti)
connector simulinkLib.act4 actConn5_1(DiscretePulseGenerator.act, Enable.act, Scope1.act, Trigger.act)
connector simulinkLib.act2 actConn5_2(actConn5_1, Enabled_Integrator_STA_out1_Scope1.acto)
connector simulinkLib.act2 actConn5_3(actConn5_2, Triggered_Integrator_STA_out1_Scope1.acto)
connector simulinkLib.act2 actConn10_1(actConn4_2, Triggered_Integrator_STA_out1_Scope1.res)

export port simulinkLib.ActPort act0 is Enable.act0
export port simulinkLib.ActPort act1 is actConn5_3.act

end
```

# Simulink2BIP Tool

- Introduction
- Description of the Simulink2BIP tool
- Simulink library in Synchronous BIP
- An example
- Experimental Results
- Ongoing and Future work

# Comparing Performances

We compare the running time of the C code generated from Synchronous BIP (using BIP2C) and Simulink (using Real Time Workshop) for N iterations.



# Experimental Results

| FILE NAME | ITTERATIONS | SIMULINK | MFG | ATOMIC | TRIGGERED | PERIODIC | ENABLED |
|---|---|---|---|---|---|---|---|
| 16-bit counter | 1,000,000 | 1.223s | 0.565s | 97 | 16 | 0 | 0 |
| -//- | 10,000,000 | 11.744s | 5.644s | 97 | 16 | 0 | 0 |
| 64-bit counter | 1,000,000 | 5.347s | 3.115s | 365 | 60 | 0 | 0 |
| -//- | 10,000,000 | 53.652s | 31.112s | 365 | 60 | 0 | 0 |
| Big ABC | 1,000,000 | 0.359s | 0.239s | 23 | 0 | 2 | 0 |
| -//- | 10,000,000 | 3.105s | 2.024s | 23 | 0 | 2 | 0 |
| Anti-lock breaking | 1,000,000 | 0.345s | 1.394s | 39 | 0 | 2 | 0 |
| -//- | 10,000,000 | 3.200s | 13.515s | 39 | 0 | 2 | 0 |
| Steering Wheel | 1,000,000 | 0.406s | 1.676s | 120 | 1 | 15 | 0 |
| -//- | 10,000,000 | 3.417s | 16.755s | 120 | 1 | 15 | 0 |
| Multi period | 1,000,000 | 0.465s | 0.411s | 14 | 0 | 0 | 1 |
| -//- | 10,000,000 | 4.012s | 3.685s | 14 | 0 | 0 | 1 |
| Enabled Subsystem` | 1,000,000 | 0.382s | 0.380s | 24 | 0 | 0 | 2 |
| -//- | 10,000,000 | 3.201s | 3.458s | 24 | 0 | 0 | 2 |
| Thermal model house | 1,000,000 | 0.559s | 0.853s | 45 | 0 | 3 | 0 |
| -//- | 10,000,000 | 5.196s | 9.624s | 45 | 0 | 3 | 0 |

# Simulink2BIP Tool

- Introduction
- Description of the Simulink2BIP tool
- Simulink library in Synchronous BIP
- An example
- Experimental Results
- Ongoing and Future work

# Ongoing and Future work

Limitations for the tool

- naming of the blocks must follow certain rules
- sample time of the blocks must be defined in the model
- user defined functions or variables are not supported

Current work involves:

- Translation of the Clutch Model of MATLAB
- Model of a discrete integrator in Synchronous BIP containing the state port
- Complete the documentation of the tool
- Validate correctness of the tool using Dfinder.

Future work will concern:

- Automatic generation of several blocks e.g. sum block, etc.
- Finalizing of clock generator.
- Translate user defined functions from Simulink models
- Translation of Nuclear Reactor model