Coside Testcase: CMU_CLK

AE/PAI-IP | 2023-11-10

BOSCH

Coside Testcase: CMU_CLK
General Information about this document

Coside Testcase: CMU_CLK
Agenda

- Clocking introduction
- Coside introduction
- Testcase clarification

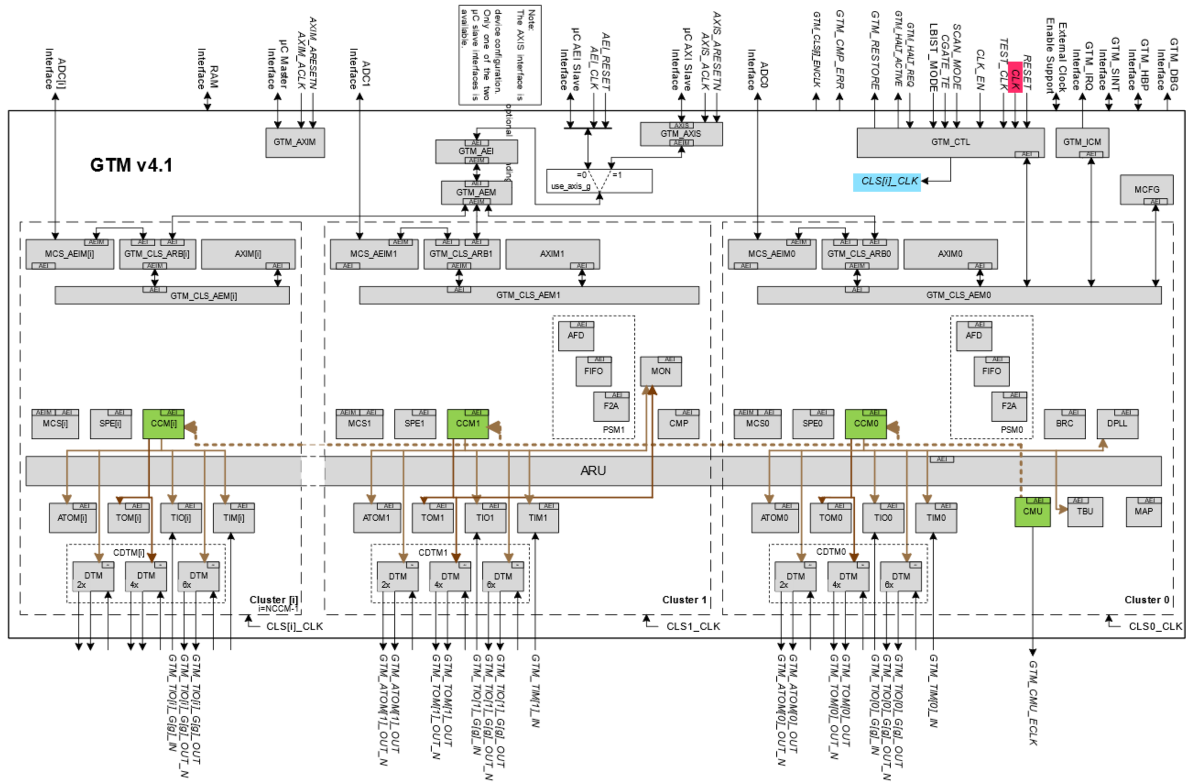
* *This document is based on the GTM specification and is available on our website: https://www.bosch-semiconductors.com/ip-modules/gtm-platform/gtm-generic-timer-ip-module/*

01

Clocking introduction

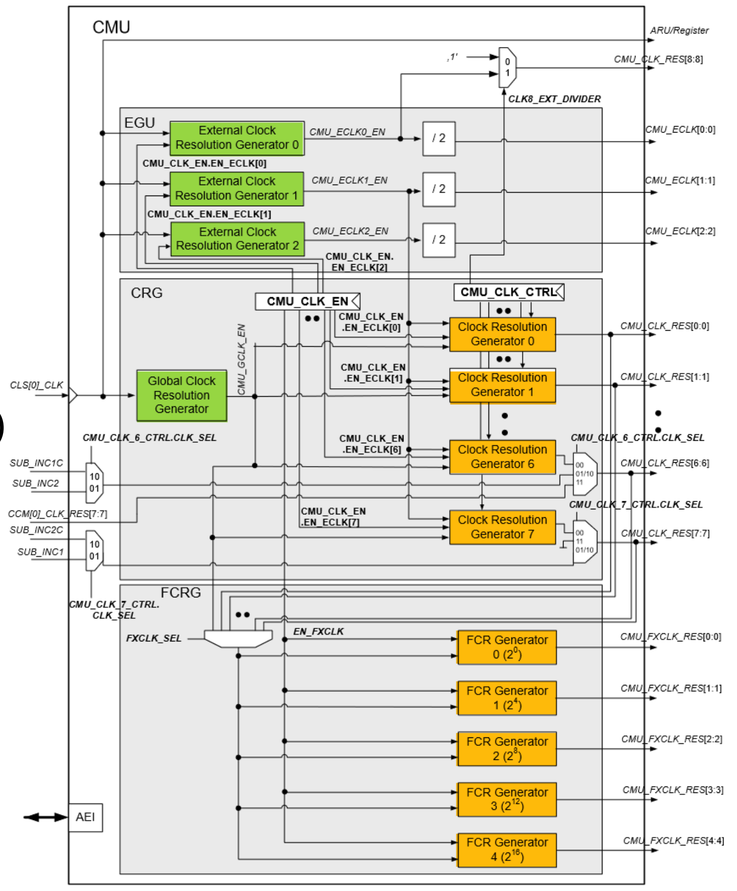
Clocking introduction
Configurability

- GTM IP clock
  - As GTM global clock
- Cluster clock
  - Configured in full or half GTM IP clock
- Module clock
  - Offered by Clock Managment Unit(CMU) and Cluster Configuration Module(CCM)
    - CMU uses internal clock divider to generate different clock resolutions
    - CCM samples the clock resolution signals from CMU and routes them to different modules within the cluster



Clocking introduction
CMU architecture

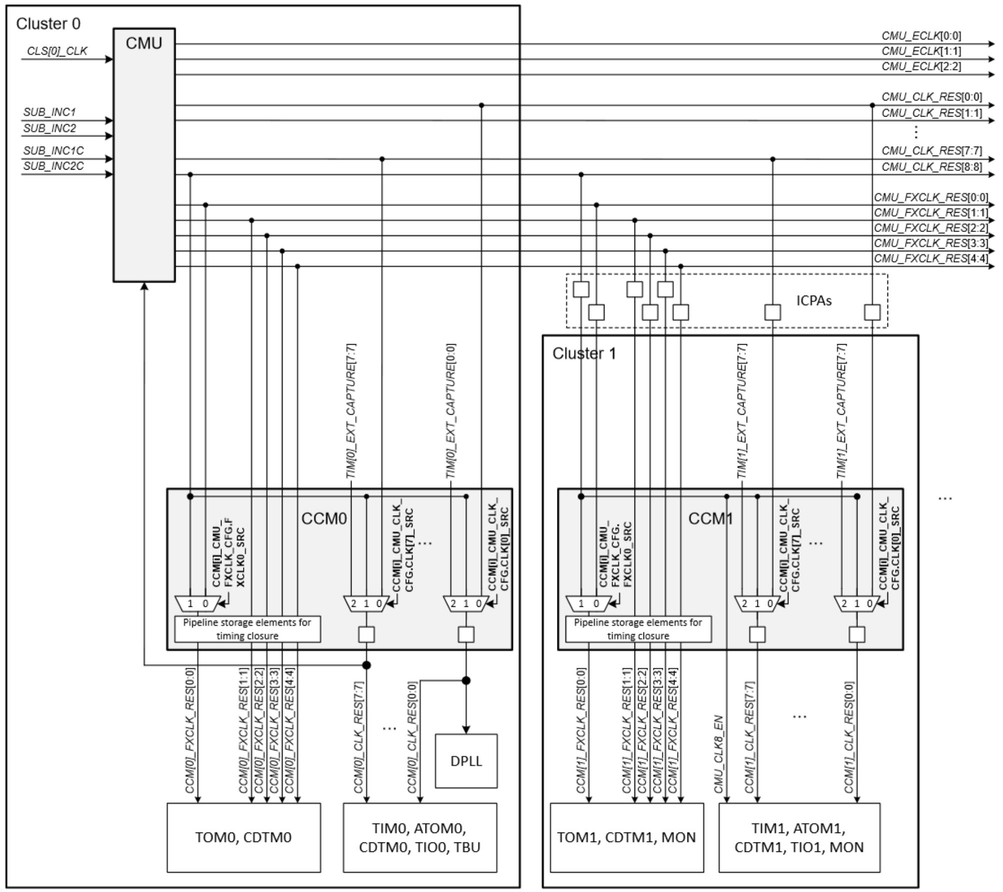
- The first layer of clock dividers
  - Global clock resolution generator
    - Configuration of the numerator CMU_GCLK_NUM and the denominator CMU_GCLK_DEN to generate a common clock resolution signal (CMU_GCLK_EN)
  - External clock resolution generator
    - Configuration of the numerator CMU_ECLKx_NUM and the denominator CMU_ECLKx_DEN to generate clock resolution signals (CMU_ECLKx_EN)
- The second layer of clock dividers
  - Clock resolution generator
    - Generates configurable clock resolution signals (CMU_CLK_RESx) with CMU_GCLK_EN or CMU_ECLK1_EN except for CMU_CLK_RES [8:8]
  - Fixed clock resolution generator
    - Generates fixed clock resolutions signals (CMU_FXCLK_RESx) based on the CMU_GCLK_EN signal or one of CMU_CLK_RESx resolution signals



Clocking introduction
Connectivity between CMU and CCM

- CMU generates CMU_ECLKx that can be used external of the GTM or microcontroller

- CCM selects the different clock resolution signals that are used internal of the GTM
  - For TIM, ATOM, CDTM, TIO, TBU modules
    - CMU_CLK_RESx
    - TIMx_EXT_CAPTUREx
  - For TOM, CDTM, MON modules
    - CMU_FXCLK_RESx



