

Multi-Cycle Path Tutorial

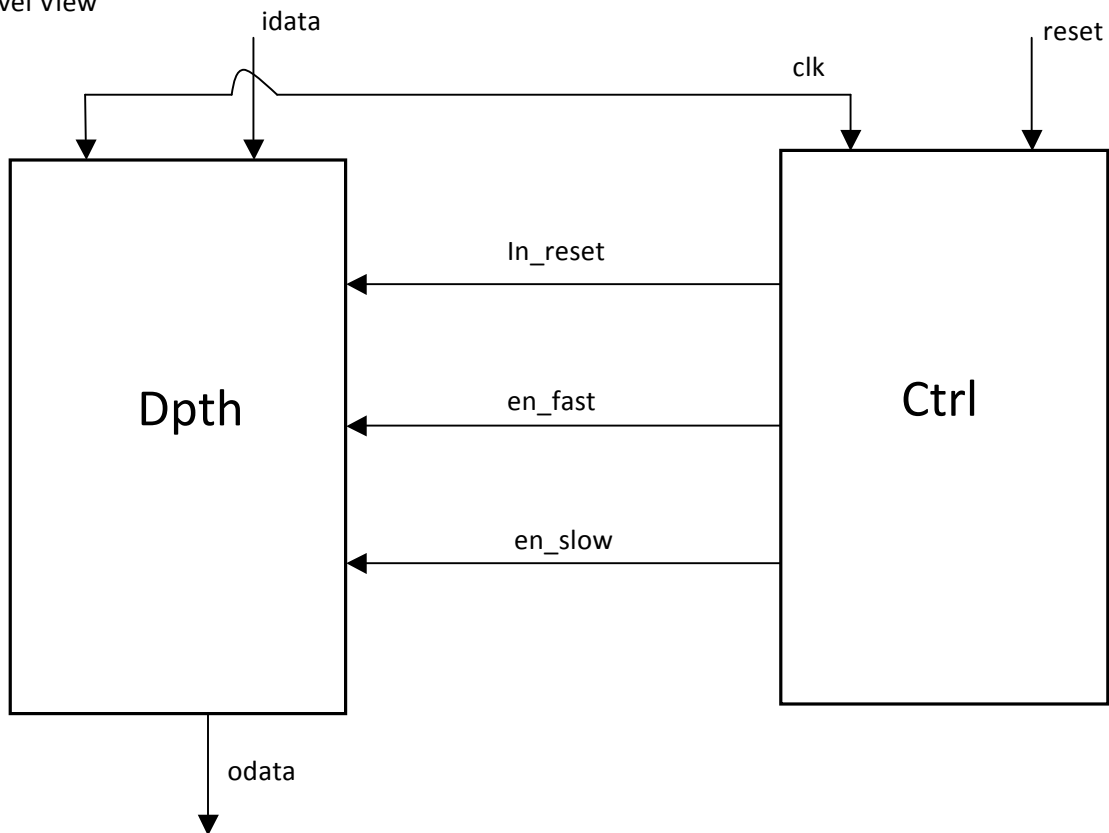
by Xin Xin, ECE Department, GMU

Ver. 1.0 : 04/20/2009

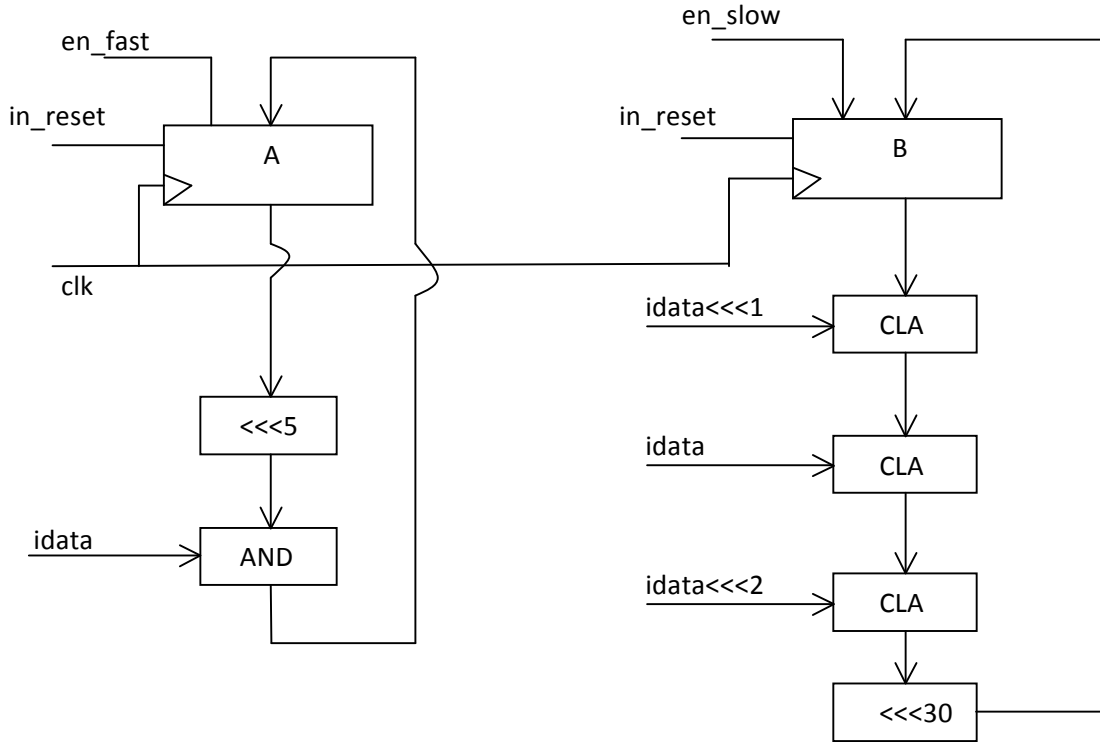
When a combinational path between two registers takes more than one clock cycle to propagate data through, we need to specify special constraints in the UCF file in order to make the synthesis tool aware of the existence of this multi-cycle-path. By default, the synthesis tools will always treat all paths as single cycle paths.

Design Example:

Top-level View



Detailed Data Path View:



En_fast and en_slow are signals from the controller. En_fast is 5 times faster than en_slow. Therefore, a path from the output of register B, through three CLAs and one left rotate logic, back to the input of register B is now a multi-cycle-path, more precisely a 5-cycle-path.

See timing results in the Post Place and Route report file without specifying multi-cycle-path first.

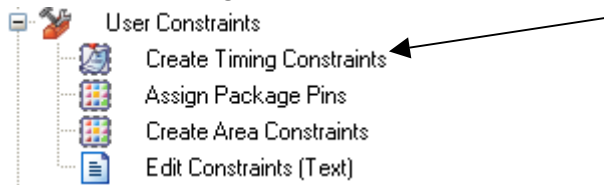
Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
Autotimespec constraint for clock net clk	SETUP	N/A	24.119ns	N/A	0
_BUFGP	HOLD	2.253ns		0	0

(Device: xcv1000-6)

