HEPSYCODE Guide

Written by	***Responsibility***
Vittoriano Muttillo	RTD-A
Verified by	
Luigi Pomante	Project Manager
Approved by	
Luigi Pomante	Project Manager

CHANGE RECORDS

ISSUE	DATE	§ CHANGE RECORDS	AUTHOR
1.0	12/08/2019	First issue.	V. Muttillo
2.0	02/04/2024	Second issue.	V. Muttillo

TABLE OF CONTENTS

1. INTRODUCTION

1.1. SCOPE AND PURPOSE

The purpose of this document is to be able to use HEPSYCODE.

1.2. REFERENCE DOCUMENTS

Acronym	Reference	Title
RD01	http://www.pomante.net/sito_gg/attd.htm	HW/SW Co-Design PhD Courses
RD02		PhD Thesis Vittoriano Muttillo

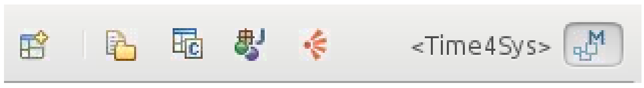
1.3. DEFINITIONS AND ACRONYMS

GUI	Graphical Users Interface
HML	HEPSYCODE Modeling Language
PAM 1-2	Partitioning Architecture Definition and Mapping phase 1-2
CC4CS	Clock Cycles for C Statement
ISA	Instruction Set Architecture

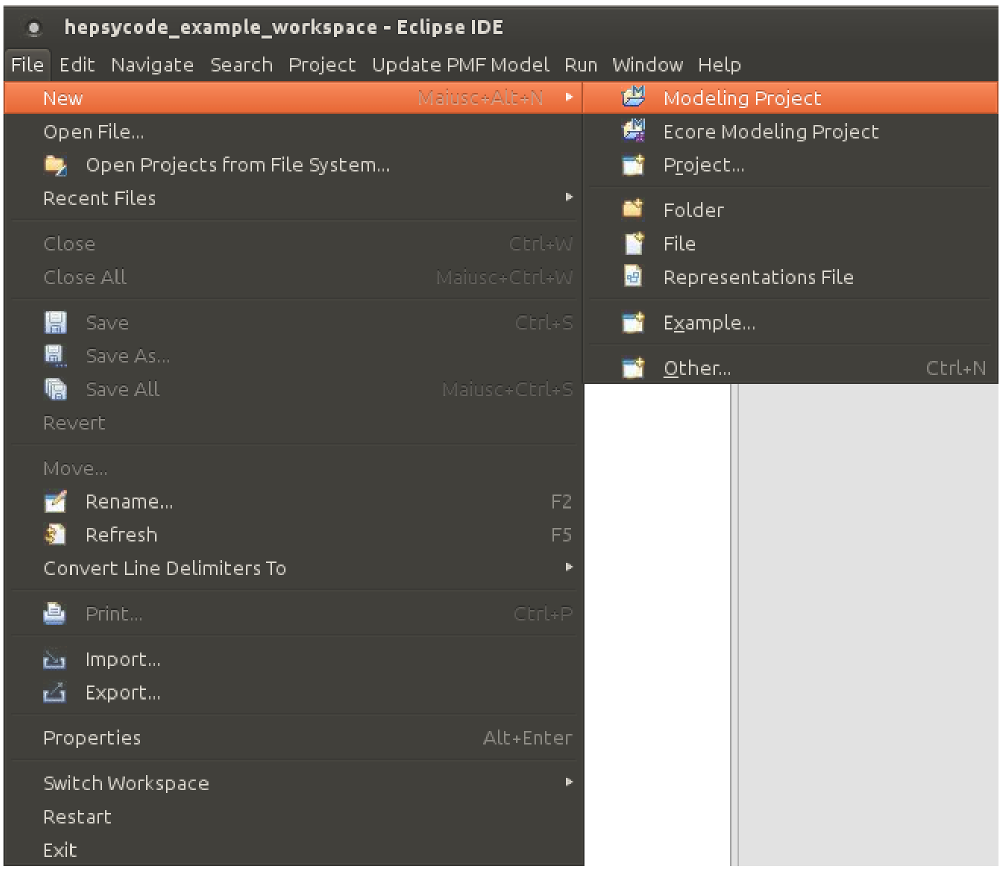
2. TUTORIAL

2.1. PROJECT CREATION

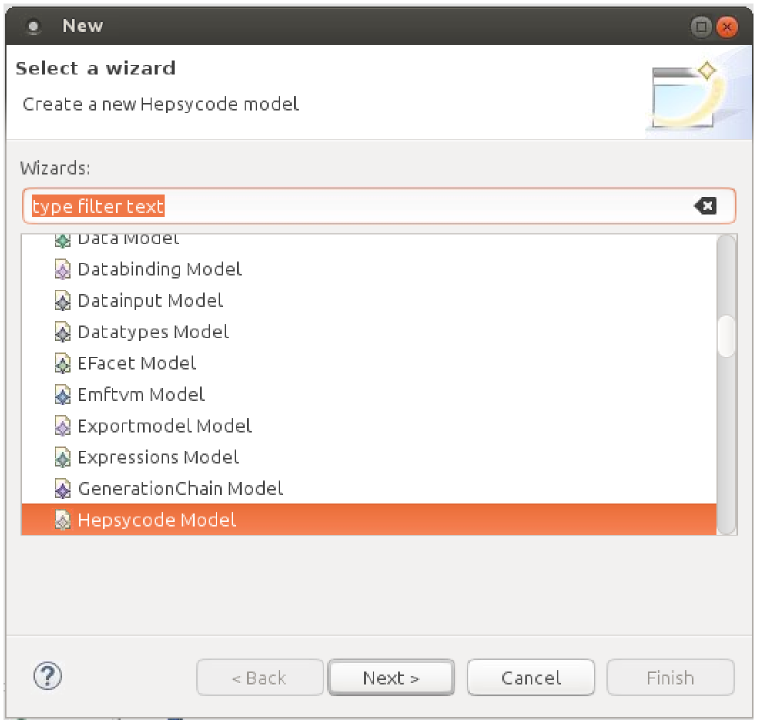
- Open Eclipse and select the workspace you want
- Check the "Modeling" view at the top right of the IDE.