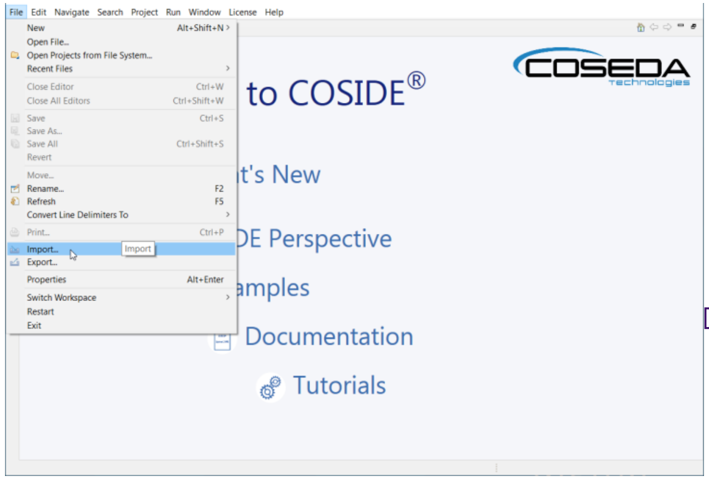
02

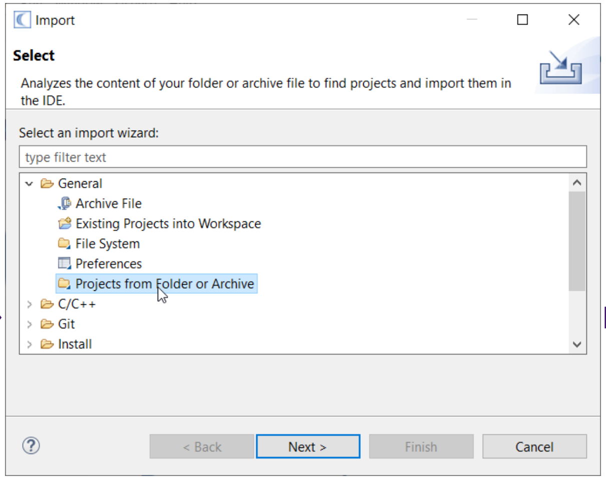
Coside introduction

Coside introduction
Installation

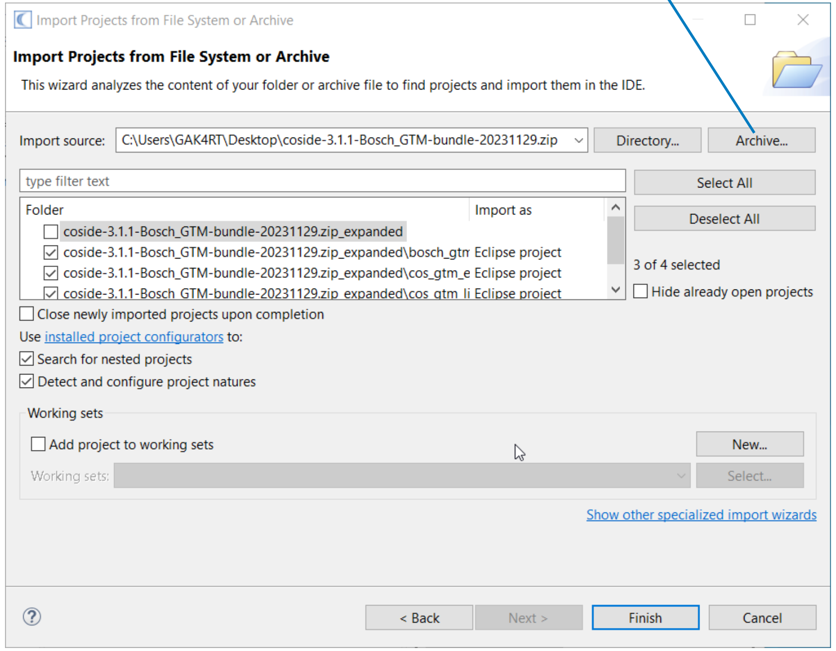
- Import the virtualGTM in a new Coside workspace

Select VirtualGTM.zip file in the correct archive





Analyzes the content of your folder or archive file to find projects and import them in the IDE.


Import Projects from File System or Archive
Import Projects from File System or Archive
Directory...
Archive...
type filter text
Select All
Deselect All
Folder
Import as
Hide already open projects
Working sets
New...
Select...
Show other specialized import wizards
< Back
Next >
Finish
Cancel

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\Users\GAK4RT\Desktop\coside-3.1.1-Bosch_GTM-bundle-20231129.zip

Folder	Import as
☐ coside-3.1.1-Bosch_GTM-bundle-20231129.zip_expanded	
☑ coside-3.1.1-Bosch_GTM-bundle-20231129.zip_expanded\bosch_gtm	Eclipse project
☑ coside-3.1.1-Bosch_GTM-bundle-20231129.zip_expanded\cos_gtm_e	Eclipse project
☑ coside-3.1.1-Bosch_GTM-bundle-20231129.zip_expanded\cos_gtm_li	Eclipse project

3 of 4 selected

☐ Close newly imported projects upon completion

Use installed project configurators to:

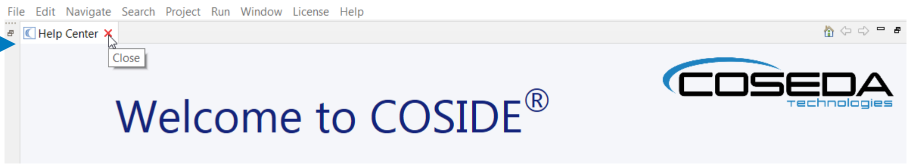
☑ Search for nested projects

☑ Detect and configure project natures

☐ Add project to working sets

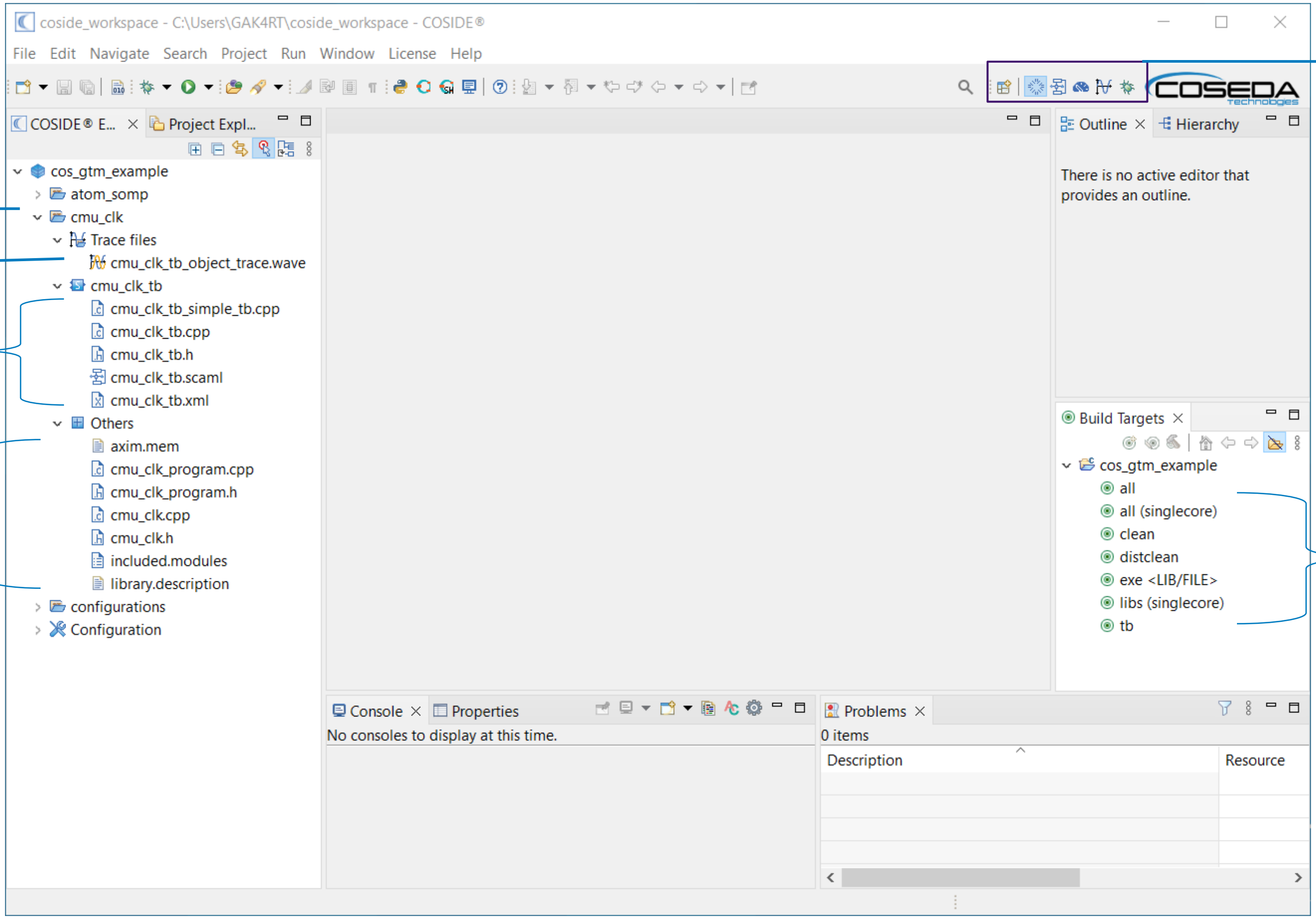
Working sets:

Close the help center page



BOSCH

Coside introduction
Structure



coside_workspace - C:\Users\GAK4RT\coside_workspace - COSIDE®

File Edit Navigate Search Project Run Window License Help

COSIDE® E... | Project Expl...

- cos_gtm_example
  - atom_somp
  - cmu_clk
    - Trace files
      - cmu_clk_tb_object_trace.wave
    - cmu_clk_tb
      - cmu_clk_tb_simple_tb.cpp
      - cmu_clk_tb.cpp
      - cmu_clk_tb.h
      - cmu_clk_tb.scaml
      - cmu_clk_tb.xml
    - Others
      - axim.mem
      - cmu_clk_program.cpp
      - cmu_clk_program.h
      - cmu_clk.cpp
      - cmu_clk.h
      - included.modules
      - library.description
  - configurations
  - Configuration

Outline | Hierarchy

There is no active editor that provides an outline.

Build Targets

- cos_gtm_example
  - all
  - all (singlecore)
  - clean
  - distclean
  - exe <LIB/FILE>
  - libs (singlecore)
  - tb

Console | Properties

No consoles to display at this time.

