Summer Fellowship Report

On

Rendering Xcos Simulation Output in LaTeX

Submitted by

Makrand Rajagopal

Kanad Gaikwad

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

July 14, 2019

Contents

Acknowledgment

We are grateful to everyone who has helped us in completing this project successfully. We would like to thank **Prof.Kannan Moudgalya** and the entire FOSSEE Team at IIT Bombay, for providing us with this wonderful opportunity to work on this project. We would like to thank our mentors **Ms.Firuza Aibara** and **Mr.Abhijit Bonik** for being a constant support and pointing us out in the right direction. Their insights have been a key in helping us complete the project well within the stipulated deadline. We'd also like to thank **Mr. Nagesh Karmali** for his guidance and invaluable suggestions. Last but not the least, we truly appreciate all our fellow interns' efforts to help us out, with the problems we faced, and making our internship experience a memorable and an enjoyable one.

Chapter 1

Introduction

1.1 Objective

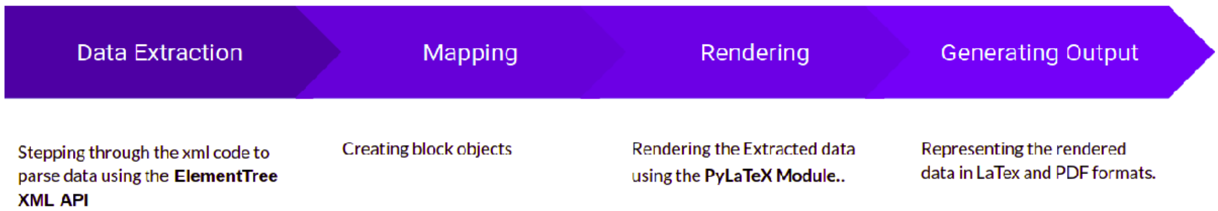
Xcos is a graphical editor to design hybrid dynamical systems models. Models can be designed, loaded, saved, compiled and simulated.

The aim of this project is to automate the process of representing these simulations in a LaTeX file.

1.2 Product Scope

This product aides scientific researchers,students and individuals looking forward to publish and document their Xcos simulation experiments in LaTeX format.

1.3 Approach



Chapter 2

Technical Specifications

2.1 Scilab

Scilab is a free and open-source, cross-platform numerical computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, numerical optimization, and modeling, simulation of explicit and implicit dynamical systems and (if the corresponding toolbox is installed) symbolic manipulations.[1]

2.2 Xcos

Xcos is a graphical editor to design hybrid dynamical systems models. Models can be designed, loaded, saved, compiled and simulated. Xcos provides functionalities for modeling of mechanical systems (automotive, aeronautics...), hydraulic circuits (dam, pipe modeling...), control systems, etc.[2]

2.3 The ElementTree XML API

ET(ELementTree) has two classes for this purpose - ElementTree represents the whole XML document as a tree, and Element represents a single node in this tree. Interactions with the whole document (reading and writing to/from files) are usually done on the ElementTree level.[3]

2.4 PyLaTeX

PyLaTeX is a Python library for creating and compiling LaTeX files. The goal of this library is to be an easy, but extensible interface between Python and LaTeX.[4]