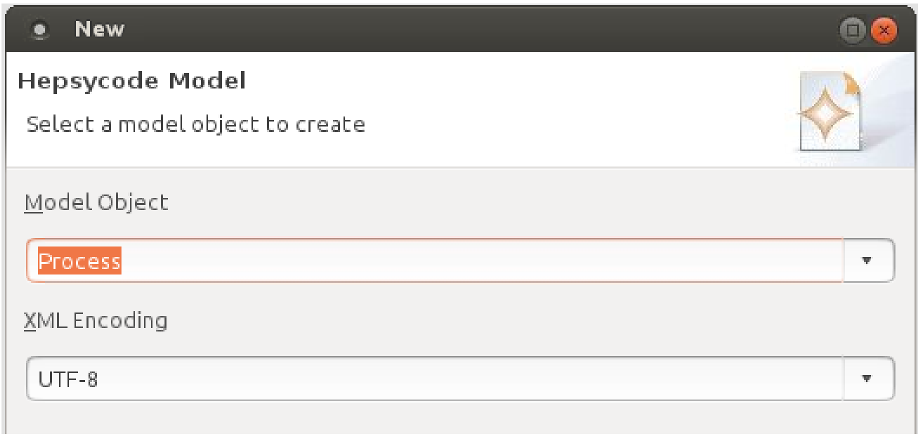
- Click at the top left on *File > New > Modeling Project*



- Choose the name of your new **Modeling Folder**
- Right Click on the *Modeling Project Folder* and choose *New > Other > Hepsycode Model*



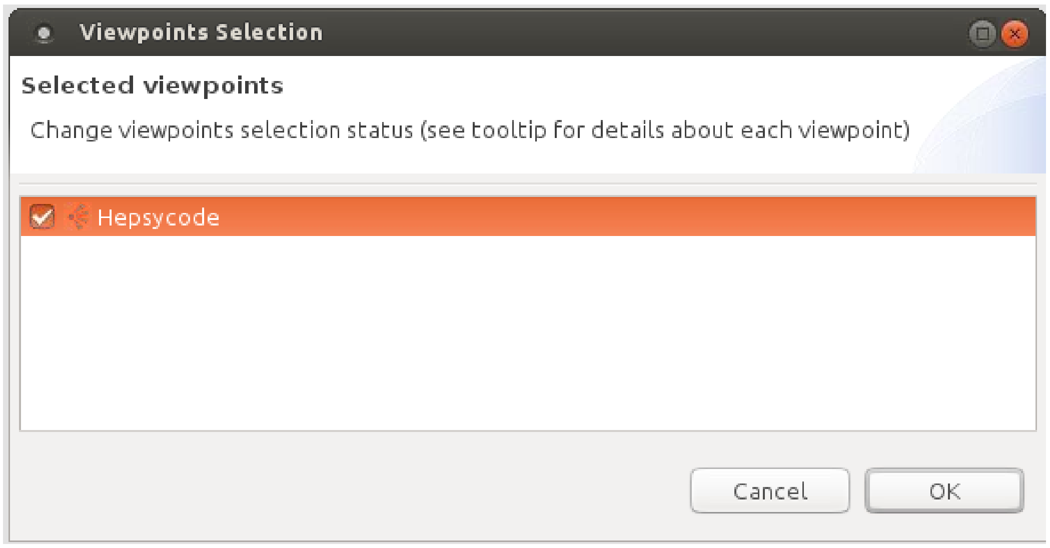
- Then in the "*Hepsycode Model"* windows choose "**Process**"