Problems

0 items

Description	Resource

Coside introduction
Execution

1.Write and save testcase name in **included.libs**

(Skip this step if the name already in it)

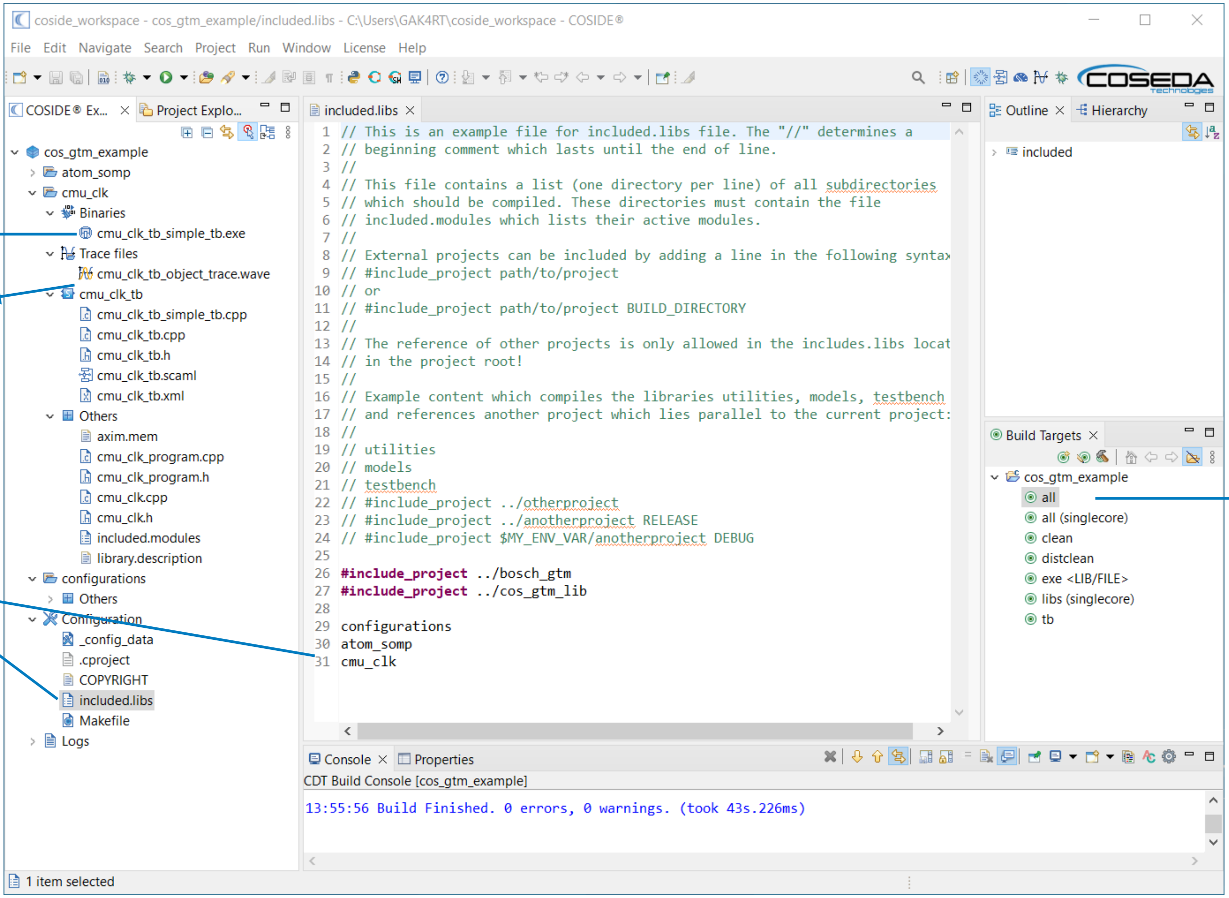
2. Double-click **all** button to build cos_gtm_user

3. Double-click the **.exe file** to execute the test case

4. Click **OK** button

5. Open **.wave files** to see the simulation waveforms

6. Click **Yes** button

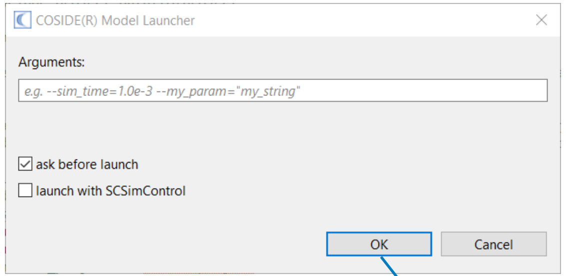


```
// This is an example file for included.libs file. The "//" determines a
// beginning comment which lasts until the end of line.
//
// This file contains a list (one directory per line) of all subdirectories
// which should be compiled. These directories must contain the file
// included.modules which lists their active modules.
//
// External projects can be included by adding a line in the following synta
// #include_project path/to/project
// or
// #include_project path/to/project BUILD_DIRECTORY
//
// The reference of other projects is only allowed in the includes.libs locat
// in the project root!
//
// Example content which compiles the libraries utilities, models, testbench
// and references another project which lies parallel to the current project:
//
// utilities
// models
// testbench
// #include_project ../otherproject
// #include_project ../anotherproject RELEASE
// #include_project $MY_ENV_VAR/anotherproject DEBUG

#include_project ../bosch_gtm
#include_project ../cos_gtm_lib

configurations
atom_somp
cmu_clk
```

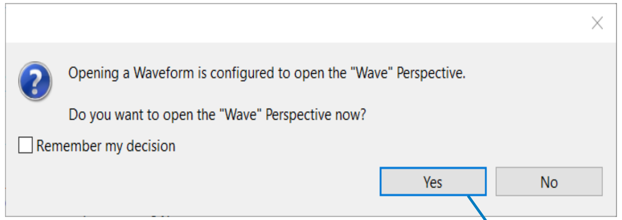
```
13:55:56 Build Finished. 0 errors, 0 warnings. (took 43s.226ms)
```


COSIDE(R) Model Launcher

Arguments: e.g. --sim_time=1.0e-3 --my_param="my_string"

[✓] ask before launch

[ ] launch with SCSimControl


Opening a Waveform is configured to open the "Wave" Perspective.

Do you want to open the "Wave" Perspective now?

[ ] Remember my decision

Coside introduction

Execution

10. Save it

Zoom Help Tools

11. Click **Modeling** button to change the layout

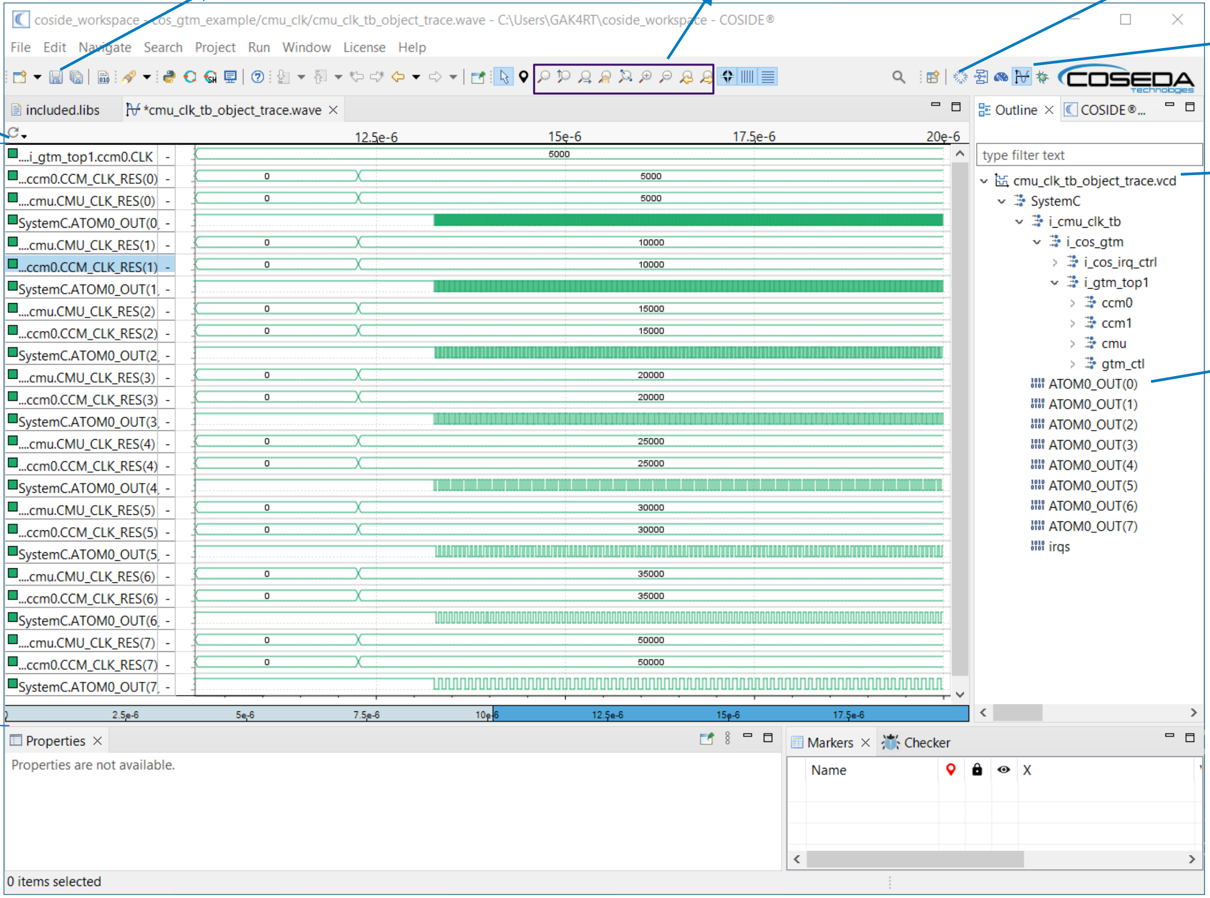
12. Click **Wave** button to change the layout back

Refresh button

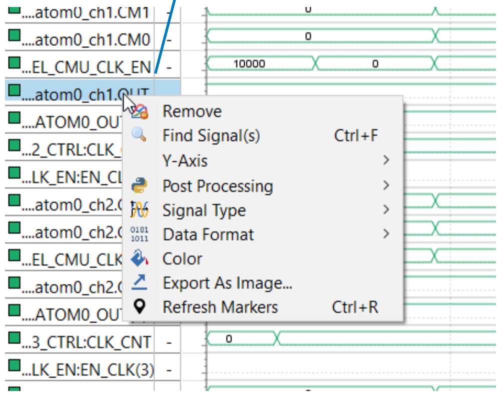
7. Find the observed signals in **.vcd file**

8. Select the observed signals and click right or drag to add them in **.wave file**

Simulation waveforms



9. Select the signal and click right to modify the waveform



BOSCH

03

Testcases clarification

BOSCH

Testcases clarification
Tasks

- Generate clock resolution signals (CMU_CLK_RESx) via two layers of clock dividers in the CMU
- Visualize the behaviour of the CMU clock resolution on ATOM outputs
  - Each ATOM generates a PWM with duty cycle 50 % and period of 2 clock resolutions
- ATOM clock configuration
  - ATOM CH0 uses CCM_CLK_RES[0:0]
  - ATOM CH1 uses CCM_CLK_RES[1:1]
  - ATOM CH2 uses CCM_CLK_RES[2:2]
  - ATOM CH3 uses CCM_CLK_RES[3:3]
  - ATOM CH4 uses CCM_CLK_RES[4:4]
  - ATOM CH5 uses CCM_CLK_RES[5:5]
  - ATOM CH6 uses CCM_CLK_RES[6:6]
  - ATOM CH7 uses CCM_CLK_RES[7:7]