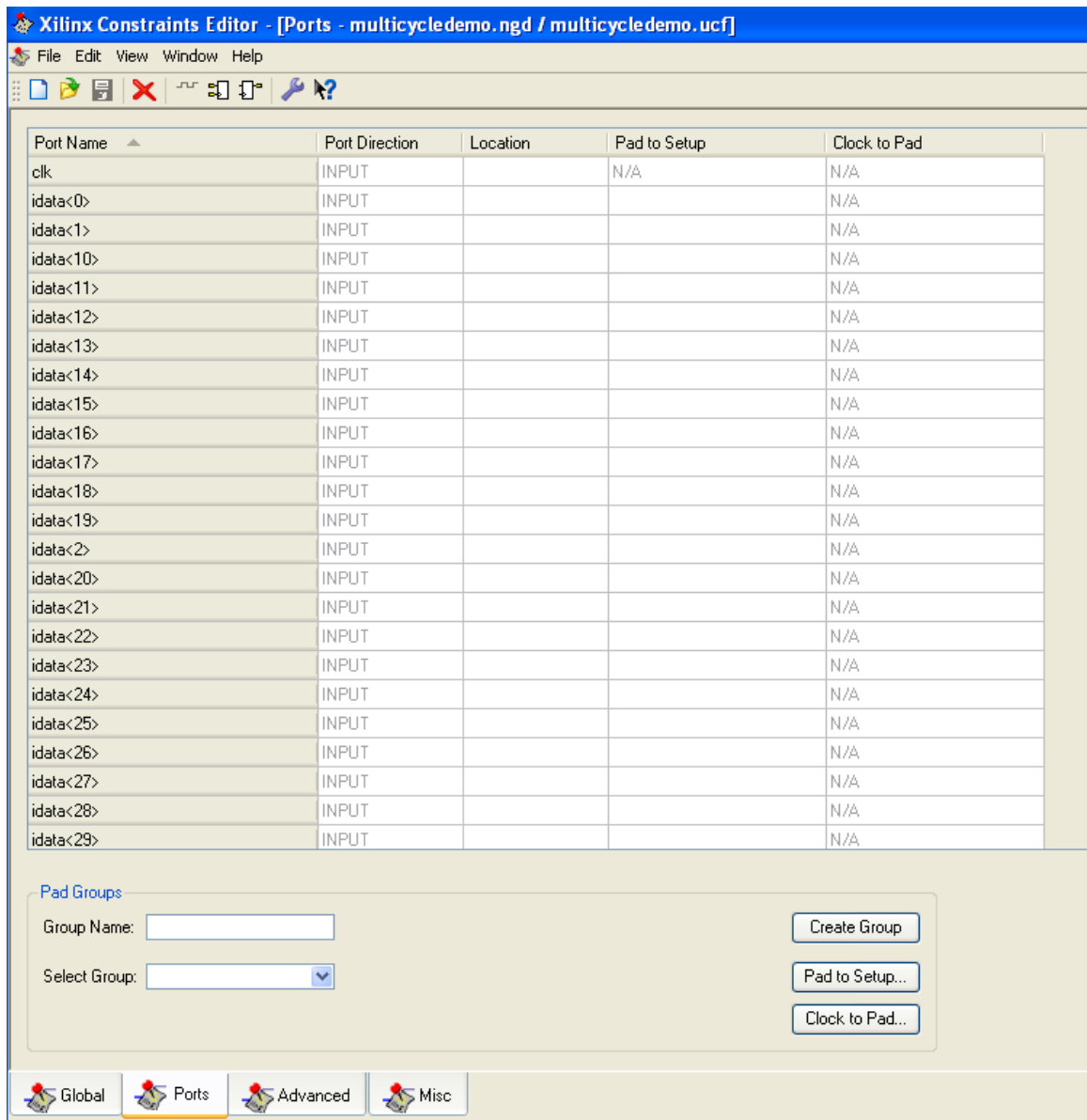
Notice the current critical path delay is 24.119 ns.

Now begin to setup a multi-cycle-path in UCF (using ISE as an example):