Model Object: Process

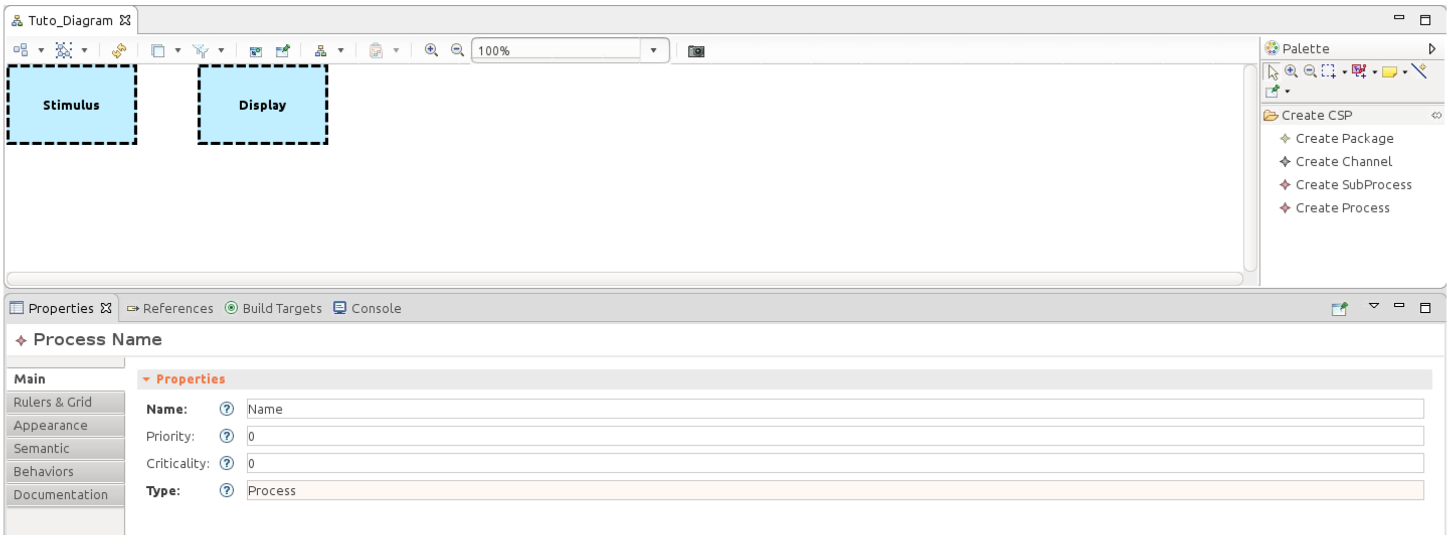
XML Encoding: UTF-8

- Again, right Click on the *Modeling Project Folder* and select *Viewpoints Selection*. Then check "Hepsycode" and click OK


Viewpoints Selection
Selected viewpoints
Change viewpoints selection status (see tooltip for details about each viewpoint)
Hepsycode
Cancel
OK

- One last time, right click on the Modeling Project Folder and select *Create representation > hepsycode diagram*. Click on Next and then Finish. (Don't change the name of the process)

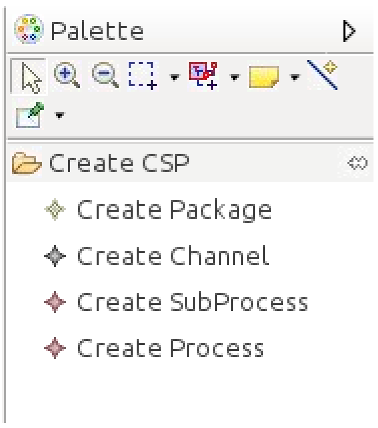
- After choosing the name of the hepsycode diagram, the Graphical Users Interface (GUI) should open automatically



Name:	Name
Priority:	0
Criticality:	0
Type:	Process

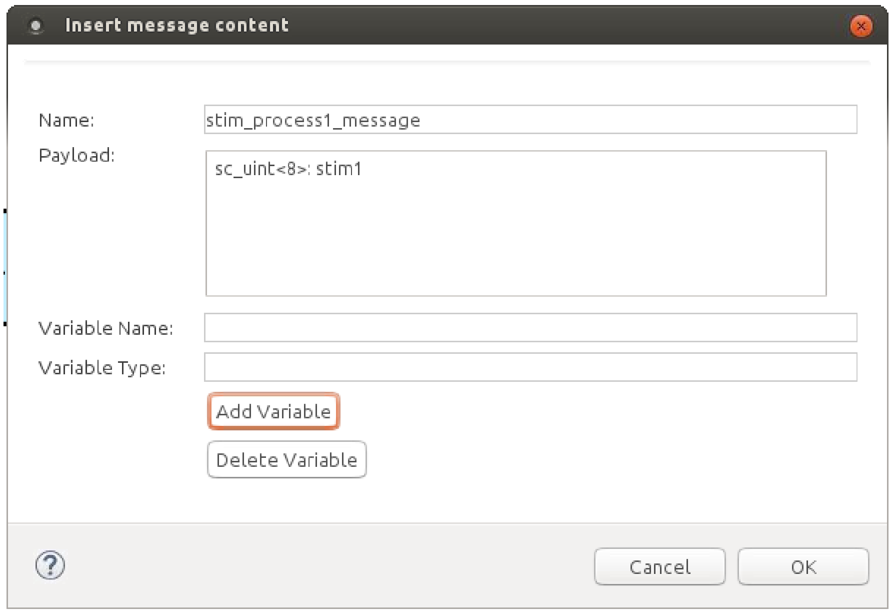
We can noticed two process are instantly created : stimulus and display. We cannot delete these process.
The GUI allows you to create four different elements :

- A package : package a set of process rely by channels
- Channel : allows the communication between the process and SubProcess
- SubProcess : allows to create process inside a package.
- Process : describe a part of your system or your entire system (at the moment of the design flow it's just a black box).



- At least you must give a name at all these elements and for each channel you have to add one variable (name + type).

- To add a variable, double click on a channel. This windows would appears:

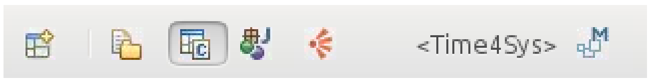


Name: stim_process1_message
Payload: sc_uint<8>: stim1
Variable Name:
Variable Type:

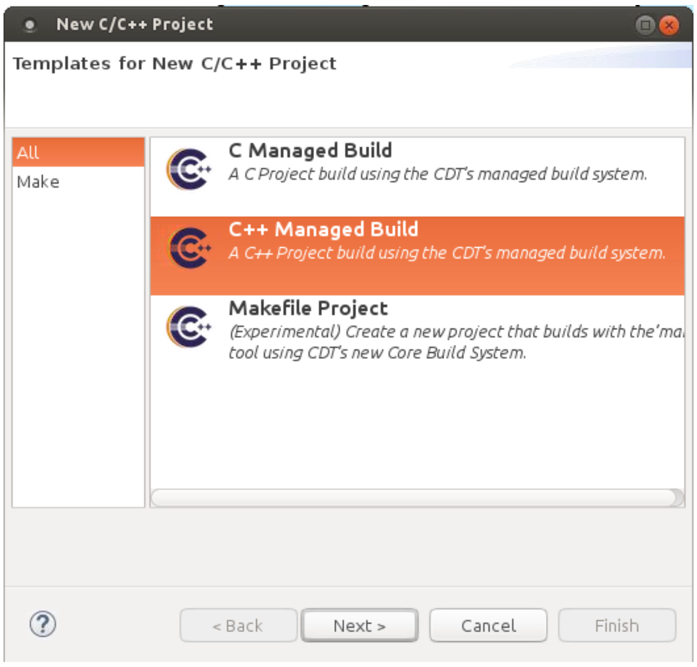
- Enter the *Variable Name* and the *Variable Type* and click "Add Variable".