Testcases clarification
LAB overview

- LAB0
  - GTM IP clock is 200 MHz, Cluster clock is full GTM IP clock
  - External Clock Resolution Generator0 = 7/17
  - Global Clock Resolution Generator = 200/100
- LAB1
  - Based on LAB0 only modify Global Clock Resolution Generator = 80/100
- LAB2
  - Based on LAB0 only modify cluster clock is half GTM IP clock

GTM IP clock = 200 MHz	LAB0	LAB1	LAB2
Cluster Clock	200 MHz	200 MHz	100 MHz
External Clock Resolution Generator0 (CMU_ECLK0_DEN/ CMU_ECLK0_NUM)	7/17	7/17	7/17
CMU_ECLK[0:0]	41.17 MHz	41.17 MHz	41.17 MHz
Global clock Resolution Generator (CMU_GCLK_DEN/ CMU_GCLK_NUM)	200/200	80/200	200/200
CMU_GCLK_EN	200 MHz	80 MHz	100 MHz

Testcases clarification
LAB overview

- CMU_TEST0
  - External Clock Resolution Generator1 = 200/200
  - Clock Resolution Generator0 =1, with CMU_GCLK_EN
  - Clock Resolution Generator1 =2, with CMU_GCLK_EN
  - Clock Resolution Generator2 =3, with CMU_GCLK_EN
  - Clock Resolution Generator3 =4, with CMU_GCLK_EN
  - Clock Resolution Generator4 =5, with CMU_GCLK_EN
  - Clock Resolution Generator5 =6, with CMU_GCLK_EN
  - Clock Resolution Generator6 =7, with CMU_GCLK_EN
  - Clock Resolution Generator7 =10, with CMU_ECLK1_EN
  - CMU_CLK_RESx as CCM0_CLK_RESx signal within Cluster.

Testcases clarification
LAB overview

- CMU_TEST1
  - External Clock Resolution Generator1 = 97/201
  - External Clock Resolution Generator2 = 101/403
  - Clock Resolution Generators Configuration same as CMU_TEST0
  - CMU_CLK_RESx as CCM0_CLK_RESx signal within Cluster and then the signal is modified as follows
    - CMU_CLK_RES[2:2]as CCM0_CLK_RES[2:2] signal within Cluster
    - CMU_CLK_RES[5:5]as CCM0_CLK_RES[5:5] signal within Cluster
    - TIM_EXT_CAPTURE as CCM0_CLK_RES[0:0] , CCM0_CLK_RES[3:3] , CCM0_CLK_RES[6:6] signal within Cluster
    - CMU_CLK_RES [8:8] as CCM0_CLK_RES[1:1] , CCM0_CLK_RES[4:4] , CCM0_CLK_RES[7:7] signal within Cluster
- LAB 3
  - Only for investigating the CMU_ECLK1 signal
  - GTM IP clock is 200 MHz, Cluster clock is half GTM IP clock
  - Visualize the behaviour of the CMU_ECLK1 clock resolution on ATOM_CH7 outputs

Testcases clarification
LAB overview

CMU_TEST0	LAB0	LAB1	LAB2
Cluster Clock	200 MHz	200 MHz	100 MHz
External Clock Resolution Generator0 (CMU_ECLK0_DEN/ CMU_ECLK0_NUM)	7/17	7/17	7/17
CMU_ECLK[0:0]	41.17 MHz	41.17 MHz	41.17 MHz
CMU_CLK_RES[8:8]	82.24 MHz	82.24 MHz	82.24 MHz
External Clock Resolution Generator1 (CMU_ECLK1_DEN/ CMU_ECLK1_NUM)	200/200	200/200	200/200
CMU_ECLK1_EN	200 MHz	200 MHz	100 MHz
Global clock Resolution Generator (CMU_GCLK_DEN/ CMU_GCLK_NUM)	200/200	80/200	200/200
CMU_GCLK_EN	200 MHz	80 MHz	100 MHz
CCM_CLK_RES[0:0] = CMU_CLK_RES[0:0] (Clock Resolution Generator=1, with CMU_GCLK_EN)	200 MHz	80 MHz	100 MHz
CCM_CLK_RES[1:1] = CMU_CLK_RES[1:1] (Clock Resolution Generator=2, with CMU_GCLK_EN)	100 MHz	40 MHz	50 MHz
CCM_CLK_RES[2:2] = CMU_CLK_RES[2:2] (Clock Resolution Generator=3, with CMU_GCLK_EN)	66.67 MHz	26.67 MHz	33.34 MHz
CCM_CLK_RES[3:3] = CMU_CLK_RES[3:3] (Clock Resolution Generator=4, with CMU_GCLK_EN)	50 MHz	20 MHz	25 MHz
CCM_CLK_RES[4:4] = CMU_CLK_RES[4:4] (Clock Resolution Generator=5, with CMU_GCLK_EN)	40 MHz	16 MHz	20 MHz
CCM_CLK_RES[5:5] = CMU_CLK_RES[5:5] (Clock Resolution Generator=6, with CMU_GCLK_EN)	33.33 MHz	13.33 MHz	16.67 MHz
CCM_CLK_RES[6:6] = CMU_CLK_RES[6:6] (Clock Resolution Generator=7, with CMU_GCLK_EN)	28.57 MHz	11.42 MHz	14.28 MHz
CCM_CLK_RES[7:7] = CMU_CLK_RES[7:7] (Clock Resolution Generator= 10, with CMU_ECLK1_EN)	20 MHz	20 MHz	10 MHz

BOSCH

Testcases clarification
LAB overview

CMU_TEST1	LAB0	LAB1	LAB2
Cluster Clock	200 MHz	200 MHz	100 MHz
External Clock Resolution Generator0 (CMU_ECLK0_DEN/ CMU_ECLK0_NUM)	7/17	7/17	7/17
CMU_ECLK[0:0]	41.17 MHz	41.17 MHz	41.17 MHz
CMU_CLK_RES[8:8]	82.24 MHz	82.24 MHz	82.24 MHz
External Clock Resolution Generator1 (CMU_ECLK1_DEN/ CMU_ECLK1_NUM)	97/201	97/201	97/201
CMU_ECLK1_EN	96.51 MHz	96.51 MHz	96.51 MHz
Global clock Resolution Generator (CMU_GCLK_DEN/ CMU_GCLK_NUM)	200/200	80/200	200/200
CMU_GCLK_EN	200 MHz	80 MHz	100 MHz
CCM_CLK_RES[0:0] = TIM_EXT_CAPTURE signal	0 MHz	0 MHz	0 MHz
CCM_CLK_RES[1:1] = CMU_CLK_RES[8:8]	82.24 MHz	82.24 MHz	82.24 MHz
CCM_CLK_RES[2:2] = CMU_CLK_RES[2:2] (Clock Resolution Generator=3, with CMU_GCLK_EN)	66.67 MHz	26.67 MHz	33.34 MHz
CCM_CLK_RES[3:3] = TIM_EXT_CAPTURE signal	0 MHz	0 MHz	0 MHz
CCM_CLK_RES[4:4] = CMU_CLK_RES[8:8]	82.24 MHz	82.24 MHz	82.24 MHz
CCM_CLK_RES[5:5] = CMU_CLK_RES[5:5] (Clock Resolution Generator=6, with CMU_GCLK_EN)	33.33 MHz	13.33 MHz	16.67 MHz
CCM_CLK_RES[6:6] = TIM_EXT_CAPTURE signal	0 MHz	0 MHz	0 MHz
CCM_CLK_RES[7:7] = CMU_CLK_RES[8:8]	82.24 MHz	82.24 MHz	82.24 MHz

Testcases clarification
Code in cmu_clk.cpp

LAB definition

To execute LAB1, modify the code to #define LAB 0x1 and then recompile it

Cluster clock configuration

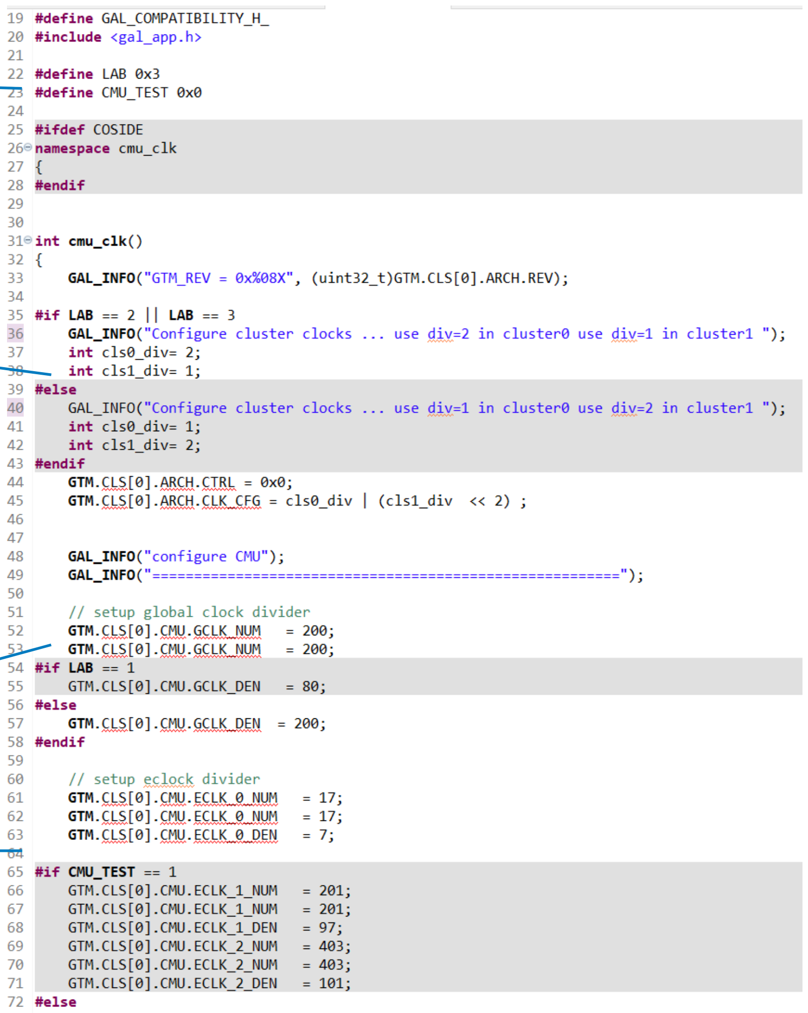
Cls_clk is divided by 1 which means it works at full (GTM) clk (200 MHz)

