# UC Berkeley Simulated SRAM Compiler

CS250 Tutorial 9 (Version 120509a)
December 5, 2009
Kyle Wecker & Yunsup Lee

## How to Run

This program generates memories the Verilog, lef, and lib files necessary to simulate and synthesize one and two port SRAM memories. It also automatically converts the lib file to a db file and the lef file to a Milkyway database. Caution must be used because file conflicts will lead to both silent overwrites and runtime errors. Please make sure you are running the program in a safe directory.

It uses Python and must be run directly from the command line using the following command:

```
python memComp.py <Base File Name> <Word Length> <Memory Depth> \
                  <Number of R/W Ports>  <Columns in Memory> \
                  <Operational Conditions> <Debug> <Byte Masking Off>
                  <Overwrite Name>
```

- `Base File Name` - Name that each of the files that are output will begin with.
- `Word Width` - Width of each word in your memory, in bits.
- `Memory Depth` - Number of words the memory contains. If this value is less than 32, rough default values will be used. Depths of less than 32 should only be used for initial functional testing. If your design will require memories with depths less than 32, this is not the appropriate tool to be using.
- `Number of R/W Ports` - Number of read/write ports the memory contains. Currently, only values of one or two are supported.
- `Columns in Memory` - Number equal to the number of bits physically stored in one row of the memory. Can be used to change the aspect ratio. It should be equal to $width * ports * 2^n$, where $n$ is a non-negative integer.
- `Operational Conditions` - Used in the creation of the lib file to set values such as voltage and temperature. Currently, the only acceptable value is "Typical" as the power model does not yet support "BEST" and "WORST".
- `Debug` - Entering "True" will cause the program to run in a more verbose output mode, which in addition to debugging information
- `Byte Masking Off` - Entering "True" will forcefully turn off byte-masking. (By default, byte masking is turned off whenever $width$ %8 != 0.
- `Overwrite Name` - Entering any value here will cause the default file name to be completely overwritten with this value.

The last four options are optional and will default to Typical, False, False, and None, respectively.

**Example 1**

```
python memComp.py SRAM 32 512 2 128
```

This would create in the directory from which the program is run the files/directories: SRAM32x512_2_128.{v,wrap.v,lef,lib,db,mw}. These represent a 32x512 memory with two read/write ports and four words per row. Operating conditions simulate a typical operating environment, no debugging information is printed, and byte masking will be enabled.

**Example 2**

```
python memComp.py SRAM 31 128 1 62 Typical False False MEM
```

This would create in the directory from which the program is run the files/directories: MEM.{v,wrap.v,lef,lib,db,mw}. These represent a 31x128 memory with one read/write port and two words per row. Operating conditions simulate a typical operating environment, no debugging information is printed, and though byte masking is not turned off, it will be disabled because $31 \% 8 != 0$.

**Notes**

- There will be warnings during the running of the program that should be safe to ignore. They are due to the fact that this program only compiles the bare minimum to black-box a SRAM. Most of the information required for a real SRAM model is missing.
- The warnings during synthesis regarding tri-state drivers are also safe to ignore.

## Acknowledgements

The program was originally developed for CS250 VLSI Systems Design course at University of California at Berkeley by Kyle Wecker & Yunsup Lee. Contributors include: Krste Asanović, Andrew Waterman, and John Wawrzynek. Versions of this program have been used in the following courses:

- CS250 VLSI Systems Design (2009) - University of California at Berkeley