1. Click “Create Timing Constraints” to launch Constraints Editor.



2. You should see the following windows:

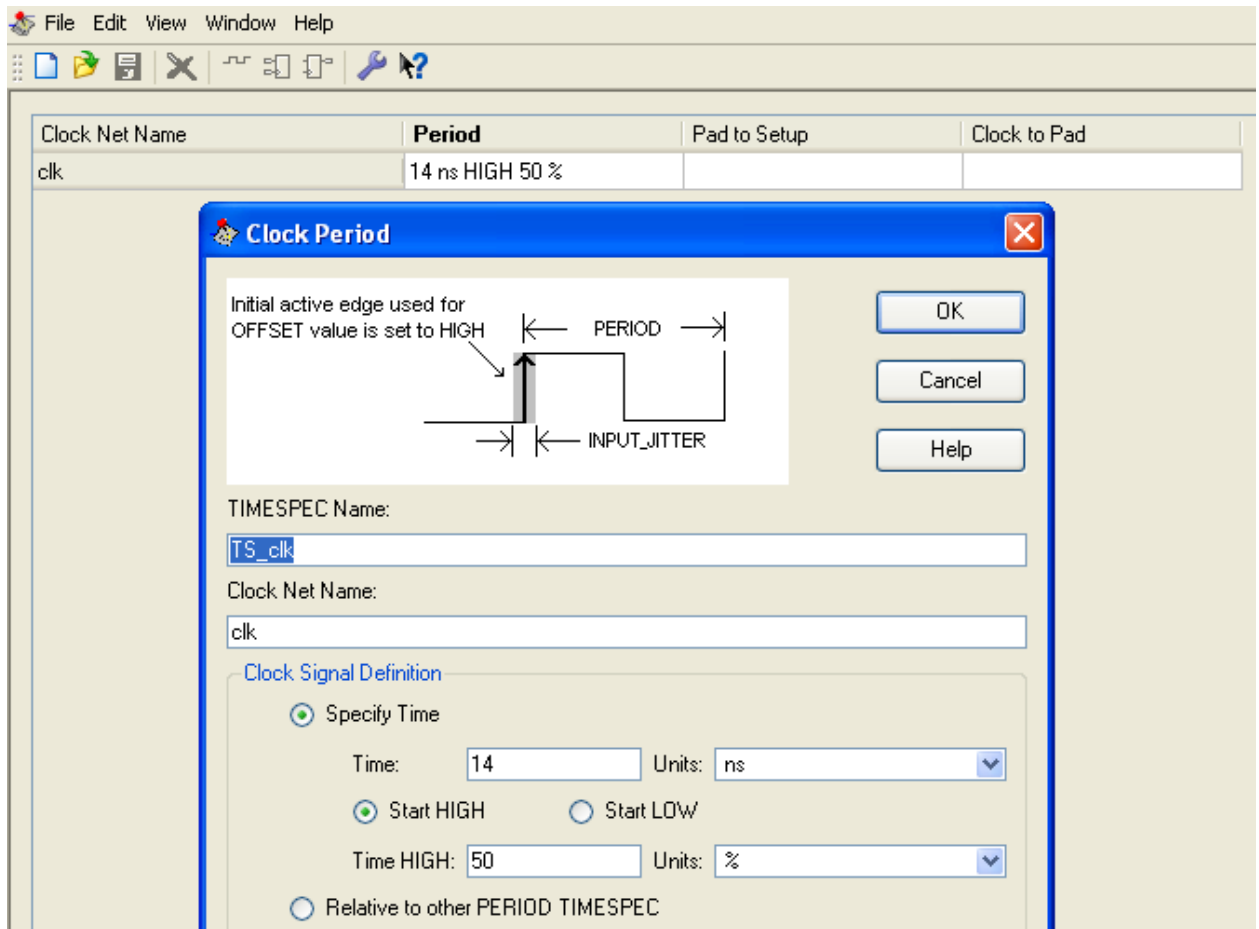


3. Select option “Global”, and then double click “clk” and specify clock period in the new window.

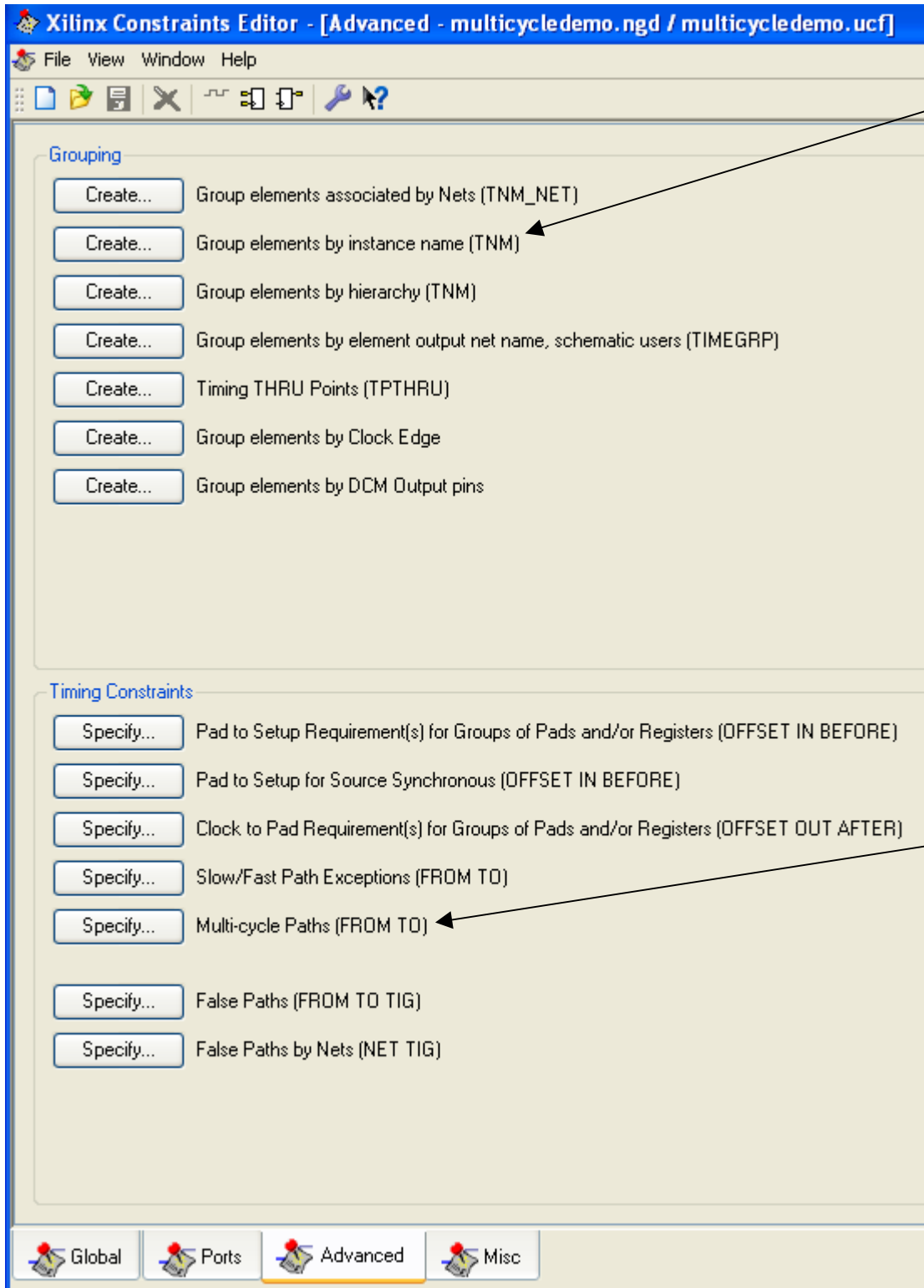
4. The above three steps are equivalent to typing in the following two commands manually in the user constraint file.

```
NET "clk" TNM_NET = "clk";  
TIMESPEC "TS_clk" = PERIOD "clk" 23 ns HIGH 50 %;
```

This actually means grouping all components that connect to input "clk", and naming this group as "clk". Additionally, the time specification, named TS_clk, for the group "clk" is specified as "23 ns HIGH 50%"