Global Clock Resolution Generators configuration

A secure way for altering the values is writing twice to the register CMU_GCLK_NUM followed by a single write to register CMU_GCLK_DEN

External Clock Resolution Generators configuration

A secure way for altering the values is writing twice to the register CMU_ECLK_NUM followed by a single write to register CMU_ECLK_DEN



```
#define GAL_COMPATIBILITY_H_
#include <gal_app.h>

#define LAB 0x3
#define CMU_TEST 0x0

#ifdef COSIDE
namespace cmu_clk
{
#endif


int cmu_clk()
{
    GAL_INFO("GTM_REV = 0x%08X", (uint32_t)GTM.CLS[0].ARCH.REV);

#if LAB == 2 || LAB == 3
    GAL_INFO("Configure cluster clocks ... use div=2 in cluster0 use div=1 in cluster1 ");
    int cls0_div= 2;
    int cls1_div= 1;
#else
    GAL_INFO("Configure cluster clocks ... use div=1 in cluster0 use div=2 in cluster1 ");
    int cls0_div= 1;
    int cls1_div= 2;
#endif
    GTM.CLS[0].ARCH.CTRL = 0x0;
    GTM.CLS[0].ARCH.CLK_CFG = cls0_div | (cls1_div  << 2) ;


    GAL_INFO("configure CMU");
    GAL_INFO("========================================================");

    // setup global clock divider
    GTM.CLS[0].CMU.GCLK_NUM   = 200;
    GTM.CLS[0].CMU.GCLK_NUM   = 200;
#if LAB == 1
    GTM.CLS[0].CMU.GCLK_DEN   = 80;
#else
    GTM.CLS[0].CMU.GCLK_DEN  = 200;
#endif

    // setup eclock divider
    GTM.CLS[0].CMU.ECLK_0_NUM   = 17;
    GTM.CLS[0].CMU.ECLK_0_NUM   = 17;
    GTM.CLS[0].CMU.ECLK_0_DEN   = 7;

#if CMU_TEST == 1
    GTM.CLS[0].CMU.ECLK_1_NUM   = 201;
    GTM.CLS[0].CMU.ECLK_1_NUM   = 201;
    GTM.CLS[0].CMU.ECLK_1_DEN   = 97;
    GTM.CLS[0].CMU.ECLK_2_NUM   = 403;
    GTM.CLS[0].CMU.ECLK_2_NUM   = 403;
    GTM.CLS[0].CMU.ECLK_2_DEN   = 101;
#else
```

Testcases clarification
Code in cmu_clk.cpp

External Clock Resolution Generators configuration

Clock Resolution Generator with CMU_ECLK1_EN

Clock Resolution Generators configuration

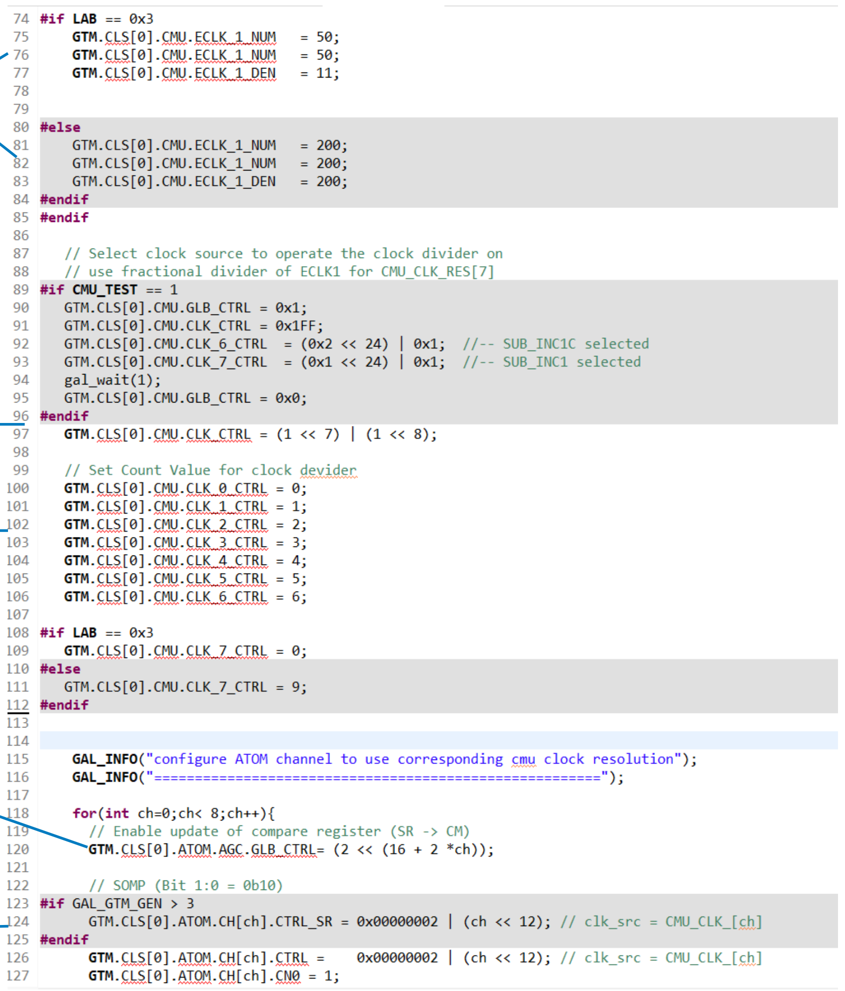
Enabling UPEN

Values in shadow registers can be updated in the operation registers

ATOM Channels configuration

CH0 uses the CCM_CLK_RES[0:0], CH1 uses the CMU_CLK_RES[1:1], …

All channels work in the SOMP mode



```
#if LAB == 0x3
    GTM.CLS[0].CMU.ECLK_1_NUM   = 50;
    GTM.CLS[0].CMU.ECLK_1_NUM   = 50;
    GTM.CLS[0].CMU.ECLK_1_DEN   = 11;


#else
    GTM.CLS[0].CMU.ECLK_1_NUM   = 200;
    GTM.CLS[0].CMU.ECLK_1_NUM   = 200;
    GTM.CLS[0].CMU.ECLK_1_DEN   = 200;
#endif
#endif

   // Select clock source to operate the clock divider on
   // use fractional divider of ECLK1 for CMU_CLK_RES[7]
#if CMU_TEST == 1
   GTM.CLS[0].CMU.GLB_CTRL = 0x1;
   GTM.CLS[0].CMU.CLK_CTRL = 0x1FF;
   GTM.CLS[0].CMU.CLK_6_CTRL  = (0x2 << 24) | 0x1;  //-- SUB_INC1C selected
   GTM.CLS[0].CMU.CLK_7_CTRL  = (0x1 << 24) | 0x1;  //-- SUB_INC1 selected
   gal_wait(1);
   GTM.CLS[0].CMU.GLB_CTRL = 0x0;
#endif
   GTM.CLS[0].CMU.CLK_CTRL = (1 << 7) | (1 << 8);

   // Set Count Value for clock devider
   GTM.CLS[0].CMU.CLK_0_CTRL = 0;
   GTM.CLS[0].CMU.CLK_1_CTRL = 1;
   GTM.CLS[0].CMU.CLK_2_CTRL = 2;
   GTM.CLS[0].CMU.CLK_3_CTRL = 3;
   GTM.CLS[0].CMU.CLK_4_CTRL = 4;
   GTM.CLS[0].CMU.CLK_5_CTRL = 5;
   GTM.CLS[0].CMU.CLK_6_CTRL = 6;

#if LAB == 0x3
   GTM.CLS[0].CMU.CLK_7_CTRL = 0;
#else
   GTM.CLS[0].CMU.CLK_7_CTRL = 9;
#endif



    GAL_INFO("configure ATOM channel to use corresponding cmu clock resolution");
    GAL_INFO("=======================================================");

    for(int ch=0;ch< 8;ch++){
      // Enable update of compare register (SR -> CM)
      GTM.CLS[0].ATOM.AGC.GLB_CTRL= (2 << (16 + 2 *ch));

      // SOMP (Bit 1:0 = 0b10)
#if GAL_GTM_GEN > 3
      GTM.CLS[0].ATOM.CH[ch].CTRL_SR = 0x00000002 | (ch << 12); // clk_src = CMU_CLK_[ch]
#endif
      GTM.CLS[0].ATOM.CH[ch].CTRL =    0x00000002 | (ch << 12); // clk_src = CMU_CLK_[ch]
      GTM.CLS[0].ATOM.CH[ch].CN0 = 1;
```

Testcases clarification

Code in cmu_clk.cpp

PWM parameters written in shadow registers

Using FUPD signal update

ATOM channels enable

Global enable and output enable

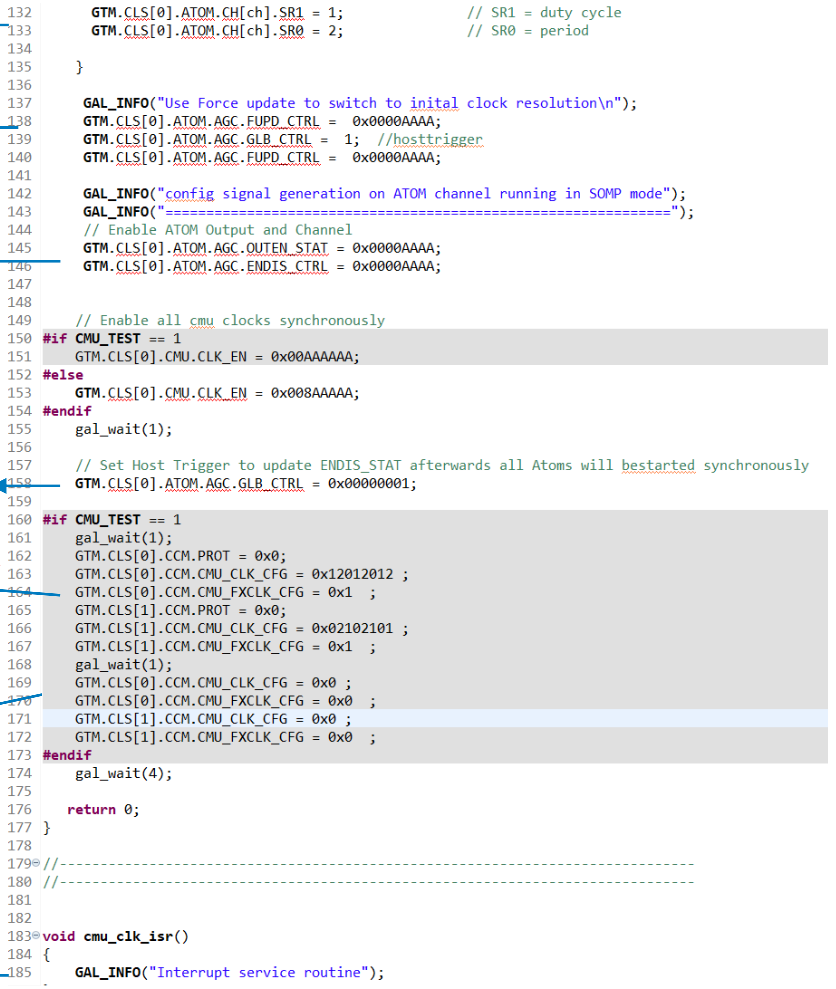
Trigger request

Updated ENDIS_STAT to make the ATOM Channels are enabled atomic

Modify theTIM_EXT_CAPTURE, CMU_CLK_RES[8:8] as CCM_CLK_RESx signal within the cluster