➔ You just learn the **HEPSYCODE Modeling Language (HML).**

- Change the view to "C/C++"



- *File* > *New* > *C/C++ Project* > *C++ Managed Build* and click Next


New C/C++ Project

Templates for New C/C++ Project

All
Make

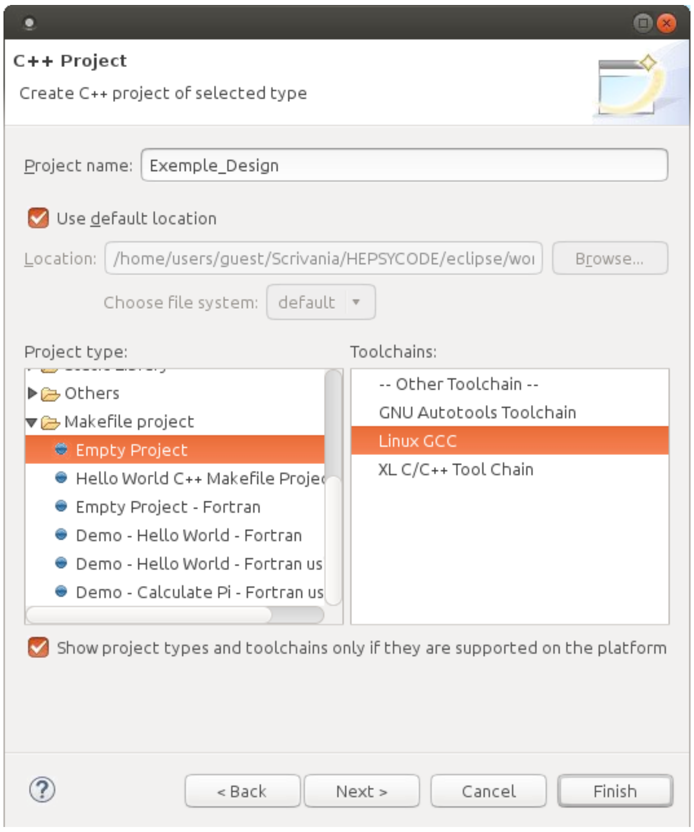
C Managed Build
A C Project build using the CDT's managed build system.

C++ Managed Build
A C++ Project build using the CDT's managed build system.

Makefile Project
(Experimental) Create a new project that builds with the 'ma
tool using CDT's new Core Build System.

< Back | Next > | Cancel | Finish

- Then choose *Makefile Project > Empty Project* with *Linux GCC compiler* and choose a name for your **Design Folder**


C++ Project

Create C++ project of selected type

Project name: Exemple_Design

☑ Use default location

Location: /home/users/guest/Scrivania/HEPSYCODE/eclipse/wo Browse...

Choose file system: default

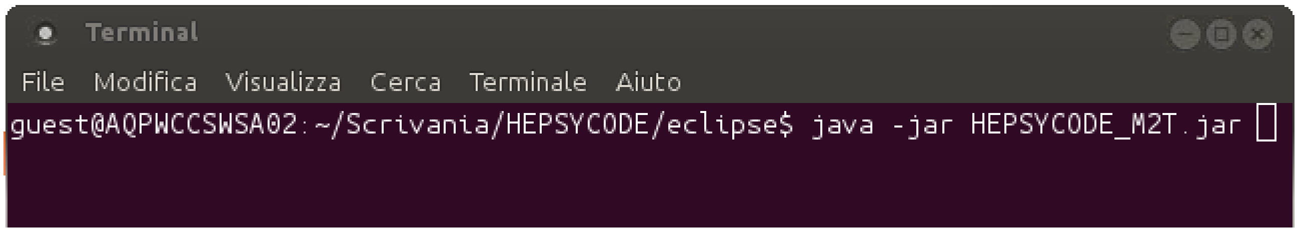
Project type:
- Others
- Makefile project
  - Empty Project
  - Hello World C++ Makefile Proje
  - Empty Project - Fortran
  - Demo - Hello World - Fortran
  - Demo - Hello World - Fortran us
  - Demo - Calculate Pi - Fortran us

Toolchains:
- -- Other Toolchain --
- GNU Autotools Toolchain
- Linux GCC
- XL C/C++ Tool Chain

☑ Show project types and toolchains only if they are supported on the platform

< Back | Next > | Cancel | Finish

- Open a Terminal in the eclipse folder
- Enter the command “java –jar HEPSYCODE_M2T.jar”



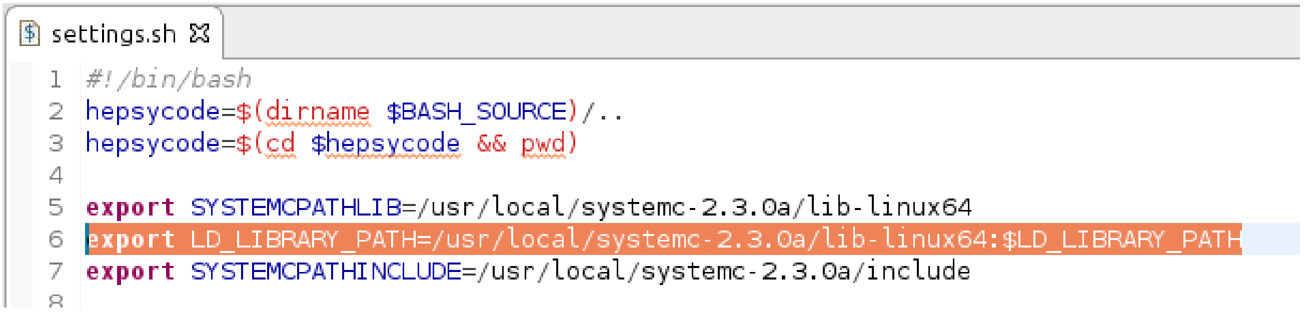
```
guest@AQPWCCSWSA02:~/Scrivania/HEPSYCODE/eclipse$ java -jar HEPSYCODE_M2T.jar
```