Chapter 3

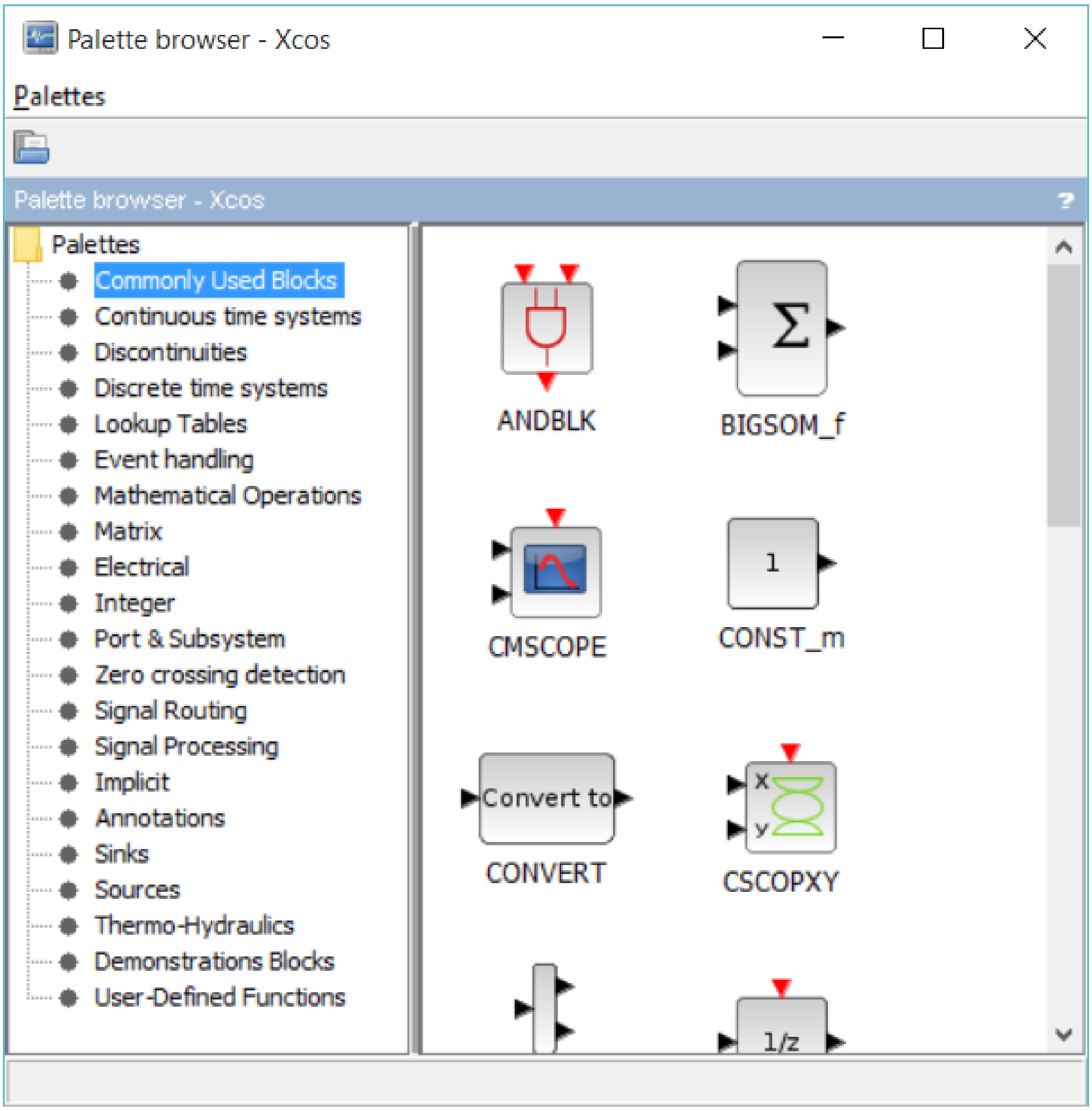
Xcos Palette

3.1 Xcos Blocks

Palette Browser

The palette browser lists all Xcos standard blocks grouped by categories

- Continous time system
- Discontinuities
- Discrete time systems
- Look up tables
- Event handling
- Maths operations
- Matrix
- Electrical
- Integer
- Port and subsytems
- Zero cross detection
- Signal routing
- Signal processing
- Implicit
- Annotions
- Sinks
- Sources
- Thermohydraulics
- Demonstration blocks
- User defined functions



Blocks

Blocks are pre-defined functions in Xcos represented with graphical interface.Every block is associated with parameters that are essential to compute the desired output.The values of the parameters can be changed according to the requirements of the user.The output varies in accordance with the parameters giving the user ability to study various trends,environments,patterns etc.[5]

3.2 Block Parameters

Following is an example of Sinewave Generator:

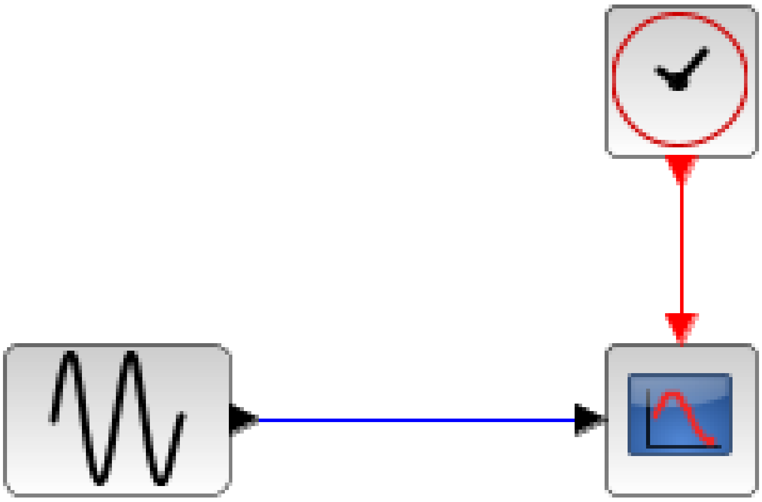


Figure 3.1: Sinewave Generator

Blocks used:

- GENSIN_f

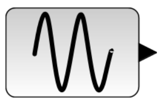


Figure 3.2: Block

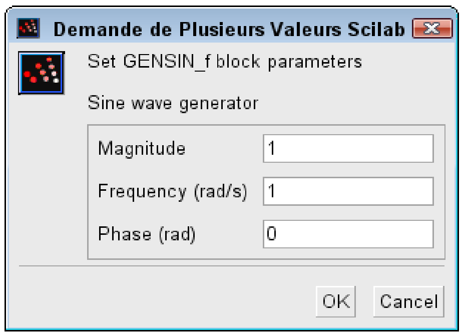


Figure 3.3: Parameters

This block is a sine wave generator and its output is defined by the equation:

Output = Msin(F.t+P)

You can adjust:

1. The magnitude M with the Magnitude parameter.
2. The frequency F in radians/second with the Frequency parameter.
3. The initial phase P in radians with the Phase parameter.

- CL0CK_c

Figure 3.4: Block

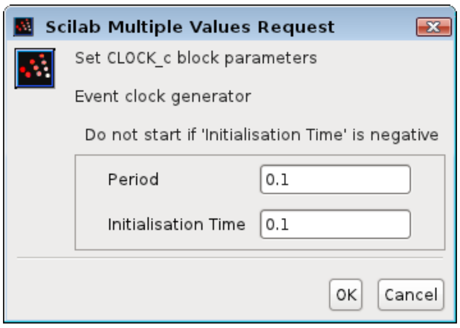


Figure 3.5: Parameters

The unique output of this block generates a regular train of events that are scheduled by parameter Period in seconds. The starting date of events generation can be set in seconds with the Initialisation Time parameter.

- CSCOPE

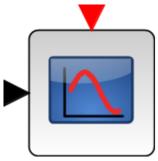


Figure 3.6: Block

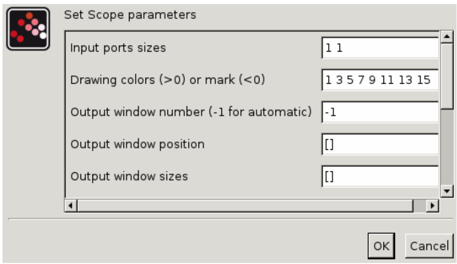


Figure 3.7: Parameters

The Scope block displays its input with respect to simulation time. Both axes have a common range. The Scope allows you to adjust the amount of time and the range of input values displayed.

Chapter 4

Extracting data from Xcos File

4.1 Xcos files Schema

Xcos files are saved with extensions:

File Description	Extension
Xcos Zip File	*.zcos
Xcos File	*.xcos
Xmi(Eclipse EMF)	*.xmi

The above formats have same schema as of a standard XML file and hence **xml.etree.ElementTree** can be used to parse data.
Following are excerpts of the Xcos file that showcase the Sinewave Generator example:

- GENSIN_f

```

 <BasicBlock id="-28483298:16b4c2b19d5:-7ff9" parent="0:2:0"
 interfaceFunctionName="GENSIN_f" blockType="c" dependsOnU="0"
 dependsOnT="1" simulationFunctionName="gensin"
 simulationFunctionType="DEFAULT" style="GENSIN_f">
<ScilabString as="exprs" height="3" width="1">
<data line="0" column="0" value="1"/>
<data line="1" column="0" value="1"/>
<data line="2" column="0" value="0"/>
```

Listing 4.1: GENSIN_f

- Clock_c

```xml

<EventOutBlock id="-28483298:16b4c2b19d5:-7fed" parent="-28483298:16b4c2b19d7:-7ff7" interfaceFunctionName="CLKOUT_f" blockType="d" dependsOnU="0" dependsOnT="0" simulationFunctionName="output" simulationFunctionType="DEFAULT" style="">
<ScilabString as="exprs" height="1" width="1">
<data line="0" column="0" value="1"/>
</ScilabString>
```