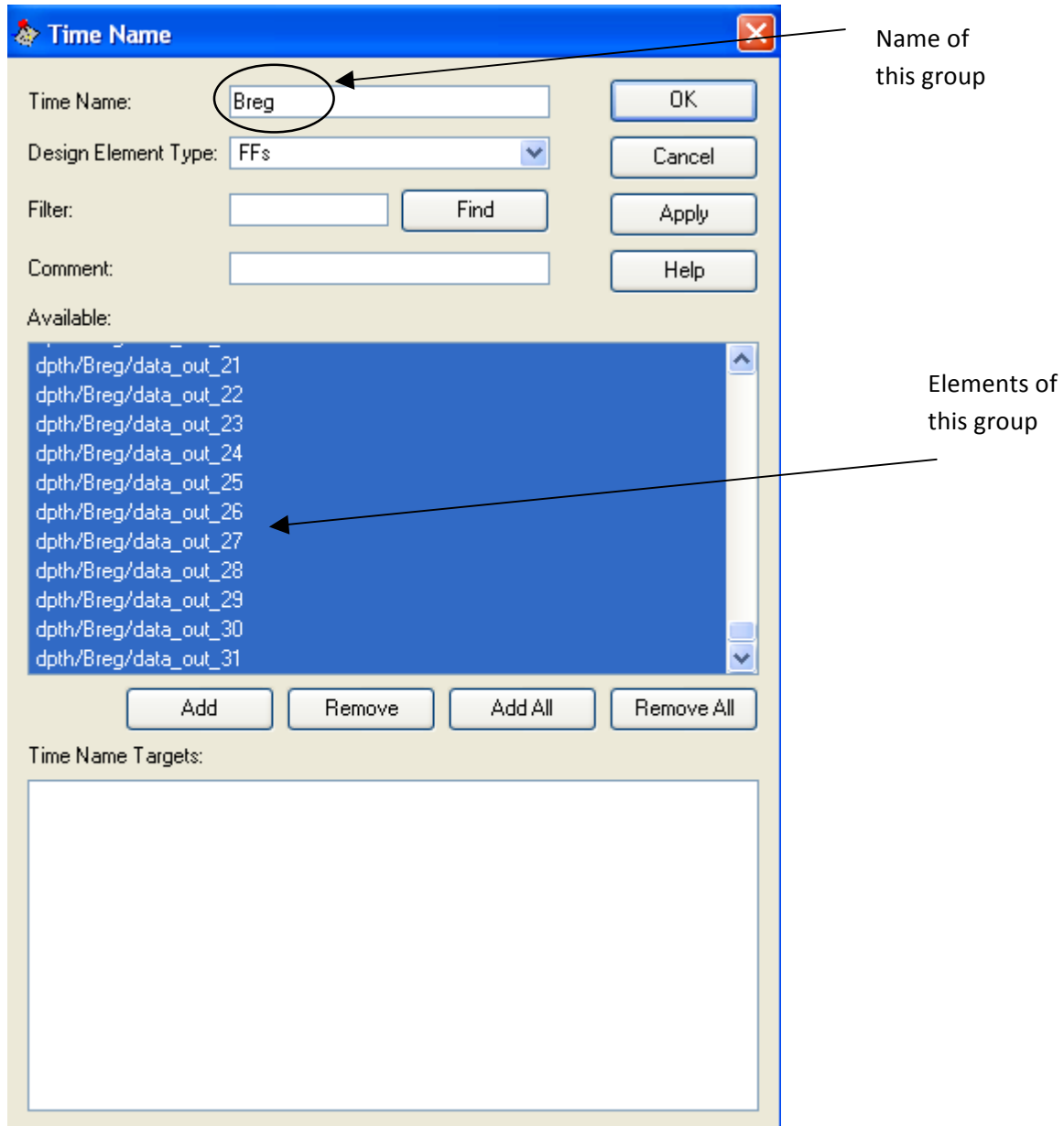
5. Go to option "Advanced"



The first thing we are going to do here is to tell the tool which D flip-flops are a part of the same group. We can click "Group element by instance name (TNM)" to do so, which is shown above. This has the same functionality as manually typing in the following command in the user constraint file:

```
INST "controller/*" TNM = "ctrl";
```