- A windows should appears you have to choose the file “*.hepsycode” in the Project Model folder
- Then choose the folder of destination (the C++ Project)

➔ This command would generate the skeleton of your system in **SystemC** from your HEPSYCODE MODEL describe in **HML** .

- Open the C++ Project Folder, you should have two news folders (*Functional* and *XML*)
  - If the news folders doesn’t appears you have to refresh your project explorer
  - Open the *src* folder (inside the Functional folder), you should be see the .cpp files generated from your model diagram.
    - If there are no .cpp files maybe your diagram is wrong. You can fix it like this :
      - Check if you are correctly save your Model diagram.
      - Check if all channels and process have a name.
      - Check if all channels have at least one variable.
    - If the problem is not fixed, you have to restart all.

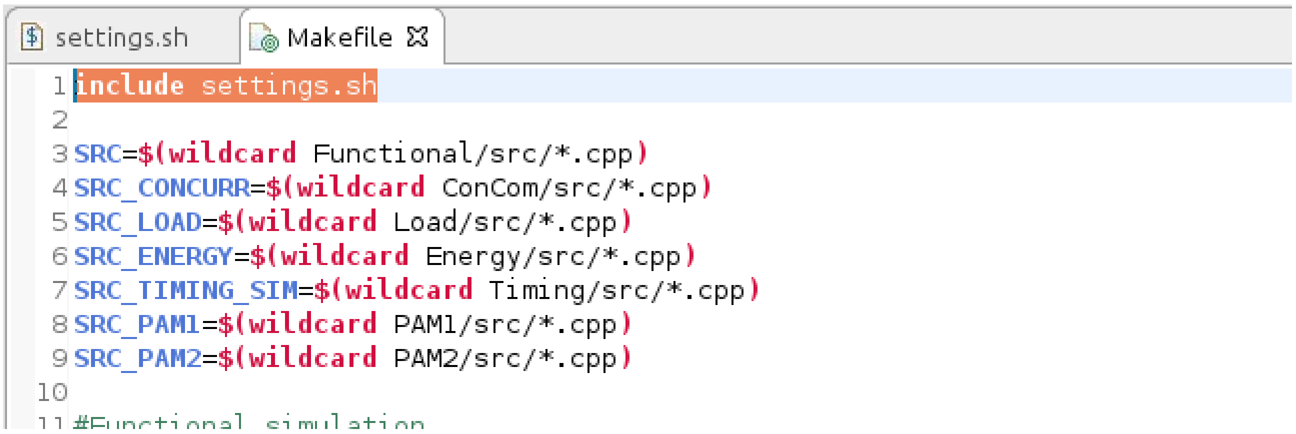
- In the C++ Project Folder
  - Open the file named “settings.sh” and add this line
    - export LD_LIBRARY_PATH=/usr/local/systemc-2.3.0a/lib-linux64:$LD_LIBRARY_PATH



```
#!/bin/bash
hepsycode=$(dirname $BASH_SOURCE)/..
hepsycode=$(cd $hepsycode && pwd)

export SYSTEMCPATHLIB=/usr/local/systemc-2.3.0a/lib-linux64
export LD_LIBRARY_PATH=/usr/local/systemc-2.3.0a/lib-linux64:$LD_LIBRARY_PATH
export SYSTEMCPATHINCLUDE=/usr/local/systemc-2.3.0a/include
```