Listing 4.2: Clock_c

- Cscope

```xml

      <BasicBlock id="-28483298:16b4c2b19d5:-7ff5" parent="0:2:0" interfaceFunctionName="CSCOPE" blockType="c" dependsOnU="1" dependsOnT="0" simulationFunctionName="cscope" simulationFunctionType="C_OR_FORTRAN" style="CSCOPE">
<ScilabString as="exprs" height="10" width="1">
<data line="0" column="0" value="1 3 5 7 9 11 13 15"/>
<data line="1" column="0" value="-1"/>
<data line="2" column="0" value="[]"/>
<data line="3" column="0" value="[600;400]"/>
<data line="4" column="0" value="-15"/>
<data line="5" column="0" value="15"/>
<data line="6" column="0" value="30"/>
<data line="7" column="0" value="20"/>
<data line="8" column="0" value="0"/>
<data line="9" column="0" value=""/>
</ScilabString>
```

Listing 4.3: Cscope_f

4.2 Schema changes over Scilab versions

- Previous Versions
  - Redundant Data
  - Inefficient schema for parsing and extraction

```xml

<?xml version="1.0" encoding="UTF-8"?>
<XcosDiagram background="-1" finalIntegrationTime="0.5" title="example_9_5">
<!--Xcos - 1.0 - scilab-5.5.2 - 20160406 2040-->
<mxGraphModel as="model">
<root>
<mxCell id="-487ef5e3:131246c8237:-799a" />
<mxCell id="-487ef5e3:131246c8237:-799b" parent="-487ef5e3:131246c8237:-799a" />
<BasicBlock angle="270" dependsOnU="1" id="-487ef5e3:131246c8237:-7906" interfaceFunctionName="Switch" ordering="1" parent="-487ef5e3:131246c8237:-799b" simulationFunctionName="Switch" simulationFunctionType="DEFAULT" style="Switch;rotation=270;flip=false;mirror=false">
<ScilabString as="exprs" height="2" width="1">
<data column="0" line="0" value="0.00000001" />
<data column="0" line="1" value="100000000" />
</ScilabString>
<ScilabDouble as="realParameters" height="2" width="1">
<data column="0" line="0" realPart="0.01" />
<data column="0" line="1" realPart="100000.0" />
</ScilabDouble>
<ScilabDouble as="integerParameters" height="0" width="0" />
<Array as="objectsParameters" scilabClass="ScilabList" />
<ScilabDouble as="nbZerosCrossing" height="1" width="1">
<data column="0" line="0" realPart="0.0" />
</ScilabDouble>
<ScilabDouble as="nmode" height="1" width="1">
<data column="0" line="0" realPart="0.0" />
</ScilabDouble>
<Array as="oDState" scilabClass="ScilabList" />
<Array as="equations" scilabClass="ScilabTList">
<ScilabString height="1" width="5">
<data column="0" line="0" value="modelica" />
<data column="1" line="0" value="model" />
<data column="2" line="0" value="inputs" />
<data column="3" line="0" value="outputs" />
<data column="4" line="0" value="parameters" />
</ScilabString>
```

Listing 4.4: Scilab 5.x.x

- Scilab 6.0.1 or Scilab 6.0.2
  - Block parameters saved in ScilabString tag
  - Consistent schema

```

<?xml version="1.0" ?>
<XcosDiagram debugLevel="0" finalIntegrationTime="30.0"
   integratorAbsoluteTolerance="1.0E-6" integratorRelativeTolerance
   ="1.0E-6" toleranceOnTime="1.0E-10" maxIntegrationTimeInterval="
   100001.0" maximumStepSize="0.0" realTimeScaling="0.0" solver="
   1.0" background="-1" gridEnabled="1" title="Untitled"><!--Xcos -
    2.0 - scilab-6.0.2 - 20190214 1102-->
<Array as="context" scilabClass="String[]"></Array>
<mxGraphModel as="model">
<root>
<mxCell id="0:1:0"/>
<mxCell id="0:2:0" parent="0:1:0"/>
<BasicBlock id="-bb3dcfc:16b30a26910:-7ff9" parent="0:2:0"
   interfaceFunctionName="GENSIN_f" blockType="c" dependsOnU="0"
   dependsOnT="1" simulationFunctionName="gensin"
   simulationFunctionType="DEFAULT" style="GENSIN_f">
<ScilabString as="exprs" height="3" width="1">
<data line="0" column="0" value="5"/>
<data line="1" column="0" value="1"/>
<data line="2" column="0" value="0"/>
</ScilabString>
```

Listing 4.5: Scilab 6.x.x

If a Xcos file of any previous version is used,it will be automatically converted into the version that has been installed on the system.

