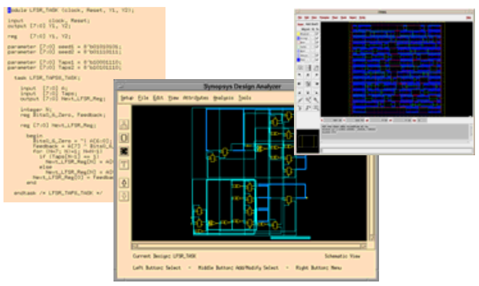
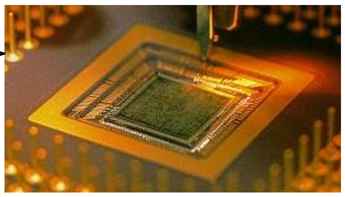


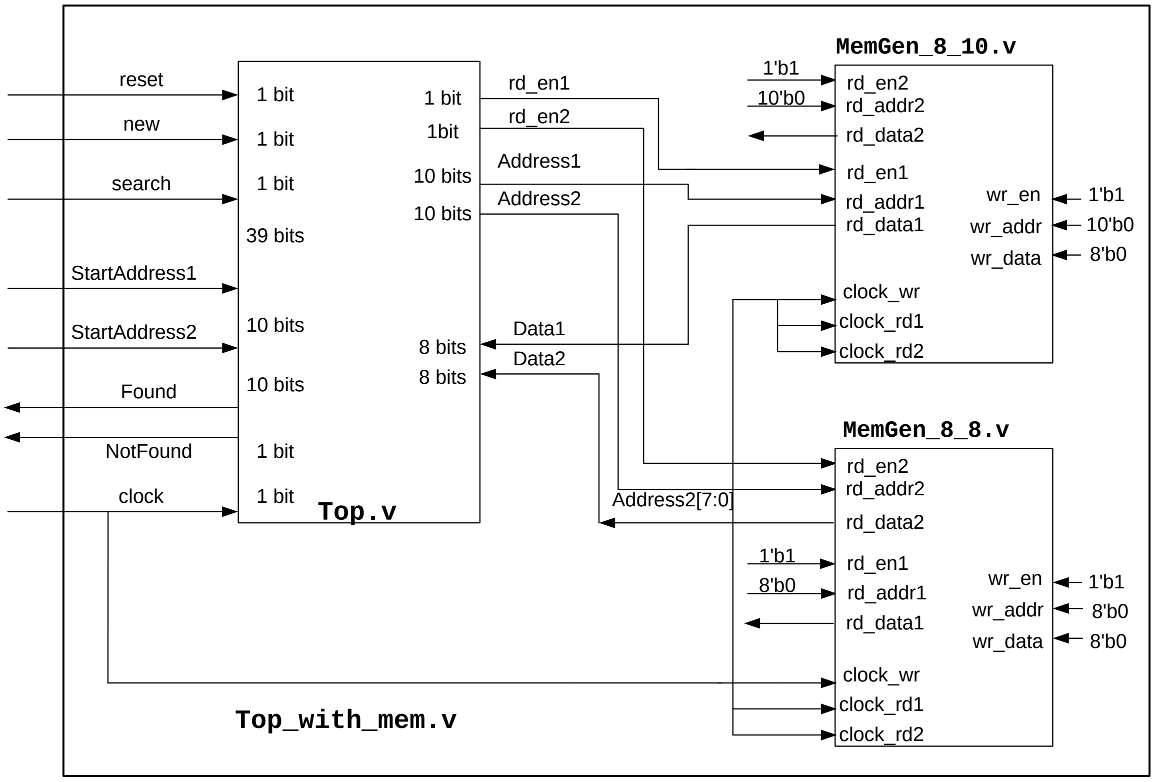


ECE 464 / ECE 520
ASIC Design

Determining Power with Multiple Memories

Tutorial 3 had provided the required know-how to perform the analysis of a design with the separation of the memories in the system and the non-memory parts of the design. The tutorial dealt with just one memory. Here, we show you how to perform the required analysis when the number of memories is increased.

Design introduction:
To understand the requirements of analyzing with two memories, we modify the top level design from Tutorial 3 to the form shown below.



This system performs reads from two different memories: one of depth **1024** (**`MemGen_8_10`**)and the other of depth **256**(**`MemGen_8_8`**). Note that one read port for each memory has been disabled. All the files remain unchanged from Tutorial 3 but for the top level integration file which we call `top_with_2mem.v`. This can be downloaded from :

This gives us the HDL files: `Controller.v` `Engine.v` `top.v` `top_with_2mem.v`. These go into the `HDL/run_s/` folder. We are going to assume that you have already set up the directory structure and all the relevant files as shown in Tutorial 3. Given this, we are now going to enumerate only the salient points on the analysis with two memories. The first step is to create the new memory model by running the following in the `MEMORY` folder by using

```
./CreateModel.pl -cacti_path ../CACTI/cacti_4_1_rel/cacti -width 8
-depth 256
```

This gives us the `MemGen_8_8.lib, MemGen_8_8.v, MemGen_8_8_RTL.v and MemGen_8_8.db` files.