- Open the Makefile and add this line
  - include settings.sh

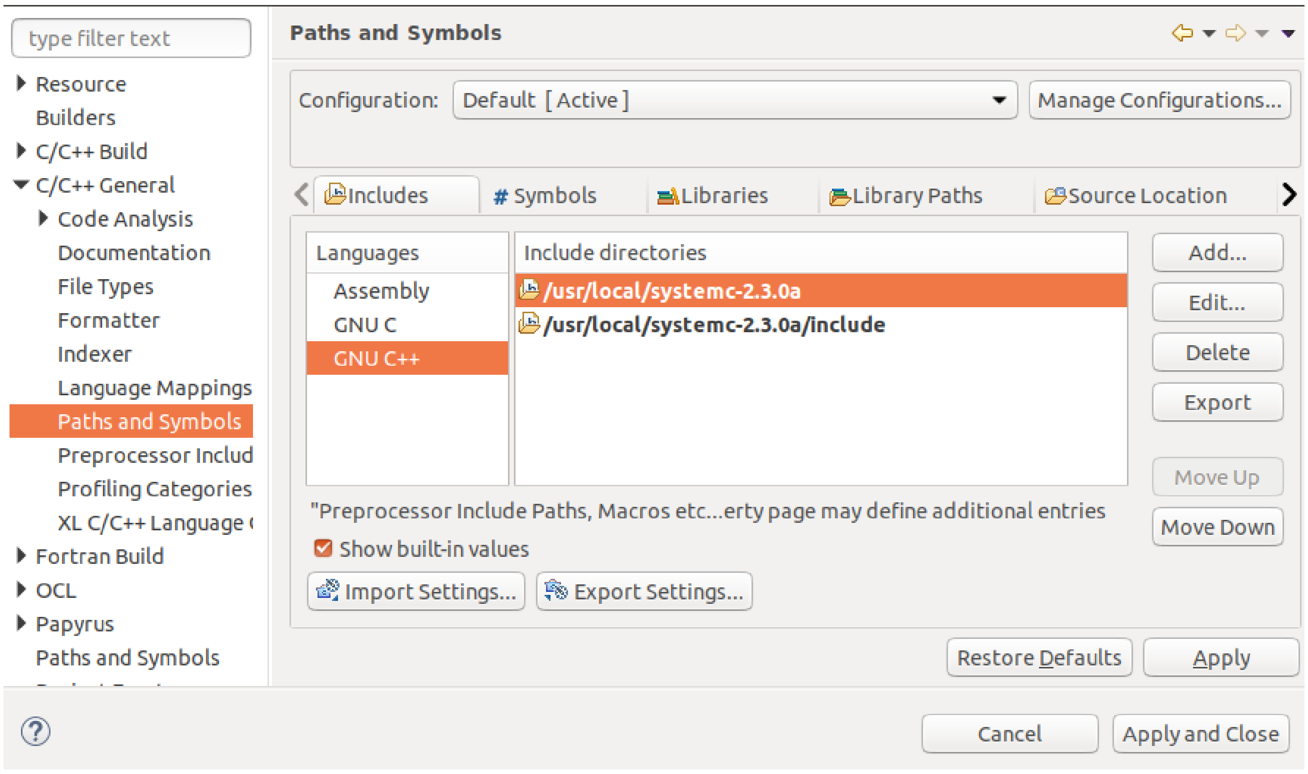


```
include settings.sh

SRC=$(wildcard Functional/src/*.cpp)
SRC_CONCURR=$(wildcard ConCom/src/*.cpp)
SRC_LOAD=$(wildcard Load/src/*.cpp)
SRC_ENERGY=$(wildcard Energy/src/*.cpp)
SRC_TIMING_SIM=$(wildcard Timing/src/*.cpp)
SRC_PAM1=$(wildcard PAM1/src/*.cpp)
SRC_PAM2=$(wildcard PAM2/src/*.cpp)

#Functional simulation
```

- To avoid some errors thereafter : Right Click on your C++ Project Folder > *Properties* > *C/C++ General* > *Paths and Symbols* > *GNU C++* and add the following path :



Configuration: Default [Active]

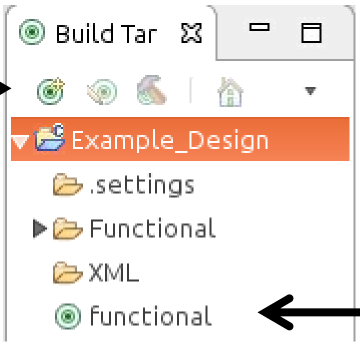
Languages	Include directories
Assembly	/usr/local/systemc-2.3.0a
GNU C	/usr/local/systemc-2.3.0a/include
GNU C++	

"Preprocessor Include Paths, Macros etc...erty page may define additional entries

☑ Show built-in values

- Create a New Build Target (It must be in the Design repository !)

Click here to create a New Build Target



The Builds Targets must be here !

- Create the following builds targets : *functional* , *functional_run* and *functional_clean* .
- Launch it and check if it works.

Now we can begin the HEPSYCODE Design Flow.

2.2. Functional Simulation

The goal of this step is to add the behavior of your system in each process you create before and verify that works. To do that open the systemc files (.cpp) generated by the tool. Be careful to change your variable name and keep the coherency of your Model.

2.3. Co-Analysis

- Create a new folder in your C++ Project: Right Click > New > Folder and name it : ConCom. Then create a src folder inside this last. (Do that each time we will create a new folder in the Design Flow).

- Copy-paste all the files in Functional/src and in HEPSY_DF/ConCom/Add in ConCom/src.

- In ConCom/src , delete sc_csp_channel.h.

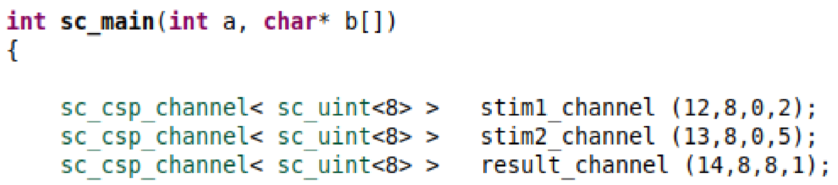
- Modify the “define.h”, replace the NPS and the NCH value expect to your design.

- In each functional files, add the macro “C” like below for each channel (except those are linked with STIMULUS or DISPLAY)

```
// write output values on channel
C result16_channel->write(acc); C
```

- Modify the “main.cpp” : copy-paste all the line missing than the “ConCom/src/main.cpp” of the FirFirGCD Project.

Don't miss to instantiate SystemManager before sc_main ()



```
int sc_main(int a, char* b[])
{

    sc_csp_channel< sc_uint<8> >   stim1_channel (12,8,0,2);
    sc_csp_channel< sc_uint<8> >   stim2_channel (13,8,0,5);
    sc_csp_channel< sc_uint<8> >   result_channel (14,8,8,1);
```

