Memory Compiler User Guide

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Comment | Author |
| 2016/10/28 | v1.0 | Initial version | Yi Li |
| 2016/11/03 | v1.1 | 1) signoff ssg -> ssgwc  2) adding naming rule  3) adding sim rtl rule  4) adding tips about 2-port RF GEN2 | Yi Li |
| 2017/1/14 | v1.2 | 1. Adding new naming rule 2. Adding memory IP path 3. Adding memory choosing 4. Adding xlsx sample path 5. Adding xlsx2ish scripts 6. Adding special pins guide | Song Pan  Yi Li |

**Notes:**

1. ***Memory Compiler can only run on eda-11 machine now.***
2. ***PVT corner,***

worst case: ssgwc0p8v1vn40c

best case: ffg0p99v125c

power case: ff0p99v125c

1. ***integrator using guide***

eda-11:/eda/synopsys/integrator/doc/integrator.pdf

1. Step1: Environment Setup

Before using memory compiler, please make sure you have set the following command (license, software and mem library) in bash.

export LM\_LICENSE\_FILE=”27000@eda-11:$LM\_LICENSE\_FILE”

export PATH=”/eda/synopsys/integrator/bin:$PATH”

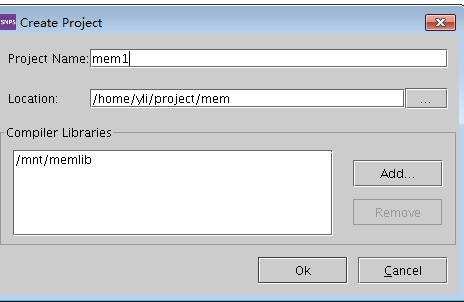
export VL\_COMPLIB\_PATH=”/mnt/memlib”

1. Step2: launch integrator

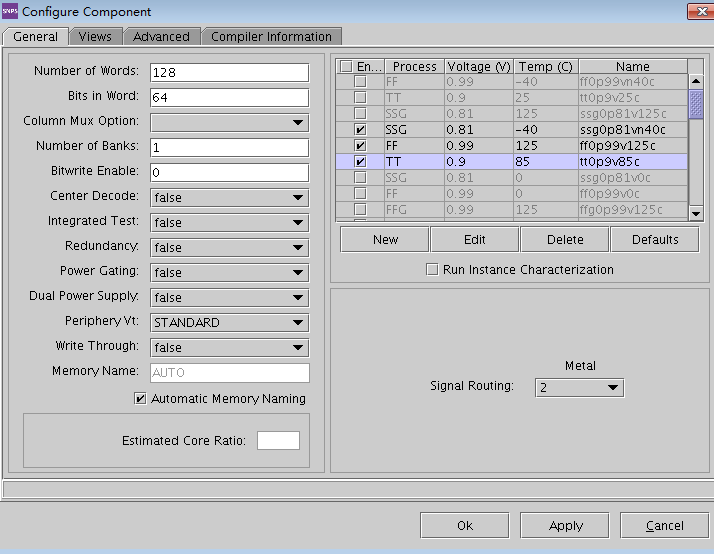
./integrator

1. Step3: Create library

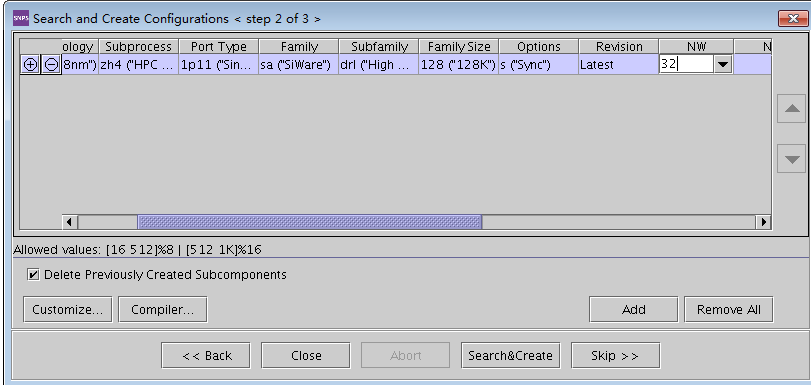
Project -> New



1. Step4: Create Component
2. Component->New (Single Memory Generation Mode)



1. Component->Group Wizard (Group Memory Generation Mode)



1. ***Naming Rule***

(ModuleName)\_(TYPE)\_(PORT)\_(DEPTH)X(WIDTH)(WriteEnable)(PowerGating)(Redundancy)

e.g. A7\_SR\_1P\_8192X64R1

Module Name :

|  |  |
| --- | --- |
| **IP** | **module name** |
| CEVA | CEVA |
| Cortexm7\_CS | M7 |
| MIPI | MIPI |
| PCIE | PCIE |
| CF50 | CF50 |
| Display | DE |
| H264 | H264 |
| HEVC | HEVC |
| JEPG | JEPG |
| typc\_subsystem | TYPC |
| HDMI | HDMI |
| usb3.0\_subsystem | USB |
| security\_subsystem | SEC |
| video\_if | VIF |
| boot\_rom | BOOT |
| cortexa7 | A7 |
| DMAC | DMAC |
| GMAC | GMAC |
| smmu | SMMU |
| SRAM | SRTP |
| BaseBand | BB |
| ISP | ISP |
| EMMC | EMMC |