4.3 The ElementTree XML API

XML is an inherently hierarchical data format, and the most natural way to represent it is with a tree.The **xml.etree.ElementTree** module implements a simple and efficient API for parsing and creating XML data.Element Tree has two classes for this purpose –

- **ElementTree** represents the whole XML document as a tree
- **Element** represents a single node in this tree.

Interactions with the whole document (reading and writing to/from files) are usually done on the ElementTree level. Interactions with a single XML element and its sub-elements are done on the Element level.

XPath is a language for addressing parts of an XML document.

XPath syntax:

'./children/grandchildren'

'.' denotes the 'root' or 'parent'

Chapter 5

Block Generator

5.1 Need for creating block objects

The values of different parameters of the individual blocks aren't mapped to their keys in the XML File.This makes it difficult to extract these values and represent them in a documented format.

```xml

<BasicBlock id="-28483298:16b4c2b19d5:-7ff9" parent="0:2:0"
    interfaceFunctionName="GENSIN_f" blockType="c" dependsOnU="0"
    dependsOnT="1" simulationFunctionName="gensin"
    simulationFunctionType="DEFAULT" style="GENSIN_f">
<ScilabString as="exprs" height="3" width="1">
<data line="0" column="0" value="1"/>
<data line="1" column="0" value="1"/>
<data line="2" column="0" value="0"/>
```

Listing 5.1: GENSIN_f

For instance,the user manipulatable parameters of GENSIN_f BasicBlock namely Magnitude,Frequency and Phase are stored in the value attribute within the data tag in ScilabString.However, these parameter names aren't explicitly mentioned in the XML code.

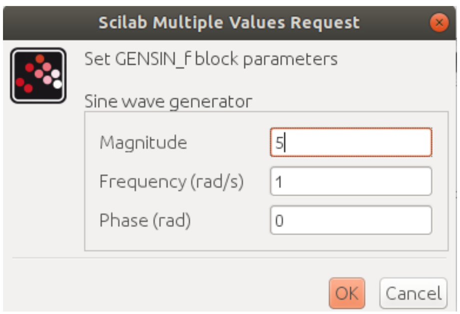


Figure 5.1: GENSIN_f BasicBlock User manipulatable parameters' dialog box

To fix this issue, we are using the Object Oriented Programming Methodology to create unique block objects of individual blocks as we encounter them while stepping through the XML file.

5.2 blocks.py and generators.py

We've deployed the Object Oriented Programming paradigm in our software by creating a */core* folder with two files namely *blocks.py* and *generators.py*.All the Xcos palette block classes are implemented in *blocks.py*.Any new Xcos blocks that may get introduced in newer versions of Scilab can be added in the form of classes in *blocks.py*. Therefore, this approach provides a standardized way of extracting data from XML files and makes the software stable and scalable for newer versions.

We've used Factory method, a creational design pattern,related to object creation. In this pattern, we are creating block objects without exposing the creation logic to client and the client uses the same common interface to create different types of BasicBlock objects.

```
from abc import ABC, abstractmethod

class Block(ABC):
"""
Abstract class for blocks
"""

@abstractmethod
def parameters(self):
pass


class GenSin(Block):
"""The GenSin Concrete Class which implements the Block interface
    """

def __init__(self,data):
self._funcname = "GENSIN_f"
self._magnitude = data[0]
self._frequency = data[1]
self._phase = data[2]

def parameters(self):
return {"Function Name:":self._funcname,"magnitude:": self.
    _magnitude, "frequency:": self._frequency, "phase:": self._phase
    }
```

Listing 5.2: *blocks.py*: Block Abstract Class and GENSIN_f BasicBlock Class

```python
from abc import ABC, abstractmethod

class Block(ABC):
"""
Abstract class for blocks
"""

@abstractmethod
def parameters(self):
pass


class GenSin(Block):
"""The GenSin Concrete Class which implements the Block interface
    """

def __init__(self,data):
self._funcname = "GENSIN_f"
self._magnitude = data[0]
self._frequency = data[1]
self._phase = data[2]

def parameters(self):
return {"Function Name:":self._funcname,"magnitude:": self.
    _magnitude, "frequency:": self._frequency, "phase:": self._phase
    }
```