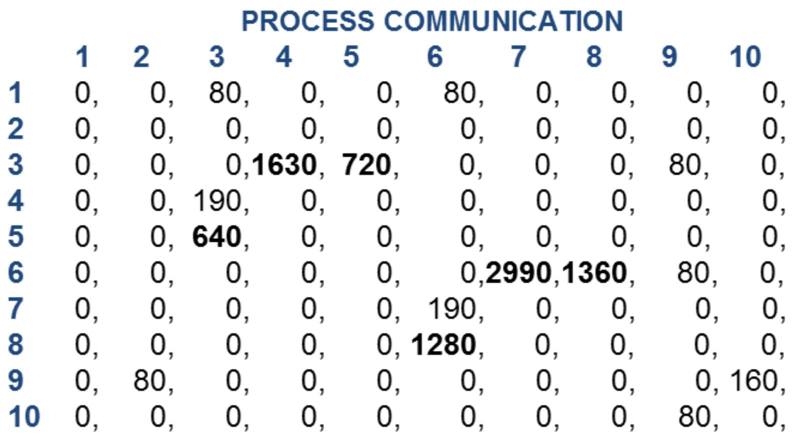
You have also to add the variable inside the parenthesis. This variable can be find in the "application.xml": To left to right you have to write : ***id, width, w_id*** and ***r_id.***

- Create three New Build Target : "concom", "concom_run" and "concom_clean"

We obtain several array like this:

```
PROCESSES NORMALIZED CONCURRENCY
           0        1        2        3        4        5        6        7        8        9
 0         0,       0,       0,       0,       0,       0,       0,       0,       0,       0,
 1         0,       0,       0,       0,       0,       0,       0,       0,       0,       0,
 2         0,       0,       0,0.105263,0.105263,0.105263,0.105263,0.105263,0.105263,0.105263,
 3         0,       0,       0,       0,0.263158,0.263158,0.263158,0.263158,0.263158,0.263158,
 4         0,       0,       0,       0,       0,0.421053,0.421053,0.421053,0.421053,0.421053,
 5         0,       0,       0,       0,       0,       0,0.526316,0.526316,0.526316,0.526316,
 6         0,       0,       0,       0,       0,       0,       0,0.684211,0.684211,0.684211,
 7         0,       0,       0,       0,       0,       0,       0,       0,0.842105,0.842105,
 8         0,       0,       0,       0,       0,       0,       0,       0,       0,       1,
 9         0,       0,       0,       0,       0,       0,       0,       0,       0,       0,
```

When the indicator is close or equal to 1, the concurrency between the two process are very high. So we can split them. For instance, in this case we can split the process : 7, 8, 9 (and maybe the 6).



PROCESS COMMUNICATION

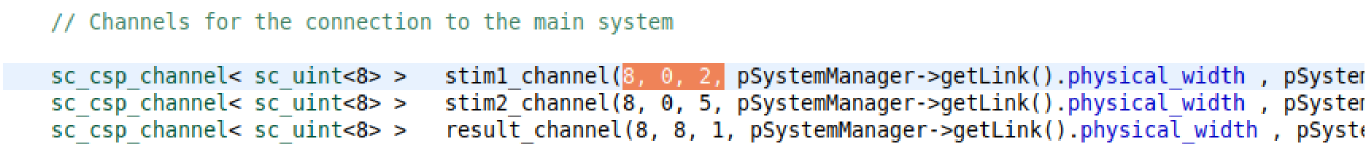
	1	2	3	4	5	6	7	8	9	10
1	0,	0,	80,	0,	0,	80,	0,	0,	0,	0,
2	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
3	0,	0,	0,	**1630**,	**720**,	0,	0,	0,	80,	0,
4	0,	0,	190,	0,	0,	0,	0,	0,	0,	0,
5	0,	0,	**640**,	0,	0,	0,	0,	0,	0,	0,
6	0,	0,	0,	0,	0,	0,	**2990**,	**1360**,	80,	0,
7	0,	0,	0,	0,	0,	190,	0,	0,	0,	0,
8	0,	0,	0,	0,	0,	**1280**,	0,	0,	0,	0,
9	0,	80,	0,	0,	0,	0,	0,	0,	0,	160,
10	0,	0,	0,	0,	0,	0,	0,	0,	80,	0,

This graph represents the number of interactions between two process (Normally it don't look like this). If this number is too high (compared to others), it would be better to implement these process together. For instance the analysis above show process 8 and 6 can be split be this graph say it can be better to implement these process on the same HW.

2.4. Co-Estimation

- Create a new folder in your C++ Project : Right Click > New > Folder and name it : Load.
- Copy-paste all the files in Functional/src (without sc_csp_channel.h and sc_csp_channel_ifs.h) and in HEPSY_DF/ConCom/Add in ConCom/src.
- Modify the "define.h", replace the NPS and the NCH value expect to your design.
- In "mainsystem.h" , replace "sc_csp_channel.h" by " sc_csp_channel_timing_L.h" and add "SystemManager_timing_L.h"
- Modify the "main.cpp" : copy-paste all the line missing than the "Load/src/main.cpp" of the FirFirGCD Project.

Don't miss to instantiate SystemManager before sc_main ()



```
// Channels for the connection to the main system

sc_csp_channel< sc_uint<8> >   stim1_channel(8, 0, 2, pSystemManager->getLink().physical_width , pSyster
sc_csp_channel< sc_uint<8> >   stim2_channel(8, 0, 5, pSystemManager->getLink().physical_width , pSyster
sc_csp_channel< sc_uint<8> >   result_channel(8, 8, 1, pSystemManager->getLink().physical_width , pSyste
```

As for Co-Analysis, you have to add this parenthesis with some variable at the beginning. To left to right you have to write : ***width, w_id*** and ***r_id.***

- Now open each files that describe the behaviour of your system and add the macro S(N).
  - N are the corresponding value of process of the files you modifying. To know the number open "application.xml" and see the id number of the process

```
// read parameters from channel
S(6) fir16e_p=fir16e_parameters_channel->read();

// fill local variables
S(6) sample_tmp=fir16e_p.sample_tmp;
S(6) for( unsigned j=0; j<TAP16; j++) coef[j]=fir16e_p.coef[j];
S(6) for( unsigned j=0; j<TAP16; j++) shift[j]=fir16e_p.shift[j];

// process
S(6) acc=sample_tmp*coef[0];

S(6) for(int i=TAP16-2; i>=0; i--)
{S(6)
    S(6) pro=shift[i]*coef[i+1];
    S(6) acc += pro;
S(6)}
```

- Create three New Build Target : load, load_run and load_clean

Double click on the build target for launch the **Co-Estimation** step of the Design Flow

2.5. PARTITIONING ARCHITECTURE DEFINITION AND MAPPING PHASE 1 (PAM1)

- Create a new folder in your C++ Project : Right Click > New > Folder and name it : *PAM1*.