TYPE: RF/SR/ROM

PORT: 1P/2P/DP

WIDTH: memory bits width

DEPTH: memory addr depth

WriteEnable: True is B1, False is none

PowerGating: True is PG, False is none

Redundancy: True is R1, False is none

Pipeline: True is 1P, False is none

1. ***Memory choosing***

Synopsys request us only use GEN2 memory, so that

1) We should use *Single Port High Density Gen2 Leakage Control Register File 128K* . (Single Port High Density Leakage Control Register File 128K is forbidden)

2) We should use *Single Port Ultra High Density Gen2 Leakage SRAM* (Single Port Ultra High Density Leakage SRAM 2M is forbidden)

1. ***Verilog model***

Synopsys memory compiler generates several corners at different PVT. And every corner has its own independent ip verilog.  Actually, it's no function difference between these Verilog but only timing parameters.  Thus, we recommend each designer to use **ssgwc0p81vn40c**/${mem\_name}.v to do front-end simulation.

1. ***2-ports Memory***

HD RF GEN2 2-ports 128K has two asynchronous clock for write and read separately.

1. ***Memory IP path***

/projects/sirius/library/mem\_ip/

notes: db files is located at each ip root directory

/projects/sirius/library/mem\_ip/$(ip\_name)/ssgwc0p81vn40c/

/projects/sirius/library/mem\_ip/$(ip\_name)/ffg0p99v125c/

1. ***Translation from Xlsx table to memory database***

Xlsx sample path,

/projects/sirius/library/mem\_ip/sample

Translation step,

Step1: generate ish files only @220;221;222 server(caused by python version)

./csv2ish\_sample.py sample.csv

Step2: generate memory data only @241 server(caused by Memory Compiler)

./sample.ish ${mem\_project}

1. ***Special Pins***
2. **TEST1**

TEST1: TEST enable signal. This signal is for DFT team. Please tie TEST1 to 1'b0.

1. **RM/RME**

**RME:**  Read Margin Enable. This signal control RM[2:0] whether select external input or internal default value.

**RM[3:0]:** Read Margin signal. RM[3:2] are synopsys test mode option. RM[1:0] are performance/yield option. 00:VDDMIN, 01:SLOW, 10:DEFAULT, 11:FAST.

There are 2 connection modes for different application.

**Case 1** : Some design will run faster than its signoff frequency. (e.g. A7 signoff frequency 1.2GHz but will run at 2GHz in some case. ) For those design, we strongly suggest the following connection relationship table,

|  |  |
| --- | --- |
| RME | Register\* (default: 1'b0) |
| RM[3] | fixed value: 1'b0 |
| RM[2] | fixed value: 1'b0 |
| RM[1] | Register (default: 1'b1) |
| RM[0] | Register (default: 1'b0) |

Register\*: These pins will be tied to global registers. Generally, we can classify memory IPs into small/medium/large memory or write/read. One type memory could be connected one set of registers. For these register paths, we usually use constraints such as "set case 1/0 " or "set\_multiple\_cycle " in sdc file.

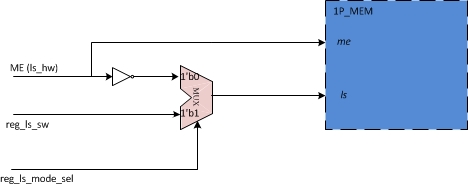
**Case 2:** Some design will only run at its signoff frequency. For those design, we suggest the following connection relationship table,

|  |  |
| --- | --- |
| RME | fixed value: 1'b0 |
| RM[3] | fixed value: 1'b0 |
| RM[2] | fixed value: 1'b0 |
| RM[1] | fixed value: 1'b1 |
| RM[0] | fixed value: 1'b0 |

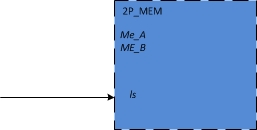
1. **LS (power pins)**

LS control

For 1P-RAM.



For 2P/DP memory, please control LS by register only.



1. Power/margin pin control in Test mode

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **scan\_mode** | **bsd\_mode** | **mbist\_mode** | **ate\_mode** |
| **memory Lightsleep/DeepSleep/Shutdown** | **Y** | **Y** | **N** | **function** |
| **memory RM/RME** | **fixed value as function** | **fixed value as function** | **fixed value as function** | **function** |

The below is example RTL:

Wire sram\_sd = (scan\_mode|bsd\_mode) ? 1’b1 : (mbist\_mode ? 1’b0: reg\_in[0]);

Wire sram\_ds = (scan\_mode|bsd\_mode) ? 1’b1 : (mbist\_mode ? 1’b0: reg\_in[1]);

Wire sram\_ls = (scan\_mode|bsd\_mode) ? 1’b1 : (mbist\_mode ? 1’b0: reg\_in[2]);

Wire sram\_rme = (scan\_mode|bsd\_mode|mbist\_mode) ? 1’b0: reg\_in[3];

Wire sram\_rm = (scan\_mode|bsd\_mode|mbist\_mode) ? 4’b0010 : reg\_in[7:4];