Simulation and Verification
The simulation and verification shown in Section 9 of tutorial 2 remains unchanged except for the additional compilation of the new memory file. The compilation now becomes:

```
   > add modelsim63
   > setenv MODELSIM modelsim.ini
   > vlib mti_lib
   > vlog ../../HDL/run_s/Engine.v
   > vlog ../../HDL/run_s/Controller.v
   > vlog ../../HDL/run_s/top.v
   > vlog ../../HDL/run_s/top_with_2mem.v
   > vlog ../../MEMORY/MemGen_8_8.v
   > vlog ../../MEMORY/MemGen_8_10.v
   > vlog ./test.v
```

To simulate we would then do:

```
> vsim –novopt test_search &
```

Synthesis of non-memory part of design:
Given that the non-memory portion of the design remains unchanged, section 10 dealing with synthesis remains completely unchanged.

SPEF Creation of non-memory part of design:
Section 10.a. dealing with SPEF creation remains completely unchanged.

SAIF Creation of non-memory part of design:
The testbench remains unchanged. The only difference is that you now need to compile the new memory Verilog file and the new top level integration file. Therefore, the compilation becomes:

```
vlog ../../PR/run_f/top_routed.v
vlog ../../MEMORY/MemGen_8_10.v
vlog ../../MEMORY/MemGen_8_8.v
vlog ../../HDL/run_s/top_with_2mem.v
vlog /afs/bp.ncsu.edu/dist/cadence_cdk/OSU018_StdCells/Typical/osu018_stdcells.v
vlog ./test_switching.v
```

and SAIF creation requires a simulation of the form shown below to create the `top_mem_back.saif` backward SAIF file

```
vsim -c -novopt test_search -pli $SYNOPSYS/amd64/power/vpower/libvpower.so
```

Creating power reports for the Design without memory.

This remains unchanged from Section 10.c of Tutorial 3. Please be sure to confirm that the results do not change much from the results from Tutorial 3 i.e. just one memory. Note that, till this point, we are only interested in working with the non-memory parts of the design which have not changed at all.

Creating power reports for the Design WITH memory.

It is in this section that we are going to have to make some drastic modifications for correctness. As stated in Tutorial 3, please re-run the SAIF creation at a minimized clock with a valid slack. Let us use 6ns here. Note that, if needed, you can later manually scale the result by removing the slack still available after the memories are included. For a 6ns clock period, the non-memory portion of the design would consume 11.08 mW as seen in Tutorial 3.

Now, let us use the same execution command for memory analysis as used in Tutorial 3. This gives us

```
./PAD_Flow.pl -saif ./SIMULATION/run_f/top_mem_back.saif -op memory
-period 6 -inst test_search/top_mem -mod top -clkname clock -memdir
./MEMORY/  -topinst  top_inst  -topfile  ./HDL/run_s/top_with_2mem.v
   -memname MemGen_8_10
```

The above results in the creation of the `./SYNTH/run_f/memory_analysis.tcl` file which is run within design compiler by the `PAD_Flow.pl` to give you a power result. The `./SYNTH/run_f/memanalysis_transcript.out` file provides progress reports for this operation. A peek into this `.out` file will show you errors/warnings with regards to the presence of linking errors because there are unresolved references. This is because, at this point, the `MemGen_8_8` design has not been linked. Moreover, you will also get errors related to the lack of annotation to the `data2` and `Address2` signals not being annotated. This is because of the above un-resolved references given that these signals are connected to the missing `MemGen_8_8` memory.

You will get the following result even with these un-resolved references. The `./SYNTH/run_f/top_pwr_post_annotation_mem.rpt` will have

```
      Cell Internal Power  =   9.2100 mW   (62%)
      Net Switching Power  =   5.5891 mW   (38%)
                              ---------
      Total Dynamic Power    =  14.7991 mW  (100%)
```

This is far lesser than the 18mW you got in Tutorial 3 given that we only have one port active at present. Also, the contributions from MemGen_8_8 have not yet been considered.

To correct this problem, you would need to manually edit the `memory_analysis.tcl` in `./SYNTH/run_f/` file to add MemGen_8_8.db to the list of link libraries by adding another "`set link_library`" to the list at the top of the file to point to the file. This would change the lines at the top from

```
set search_path [concat $search_path \
/local/home/rsjenkal/DrFranzonWork/Example/2Mem_Example/./MEMORY/]
set link_library [concat $link_library MemGen_8_10.db]
read_verilog -rtl \
/local/home/rsjenkal/DrFranzonWork/Example/2Mem_Example/PR/run_f/top_routed.v
```

to the following

```
set search_path [concat $search_path \
/local/home/rsjenkal/DrFranzonWork/Example/2Mem_Example/./MEMORY/]
set link_library [concat $link_library MemGen_8_10.db]
set link_library [concat $link_library MemGen_8_8.db]
read_verilog -rtl \
/local/home/rsjenkal/DrFranzonWork/Example/2Mem_Example/PR/run_f/top_routed.v
```

Within the `./SYNTH/run_f/` folder, now invoke `design_vision` and source the modified `analysis_memory.tcl` file. It is suggested that this be done one line at a time. The `./SYNTH/run_f/top_pwr_post_annotation_mem.rpt` file now modifies to become

```
Cell Internal Power  =  11.1803 mW   (67%)
Net Switching Power  =   5.5440 mW   (33%)
                       ---------
Total Dynamic Power    =  16.7243 mW  (100%)
```

This value is the final result with the contributions from the two memories and the non-memory portion of the design. The reason this is lower than that **18.67 mW** value from Tutorial 3 is the use of just one port in each memory and the fact that one of the memories is small with a depth of 256 versus 1024.

Any additional memories can be added for analysis using some more "`set link_library`" commands in the `analysis_memory.tcl` file. Note that this method would eventually be fixed.