- Copy-Paste all the files in PAM1_Parallel/src of the FirFirGCD project.

- Copy-Paste : “constraints.xml”, “instancesTL.xml”, “mapping.xml” and “PAM1parameter.xml” in your XML Folder.

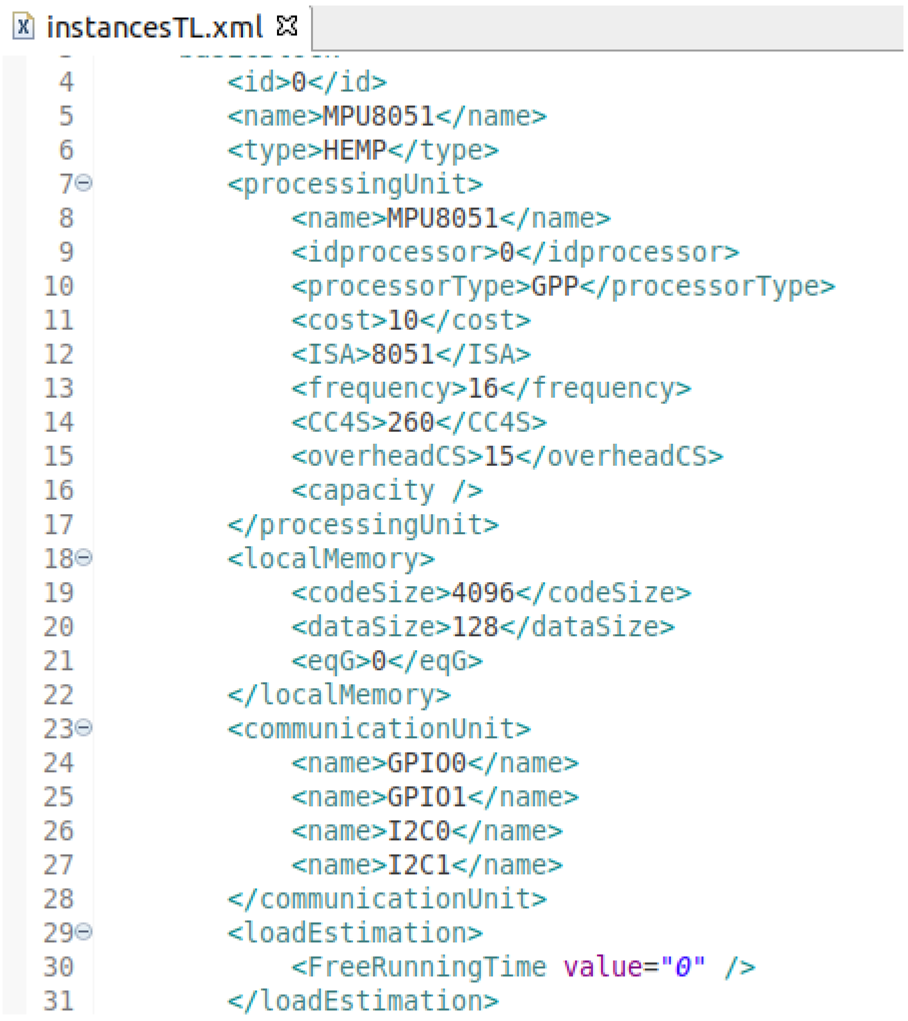
- Take a look at the “instancesTL.xml” files :

In this document we describe all the “basic block” and “logic link” of our architecture have with metrics. For basics blocks (BB) we distinguished four parts : *Processing Unit, Local Memory, Communication Unit* and *load Estimation.*

CC4CS : Clock Cycles for C Statement

The number of clock cycles needed to a specific processor technology to execute a common C statement

ISA : Instruction Set Architecture


instancesTL.xml

```
        <id>0</id>
        <name>MPU8051</name>
        <type>HEMP</type>
        <processingUnit>
            <name>MPU8051</name>
            <idprocessor>0</idprocessor>
            <processorType>GPP</processorType>
            <cost>10</cost>
            <ISA>8051</ISA>
            <frequency>16</frequency>
            <CC4S>260</CC4S>
            <overheadCS>15</overheadCS>
            <capacity />
        </processingUnit>
        <localMemory>
            <codeSize>4096</codeSize>
            <dataSize>128</dataSize>
            <eqG>0</eqG>
        </localMemory>
        <communicationUnit>
            <name>GPIO0</name>
            <name>GPIO1</name>
            <name>I2C0</name>
            <name>I2C1</name>
        </communicationUnit>
        <loadEstimation>
            <FreeRunningTime value="0" />
        </loadEstimation>
```

- Create three New Build Target : pam1, pam1_run and pam1_clean

2.6. PARTITIONING ARCHITECTURE DEFINITION AND MAPPING PHASE 2 (PAM2)

- Create a new folder in your C++ Project : Right Click > New > Folder and name it : *PAM2*.
- Copy-Paste all the files in PAM2_Parallel/src of the FirFirGCD project.
- Copy-Paste : "PAM2parameter.xml" in your XML Folder.
- Create three New Build Target : pam2, pam2_run and pam2_clean

END OF DOCUMENT