Modify the CMU_CLK_RESx as CCM0_CLK_RESx signal within Cluster

Placeholder for interrupt routines



```
        GTM.CLS[0].ATOM.CH[ch].SR1 = 1;                // SR1 = duty cycle
        GTM.CLS[0].ATOM.CH[ch].SR0 = 2;                // SR0 = period

    }

     GAL_INFO("Use Force update to switch to inital clock resolution\n");
     GTM.CLS[0].ATOM.AGC.FUPD_CTRL =  0x0000AAAA;
     GTM.CLS[0].ATOM.AGC.GLB_CTRL =  1;  //hosttrigger
     GTM.CLS[0].ATOM.AGC.FUPD_CTRL =  0x0000AAAA;

     GAL_INFO("config signal generation on ATOM channel running in SOMP mode");
     GAL_INFO("==========================================================");
     // Enable ATOM Output and Channel
     GTM.CLS[0].ATOM.AGC.OUTEN_STAT = 0x0000AAAA;
     GTM.CLS[0].ATOM.AGC.ENDIS_CTRL = 0x0000AAAA;


    // Enable all cmu clocks synchronously
#if CMU_TEST == 1
    GTM.CLS[0].CMU.CLK_EN = 0x00AAAAAA;
#else
    GTM.CLS[0].CMU.CLK_EN = 0x008AAAAA;
#endif
    gal_wait(1);

    // Set Host Trigger to update ENDIS_STAT afterwards all Atoms will bestarted synchronously
    GTM.CLS[0].ATOM.AGC.GLB_CTRL = 0x00000001;

#if CMU_TEST == 1
    gal_wait(1);
    GTM.CLS[0].CCM.PROT = 0x0;
    GTM.CLS[0].CCM.CMU_CLK_CFG = 0x12012012 ;
    GTM.CLS[0].CCM.CMU_FXCLK_CFG = 0x1   ;
    GTM.CLS[1].CCM.PROT = 0x0;
    GTM.CLS[1].CCM.CMU_CLK_CFG = 0x02102101 ;
    GTM.CLS[1].CCM.CMU_FXCLK_CFG = 0x1   ;
    gal_wait(1);
    GTM.CLS[0].CCM.CMU_CLK_CFG = 0x0 ;
    GTM.CLS[0].CCM.CMU_FXCLK_CFG = 0x0   ;
    GTM.CLS[1].CCM.CMU_CLK_CFG = 0x0 ;
    GTM.CLS[1].CCM.CMU_FXCLK_CFG = 0x0   ;
#endif
    gal_wait(4);

   return 0;
}

//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------


void cmu_clk_isr()
{
    GAL_INFO("Interrupt service routine");
```

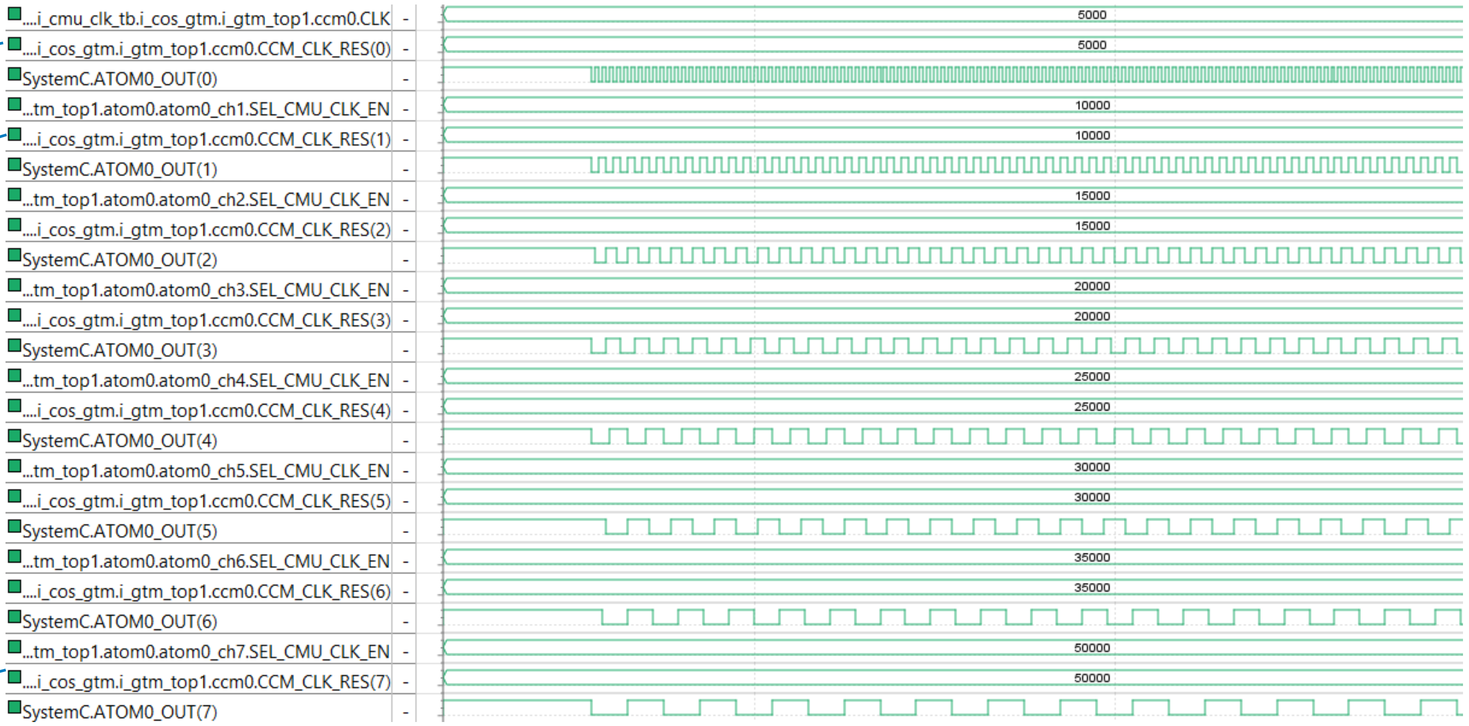
Testcases clarification
Waveform

- LAB0 + CMU_TEST0
  - Cluster clock is 200 MHz (~ 5 ns)
  - CMU_GCLK_EN is 200 MHz

ATOM_CH0 clk is 200 MHz

ATOM_CH1 clk is 100 MHz

ATOM_CH7 clk is 20 MHz



....i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CLK - 5000
....i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(0) - 5000
SystemC.ATOM0_OUT(0) -
...tm_top1.atom0.atom0_ch1.SEL_CMU_CLK_EN - 10000
....i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(1) - 10000
SystemC.ATOM0_OUT(1) -
...tm_top1.atom0.atom0_ch2.SEL_CMU_CLK_EN - 15000
....i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(2) - 15000
SystemC.ATOM0_OUT(2) -
...tm_top1.atom0.atom0_ch3.SEL_CMU_CLK_EN - 20000
....i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(3) - 20000
SystemC.ATOM0_OUT(3) -
...tm_top1.atom0.atom0_ch4.SEL_CMU_CLK_EN - 25000
....i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(4) - 25000
SystemC.ATOM0_OUT(4) -
...tm_top1.atom0.atom0_ch5.SEL_CMU_CLK_EN - 30000
....i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(5) - 30000
SystemC.ATOM0_OUT(5) -
...tm_top1.atom0.atom0_ch6.SEL_CMU_CLK_EN - 35000
....i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(6) - 35000
SystemC.ATOM0_OUT(6) -
...tm_top1.atom0.atom0_ch7.SEL_CMU_CLK_EN - 50000
....i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(7) - 50000
SystemC.ATOM0_OUT(7) -

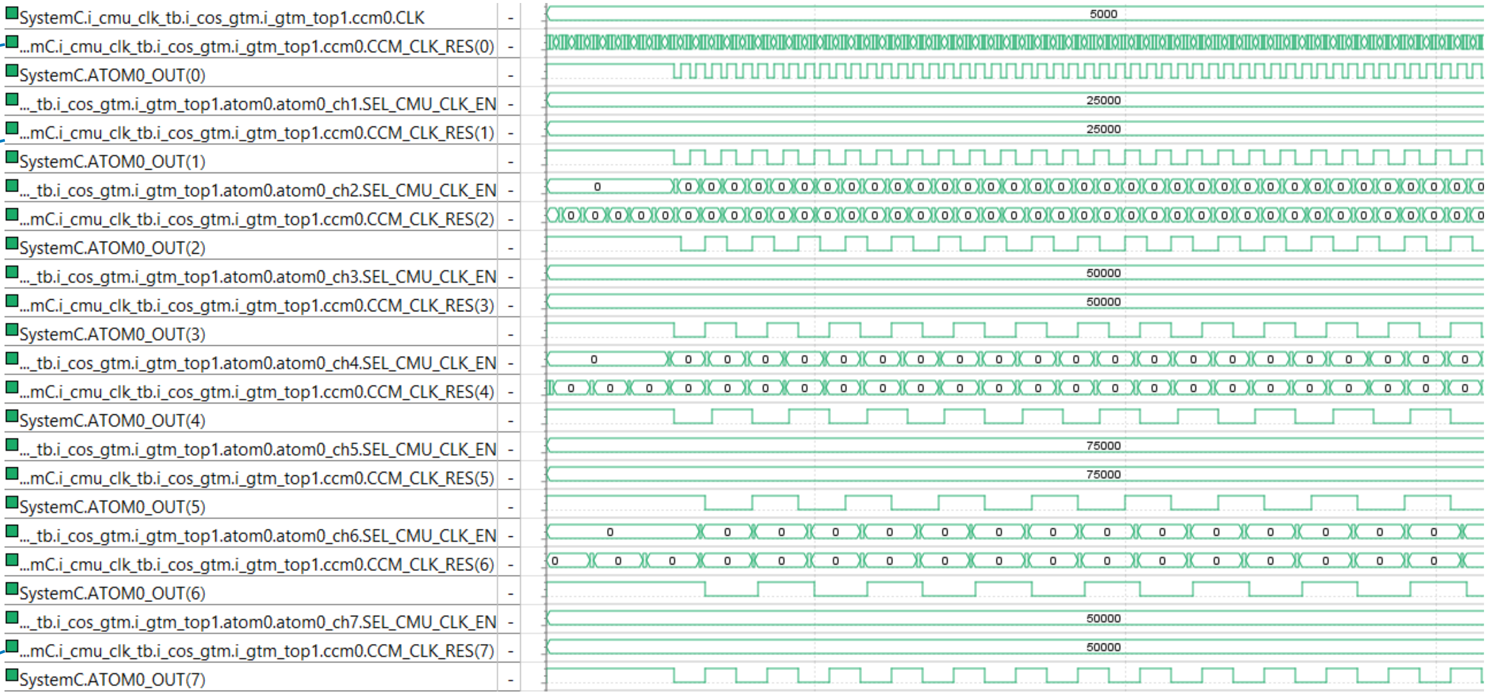
Testcases clarification
Waveform

- LAB1+ CMU_TEST0
  - Cluster clock is 200 MHz (~ 5 ns)
  - CMU_GCLK_EN is 80 MHz