Listing 5.3: *generators.py*: BlockGenerator class and Gensin_f BasicBlock Generator

5.3 List of Blocks added to the core folder

1. CLSS_f
2. GAINBLOCK_f
3. Inductor
4. SineVoltage
5. CLR_f
6. Gen_SQR
7. BOUNCE
8. Gen_Sin,
9. CScope
10. Clock
11. SampleClock
12. Rand_m
13. Matcatv
14. Matcath
15. Const_m
16. Gainblk
17. BOUNCEXY
18. IFTHEL_f
19. VanneReglante
20. Summation
21. SourceP
22. Bache
23. PerteDP
24. MatEig
25. Rootcoef
26. Extract
27. Integral_m
28. Sci_func
29. Matinv
30. SATURATION
31. Mux
32. AUTOMAT
33. TCLSS
34. DLSS
35. DLR
36. DOLLAR
37. DOLLAR_m
38. DOLLAR_f
39. Dlradapt_f
40. Samphold_m
41. Register
42. Switch
43. Resistor
44. Step_function
45. Ground
46. BIGSOM_f
47. generic_block
48. C_block
49. Fortran_block
50. Debug
51. Expression
52. CBLOCK4
53. CBLOCK
54. MBLOCK
55. BPLATFORM
56. CLKINV_f
57. TIME_f
58. RAMP
59. Const_f
60. CONST
61. PULSE_SC
62. Sigbuilder
63. Modulo_count
64. Counter
65. TkScale
66. FROMWSB
67. READAU_f
68. READC_f
69. RFILE_f
70. INIMPL_f
71. Backlash
72. Deadband
73. DELAYV_f
74. SELECT_f
75. MFCLICK_f
76. HALT_f
77. IN_f
78. DIFF_f
79. GENERAL_f
80. ZCROSS_f
81. End_c
82. Endblk
83. HYSTERESIS
84. RATELIMITER
85. SATURATION
86. M_freq
87. CLK GOTO
88. CLKOUTV_f
89. Extract_Activation
90. ANDBLK
91. Virtualclk0
92. EVTGEN_f
93. EVTVARDLY
94. ANDLOG_f
95. INTRPBLK
96. INTRP2BLK
97. CLKOUTV_f
98. Edge_trigger
99. CLKFROM
100. MCLOCK_f
101. AFFICH_m
102. BARXY
103. CMSCOPE
104. CSCOPXY3D
105. WRITEAU_f
106. CANIMXY
107. CMAT3D
108. CANIMXY3D
109. CSCOPXY
110. CMATVIEW

111. TRAS_f
112. TOWS_c
113. OUTIMPL_f
114. OUT_f
115. Evtdly_c
116. DEMUX
117. DEMUX_f
118. Extractor
119. ISELECT_m
120. MUX_f
121. M_SWITCH
122. FROM
123. FROMMO
124. NRMSOM_f
125. RELAY_f
126. SELECT_m
127. Switch_f
128. MATMAGPHI
129. TrigFun
130. Expblk_m
131. Powblk_m,
132. Relationalop,
133. SQRT,
134. MAXMIN,
135. SIGNUM
136. ABS_VALUE
137. INVBLK
138. TANBLK_f
139. LOGBLK_f
140. MATZREIM
141. CCS
142. CVS
143. Capacitor
144. OpAmp
145. VsourceAC
146. VVsourceAC
147. PMOS
148. NMOS
149. SHIFT
150. SUBMAT
151. CONVERT
152. RICC
153. INTMUL
154. MATEXPM
155. MATLU
156. MATDET
157. MATDIAG
158. MATZCONJ
159. MATBKSL
160. EXTTRI
161. BITSET
162. BITCLEAR
163. MATPINV
164. MATSING
165. MATRESH
166. DLATCH
167. NPN
168. PNP
169. DERIV
170. Voltagesensor
171. Currentsensor
172. CURV_F
173. SUM_f
174. CLOCK_f
175. NEGTOPOS_f
176. POSTONEG_f
177. CONSTRAINT2_c
178. GotoTagVisibility
179. GotoTagVisibilityMO
180. VariableResistor
181. SCALAR2VECTOR
182. CLKGotoTagVisiblity
183. CEVENTSCOPE
184. IdealTransformer
185. PotentialSensor
186. ConstantVoltage
187. EXTRACTBITS

5.4 Special Cases

Some of the Blocks have inconsistent schema and data is stored in different sub-blocks within the main Basic Block.We have incorporated these blocks,however user may find some discrepancy in the outputs.