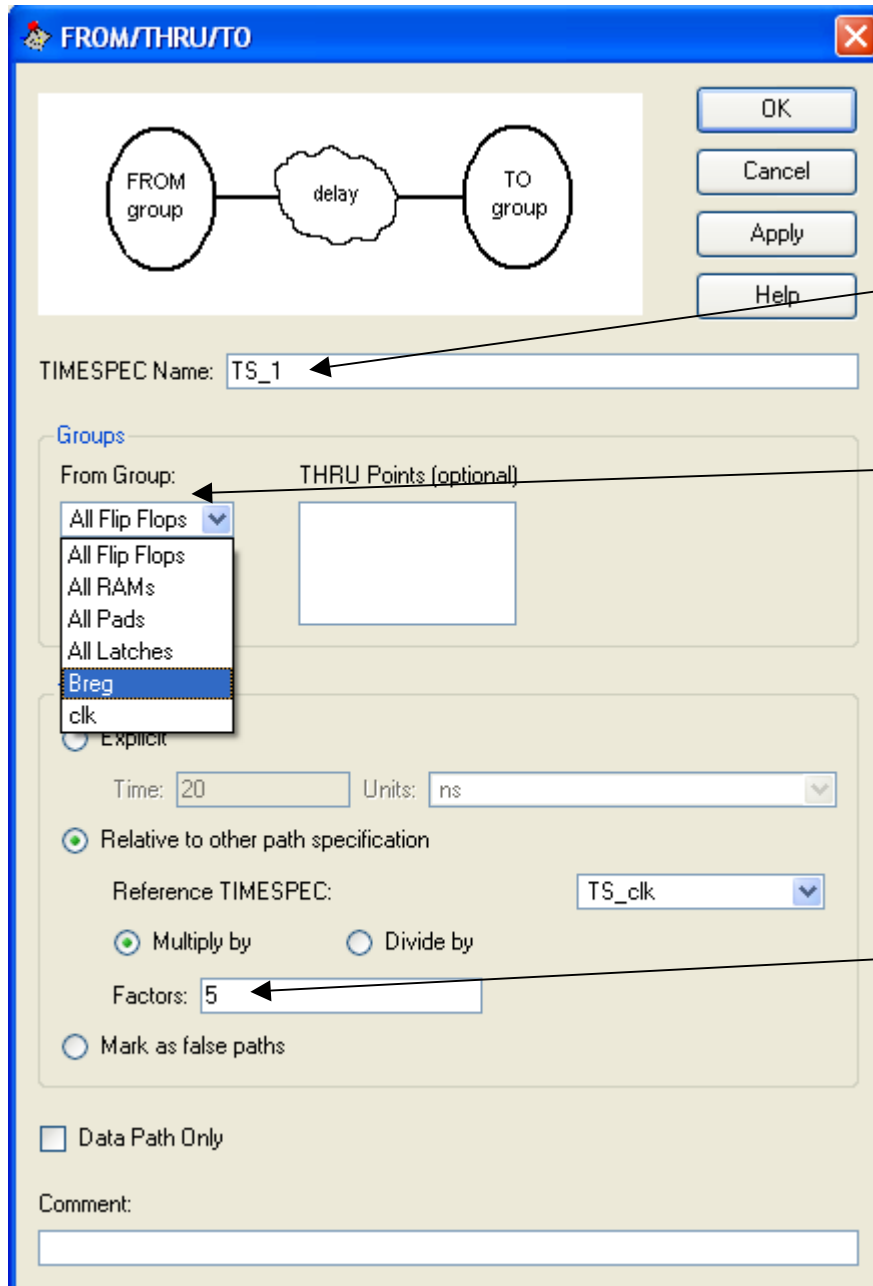
After clicking "Group element by instance name (TNM)", we will see the following window:



Select all the D flip-flops that you want to group together and then click "Add". The last thing one needs to do is to give a name for this group and then click "OK".

Now, we are going to actually specify a multi-cycle-path:

Click “Multi-Cycle Path (From To)”. You should see:



Specify a name for this time specification

Specify which group does the multi-cycle path starts from, using option “From Group”. In this case we select Breg.

Then specify the group that multi-cycle path ends with: “To Group”. In this case we select Breg too.

Specify how many cycles one wants for the multi-cycle path

These tasks conclude all the steps that we need to perform in order to set up a multi-cycle path in UCF.

Run “implementation” again and check out the timing information from the Post Place and Route report file:

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_clk = PERIOD TIMEGRP "clk" 4.8 ns HIGH 50%	SETUP HOLD	0.011ns 2.388ns	4.789ns	0 0	0 0
TS_1 = MAXDELAY FROM TIMEGRP "Breg" TO TIMEGRP "Breg" TS_clk * 5	SETUP	0.234ns	23.766ns	0	0

Notice: After setting up the multi-cycle path, the best achievable clock period decreased to 4.789 ns, which is roughly one fifth of the previous result.

Appendix A: User Constraint File that has been used in this tutorial

```
NET "clk" TNM_NET = "clk";

TIMESPEC "TS_clk" = PERIOD "clk" 4.8 ns HIGH 50 %;

INST "dpth/Breg/*" TNM = "Breg";

TIMESPEC "TS_1" = FROM "Breg" TO "Breg" "TS_clk" * 5;
```

Note: Here is a little trick used here. For grouping all FFs of a certain register, say register B from this tutorial design, one does not have to specify

```
INST "dpth/Breg/data_out_0" TNM = "Breg";
INST "dpth/Breg/data_out_1" TNM = "Breg";
INST "dpth/Breg/data_out_2" TNM = "Breg";
INST "dpth/Breg/data_out_3" TNM = "Breg";
...
...
INST "dpth/Breg/data_out_31" TNM = "Breg";
```

One can actually manually delete all of them and replace their specification by one command: `INST "dpth/Breg/*" TNM = "Breg"`. Both methods are equivalent. The latter version of the UCF file looks cleaner and is less error prone.