ATOM_CH0 average clk is 80 MHz

ATOM_CH1 clk is 40 MHz

ATOM_CH7 clk is 20 MHz



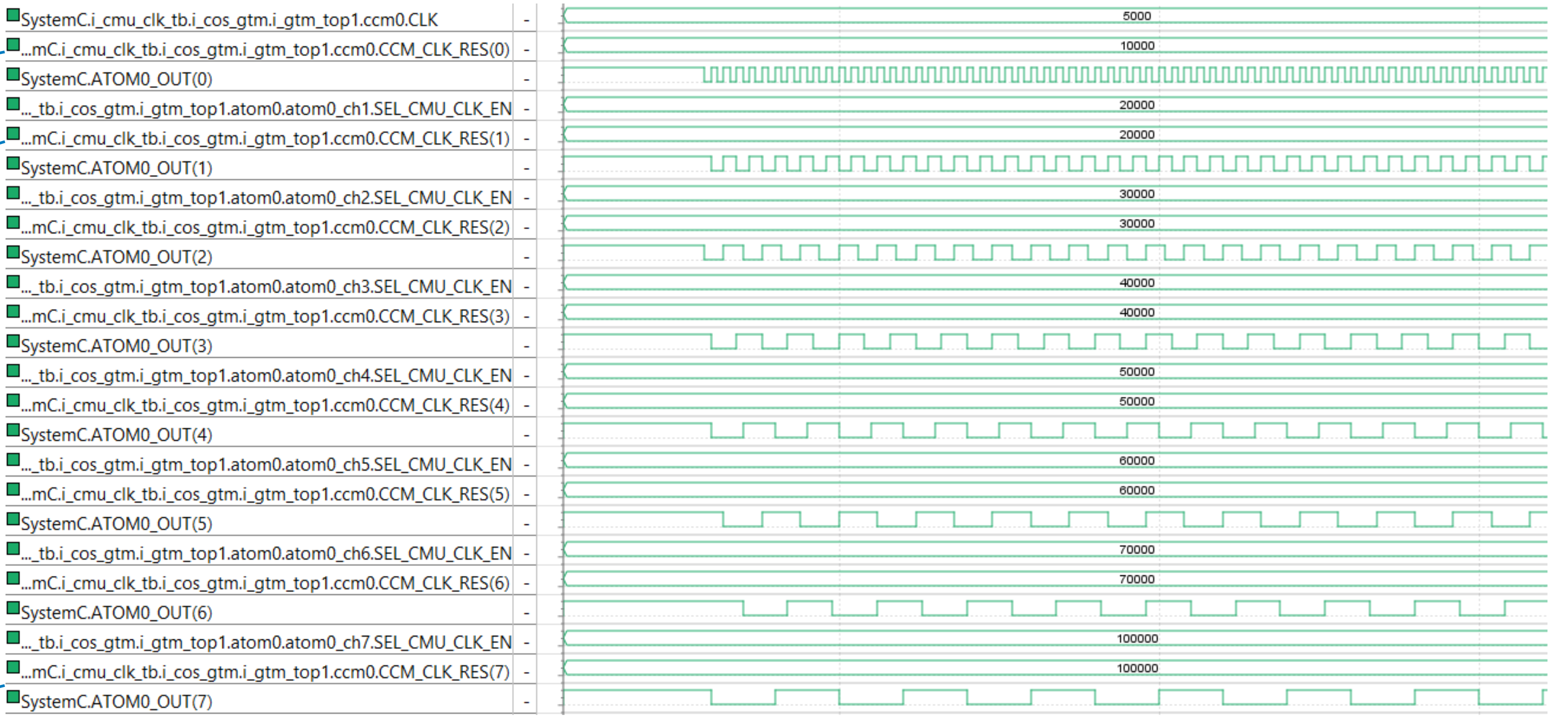
```
SystemC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CLK                    -   5000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(0)           -
SystemC.ATOM0_OUT(0)                                                  -
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch1.SEL_CMU_CLK_EN            -   25000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(1)           -   25000
SystemC.ATOM0_OUT(1)                                                  -
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch2.SEL_CMU_CLK_EN            -   0
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(2)           -   0
SystemC.ATOM0_OUT(2)                                                  -
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch3.SEL_CMU_CLK_EN            -   50000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(3)           -   50000
SystemC.ATOM0_OUT(3)                                                  -
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch4.SEL_CMU_CLK_EN            -   0
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(4)           -   0
SystemC.ATOM0_OUT(4)                                                  -
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch5.SEL_CMU_CLK_EN            -   75000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(5)           -   75000
SystemC.ATOM0_OUT(5)                                                  -
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch6.SEL_CMU_CLK_EN            -   0
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(6)           -   0
SystemC.ATOM0_OUT(6)                                                  -
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch7.SEL_CMU_CLK_EN            -   50000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(7)           -   50000
SystemC.ATOM0_OUT(7)                                                  -
```

Testcases clarification
Waveform

- LAB2+ CMU_TEST0
  - Cluster clock is 100 MHz (~ 10 ns)
  - CMU_GCLK_EN is 100 MHz

ATOM_CH0 clk is 100 MHz ~10 ns

ATOM_CH1 clk is 50 MHz ~ 20 ns

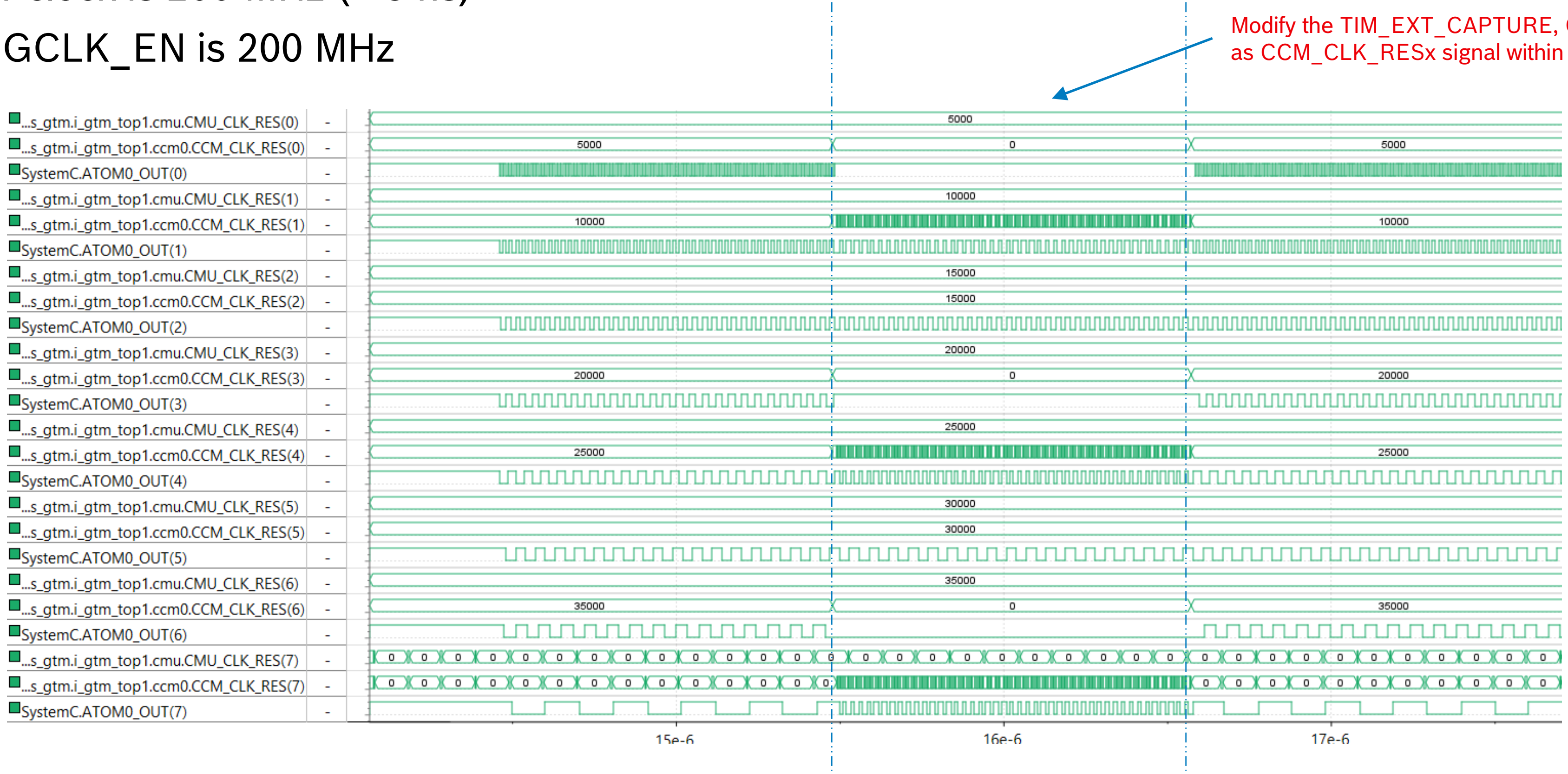

SystemC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CLK
5000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(0)
10000
SystemC.ATOM0_OUT(0)
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch1.SEL_CMU_CLK_EN
20000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(1)
20000
SystemC.ATOM0_OUT(1)
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch2.SEL_CMU_CLK_EN
30000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(2)
30000
SystemC.ATOM0_OUT(2)
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch3.SEL_CMU_CLK_EN
40000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(3)
40000
SystemC.ATOM0_OUT(3)
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch4.SEL_CMU_CLK_EN
50000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(4)
50000
SystemC.ATOM0_OUT(4)
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch5.SEL_CMU_CLK_EN
60000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(5)
60000
SystemC.ATOM0_OUT(5)
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch6.SEL_CMU_CLK_EN
70000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(6)
70000
SystemC.ATOM0_OUT(6)
..._tb.i_cos_gtm.i_gtm_top1.atom0.atom0_ch7.SEL_CMU_CLK_EN
100000
...mC.i_cmu_clk_tb.i_cos_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(7)
100000
SystemC.ATOM0_OUT(7)