1. Delay_f
2. SRFLIPFLOP
3. JKFLIPFLOP
4. LOGICAL_op
5. TextBlock
6. SELF_SWITCH

Chapter 6

Data Rendering

6.1 Processing the Input

The rendering of the *xcos* file and generation of the *pdf* and *tex* file is processed in the *input_processor.py* file.

Executing input_processor.py

In order to execute the python file,following arguments should be passed:

- **xcos_path:** the path of the xcos file
- **image_path:**the path of the image file

Genrated_folder

A folder with name same as that of a Xcos file is generated.The generated files such as tex file and pdf file are automatically stored in the folder.Simulation image with xcos file name is also stored in the folder.This gives ease of access to user to share the folder by creating a zip file.

Genrated_files

The path of the xcos file has to be provided by the user as the second argument in the terminal.The xcos file is parsed and rendered which produces two files:

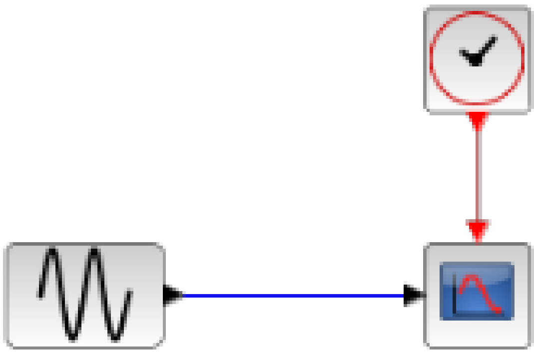
- **.tex** represents the data in LATEX format
- **.pdf** represents the data as a single PDF file.The parameters are presented in a tabular format

Sinewave Generation Example

```
\subsection{Cscope}%
\label{subsec:Cscope}%
\begin{tabular}{|c|c|}%
\hline%          z
Name&Value\\%
\hline%
Color Vector&1 3 5 7 9 11 13 15\\%
\hline%
Output window number&{-}1\\%
\hline%
Output window position&{[}{]}\\%
\hline%
Output window sizes&{[}600;400{]}\\%
\hline%
Ymin&{-}15\\%
\hline%
Ymax&15\\%
\hline%
Refresh period&30\\%
\hline%
Buffer Size&20\\%
\hline%
Accept Herited Events&0\\%
\hline%
Name of Scope&\\%
\hline%
```

Listing 6.1: Excerpt of Generated Tex file

1 Xcos



Function Name:	GENSIN_f
magnitude:	5
frequency:	1
phase:	0

Function Name:	CSCOPE
Color or mark vector:	1 3 5 7 9 11 13 15
Output window number:	-1
Output window position:	[]
Output window sizes:	[600;400]
Ymin:	-15
Ymax:	15
Refresh period:	30
Buffer Size:	20
Accept herited events:	0
Name of Scope:	

Function Name:	CLOCK_c
Period:	0.1
Initialisation Time:	0.1

Function Name:	CLKOUTV_f
Port number:	1

Figure 6.1: Generated pdf

6.2 Rendering Xcos on cloud examples

The current version of the software has been tested for a few Xcos on cloud examples and we've been able to successfully extract data out of the connected models and represent it LaTeX and PDF formats. The Google Drive Link [6] contains the generated media,also we would add further examples in near future.
https://drive.google.com/drive/folders/1IRrsmScBFUvrMADnSXciZHbIQzqow0NE

Bibliography

[1] Xcos blocks. https://www.scilab.org.

[2] Xcos. https://www.scilab.org/software/xcos.

[3] The elementtree xml api. https://docs.python.org/3/library/xml.etree.elementtree.html#module-xml.etree.ElementTree.

[4] Pylatex. https://jeltef.github.io/PyLaTeX/current/index.html.

[5] Xcos blocks. https://help.scilab.org/docs/6.0.2/en_US/section_4834819644bddf2dedeef2520b2ca171.html.

[6] Xcos on cloud example. https://drive.google.com/drive/folders/1IRrsmScBFUvrMADnSXciZHbIQzqow0NE.