Getting DDRs "write" – the 1x output circuit revisited
Addendum 1/20/2007 – using QTM instead of STAMP

Paul Zimmer

Zimmer Design Services
1375 Sun Tree Drive
Roseville, CA 95661

paulzimmer@zimmerdesignservices.com

website: www.zimmerdesignservices.com

.

1 Introduction and Background

At SNUG San Jose 2006, I presented a paper called "Getting DDRS Write". In that paper, I described a technique for timing DDRs using external STAMP models, either by instantiating them in a higher level along with the chip, or stitching them in using create_cell. The technique has a number of advantages over conventional constraint-based techniques.

Less than a year later, Synopsys removed support for STAMP models from PrimeTime (starting with version 2006.12). Sigh..

The basic external model technique is still valid, and still has all the advantages discussed in the paper, but we need a replacement for the STAMP models. So, I have modified the scripts to use QTM models.

2 Creating and hooking in the QTM model.

It turns out that the modifications required to create and hook in a QTM model instead of a STAMP model are fairly small. Instead of the ddr_write.mod and ddr_write.data files and the call to "compile_stamp_model" in the script, use the following code to create an equivalent QTM model:

```
# Create the qtm model and write out the db
create_qtm_model ddr_write
create_qtm_port -type input {DQ_I}
create_qtm_port -type clock {DQS_I}

set_qtm_global_parameter -param setup -value 0.0
set_qtm_global_parameter -param hold -value 0.0

create_qtm_constraint_arc \
  -name Tds_dq_pos_0 \
  -setup \
  -from DQS_I \
  -to DQ_I \
  -edge rise \
  -value 0.5

create_qtm_constraint_arc \
  -name Tdh_dq_pos_0 \
  -hold \
  -from DQS_I \
  -to DQ_I \
  -edge rise \
  -value 0.5

create_qtm_constraint_arc \
  -name Tds_dq_neg_0 \
  -setup \
  -from DQS_I \
  -to DQ_I \
  -edge fall \
  -value 0.5
```

```
create_qtm_constraint_arc \
  -name Tdh_dq_neg_0 \
  -hold \
  -from DQS_I \
  -to DQ_I \
  -edge fall \
  -value 0.5

set_qtm_port_load -value 0.0 {DQ_I DQS_I}

save_qtm_model -format db -output ddr_write
```

3 Changes specific to the hierarchical technique

If you use the technique of instantiating the model and the chip in a wrapper, that's all that's needed in PT version 2006.06. But in 2006.12, for reasons I don't understand, PT adds its own level of hierarchy, called "core", beneath the ddr_write instantiation.

The wrapper instantiation looks like this:

```
ddr_write ddr_write(
  .DQ_I (dq),
  .DQS_I (dqs)
);
```

In 2006.06 and earlier, the set_annotated_delay statements look like this:

```
  set_annotated_delay -net -max $_dq_wire_delay_max \
    -from [get_pins ${_chip}dqpad/Z] \
    -to [get_pins ddr_write/DQ_I]

  set_annotated_delay -net -min $_dq_wire_delay_min \
    -from [get_pins ${_chip}dqpad/Z] \
    -to [get_pins ddr_write/DQ_I]

  set_annotated_delay -net -max $_dqs_wire_delay_max \
    -from [get_pins ${_chip}dqspad/Z] \
    -to [get_pins ddr_write/DQS_I]

  set_annotated_delay -net -min $_dqs_wire_delay_min \
    -from [get_pins ${_chip}dqspad/Z] \
    -to [get_pins ddr_write/DQS_I]
```

This is unchanged from the original paper.

But in 2006.12, you need to add the extra dummy level to the path:

```
set_annotated_delay -net -max $_dq_wire_delay_max \
  -from [get_pins ${_chip}dqpad/Z] \
  -to [get_pins ddr_write/core/DQ_I]

set_annotated_delay -net -min $_dq_wire_delay_min \
  -from [get_pins ${_chip}dqpad/Z] \
  -to [get_pins ddr_write/core/DQ_I]

set_annotated_delay -net -max $_dqs_wire_delay_max \
  -from [get_pins ${_chip}dqspad/Z] \
  -to [get_pins ddr_write/core/DQS_I]

set_annotated_delay -net -min $_dqs_wire_delay_min \
  -from [get_pins ${_chip}dqspad/Z] \
  -to [get_pins ddr_write/core/DQS_I]
```

I have no idea why the tool does this. I tried changing the instance name to "ddr_writei" in the instantiation, but that didn't fix it.

4 Changes specific to the create_cell technique

You might think you'd have the same problem with the create_cell technique in 2006.12. But, no. There's no extra "core" level in the path if you use create_cell instead of a wrapper instantiation. No idea why.

But there is one other change needed for the create_cell script. Again, PT is adding some "core" stuff on its own. It adds it to the name of the model in the library. This time, it does it in all versions. The line to change is the create_cell command itself:

```
create_cell ddr_write ddr_write_lib/ddr_write_core
```

PT has chosen to call the model "ddr_write_core" inside the library instead of "ddr_write".

With those simple changes, the "STAMP" model technique becomes the "QTM" model technique.