ATOM_CH7 clk is 10 MHz ~100 ns

Testcases clarification
Waveform

- LAB0 + CMU_TEST1
  - Cluster clock is 200 MHz (~ 5 ns)
  - CMU_GCLK_EN is 200 MHz

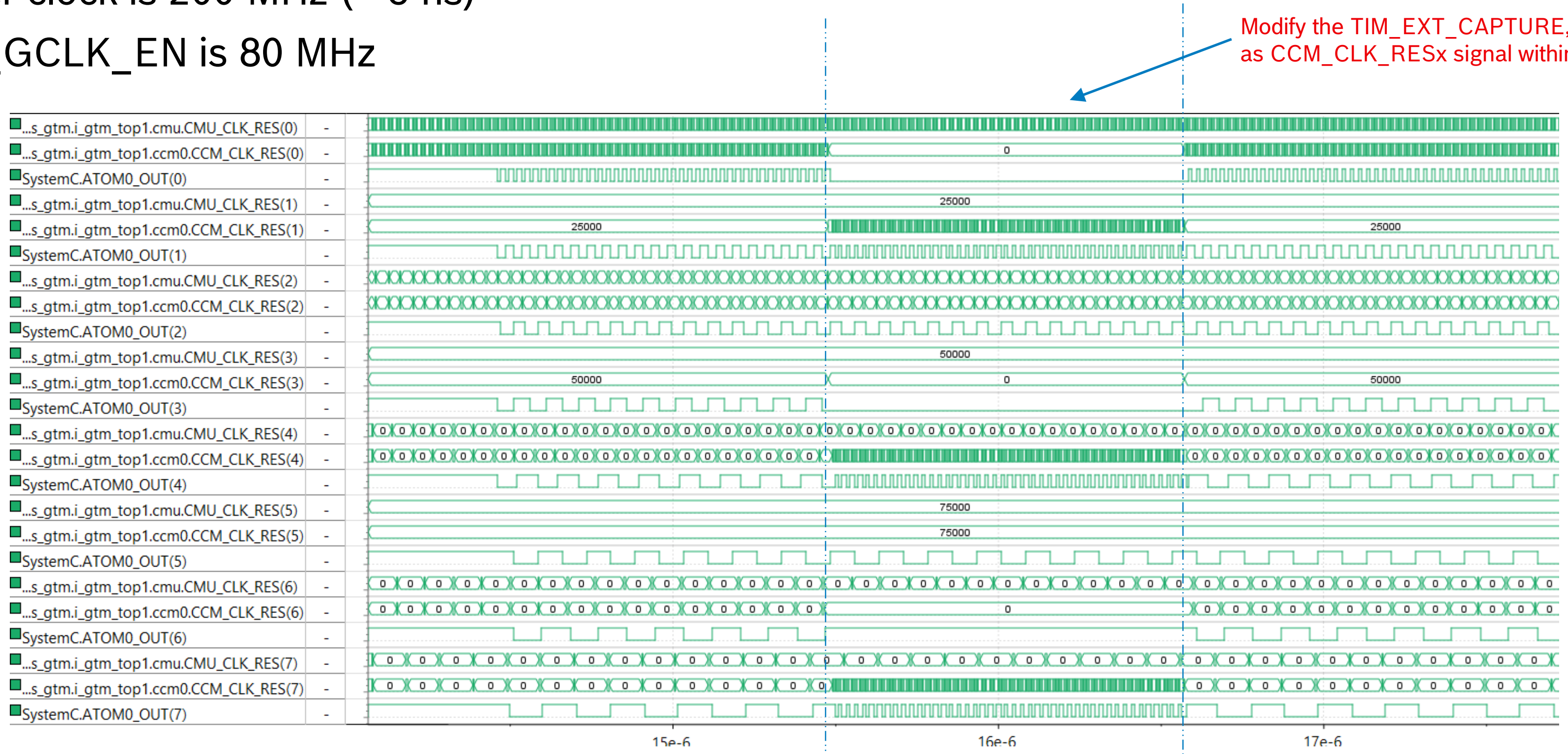
Modify the TIM_EXT_CAPTURE, CMU_CLK_RES[8:8] as CCM_CLK_RESx signal within the cluster



Testcases clarification
Waveform

- LAB1 + CMU_TEST1
  - Cluster clock is 200 MHz (~ 5 ns)
  - CMU_GCLK_EN is 80 MHz

Modify the TIM_EXT_CAPTURE, CMU_CLK_RES[8:8] as CCM_CLK_RESx signal within the cluster

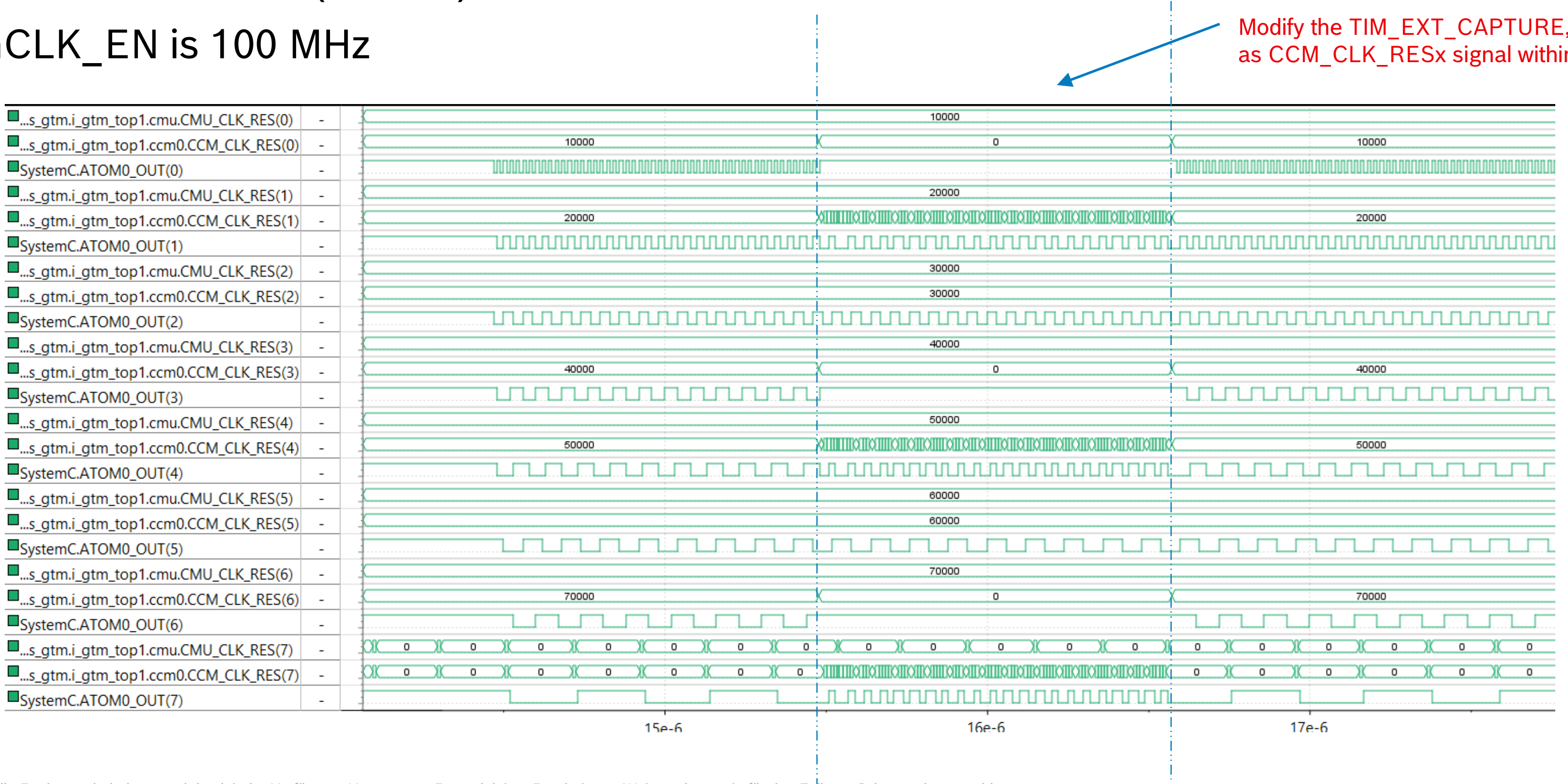


...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(0)
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(0)
SystemC.ATOM0_OUT(0)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(1)
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(1)
SystemC.ATOM0_OUT(1)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(2)
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(2)
SystemC.ATOM0_OUT(2)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(3)
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(3)
SystemC.ATOM0_OUT(3)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(4)
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(4)
SystemC.ATOM0_OUT(4)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(5)
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(5)
SystemC.ATOM0_OUT(5)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(6)
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(6)
SystemC.ATOM0_OUT(6)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(7)
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(7)
SystemC.ATOM0_OUT(7)
25000
50000
75000
0
15e-6
16e-6
17e-6

Testcases clarification
Waveform

- LAB2+ CMU_TEST1
  - Cluster clock is 100 MHz (~ 10 ns)
  - CMU_GCLK_EN is 100 MHz

Modify the TIM_EXT_CAPTURE, CMU_CLK_RES[8:8] as CCM_CLK_RESx signal within the cluster


...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(0) 10000
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(0) 10000 0 10000
SystemC.ATOM0_OUT(0)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(1) 20000
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(1) 20000 20000
SystemC.ATOM0_OUT(1)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(2) 30000
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(2) 30000
SystemC.ATOM0_OUT(2)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(3) 40000
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(3) 40000 0 40000
SystemC.ATOM0_OUT(3)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(4) 50000
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(4) 50000 50000
SystemC.ATOM0_OUT(4)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(5) 60000
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(5) 60000
SystemC.ATOM0_OUT(5)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(6) 70000
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(6) 70000 0 70000
SystemC.ATOM0_OUT(6)
...s_gtm.i_gtm_top1.cmu.CMU_CLK_RES(7) 0
...s_gtm.i_gtm_top1.ccm0.CCM_CLK_RES(7) 0
SystemC.ATOM0_OUT(7)
15e-6
16e-6
17e-6