

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

We met timing using characterize way also and the slack is positive.

**Click on the “Cancel” button in the timing report window that had popped up.**

**From the main menu bar, click-select the “Reports/Report Area” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

The area report for this synthesis run shows up. Compare this with the area report we saved from time budgeting run of synthesis. The area for time budgeting flow is smaller than the “characterize” way. The characterize way overconstrained the design resulting in larger area than necessary.

Now let’s compare the time it took for the overconstrained design to compile with a minute or two of compile time for Time Budgeted design.

**Click on the “Hdl” tab of schematic viewer. Click “Open” to pop up “Edit HDL file” window.**

**Under “Files” change the filter to “\*.txt”.**

This will show the DerivedContext.txt, a log file for synthesis run. This log file was saved from the synthesis run for design constrained using characterize way.

**Double-click on “DerivedContext.txt”.**

The DerivedContext.txt file will be displayed in the editor window.

Note the time stamp at the start of the file and the end of the file. The difference between them is the time it took for the design to synthesize.

As you can see that overconstrained design due to “characterize” takes almost an order of magnitude time for this case when compared with time budgeting.

**Click the “Close” button in the HDL editor window to close the log file.**

## **Summary**

This demonstration walked you through a time budgeting flow using NaviGates. We compared the results from time budgeting with results using the old way of constraining the designs.

Automatic time budgeting greatly improves the ability of the synthesis tool to compile large designs quickly. Time budgeting results in shorter compile times, smaller resulting area and reduced the number of synthesis iterations. Further, since the time budgeting is done automatically by the tool, the user does not have to manually massage the constraints thus improving ease of use.

In short, Time budgeting improves user productivity.

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

The timing for the worst path in the Budget\_0 is shown in the report. Notice that the path has a negative slack and hence is violating the constraints. Fill in the slack number on the line below.

Budget\_0 slack time \_\_\_\_\_ ns.

Notice that the remaining portion of the negative slack in BudgetTop got applied to the submodule Budget\_0, making the constraints realistic.

**Click on the “Cancel” button in the timing report window that had popped up.**

Thus we have seen that Automatic Time Budgeting creates more realistic constraints for submodules of a design. The user no longer has to manually edit the unrealistic constraints generated by the old method of “characterize”. Time budgeting thus improves the user productivity.

## **Step 5. Synthesize the design bottom-up using Time Budgeted constraints**

**From the main menu bar, click-select the “Tcl/Optimize” menu item.**

This will invoke a TCL script in the current directory and will synthesize the design bottom-up using the constraints generated by Time Budgeting. Notice that the whole synthesis run using the time budgeting constraints takes only a minute or so. We’ll compare this run time with the “characterize” style approach later.

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

We have met timing using time budgeted constraints and the slack is positive.

**Click on the “Cancel” button in the timing report window that had popped up.**

**From the main menu bar, click-select the “Reports/Report Area” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

This report gives the area distribution amongst the modules in the design. Keep this window around so we can compare the area results with the “characterize” constraints.

## **Step 6. Advantages of Time Budgeting over “characterize”**

Overconstraining a design results in long synthesis runs. We applied the constraints generated by “characterize” way and synthesized our design bottom-up. The runtime is very long and hence we have saved the database of that run for comparison by you.

**From the main menu bar, click-select the “Tcl/Cleanup” menu item.**

This will remove the current time-budgeted database from memory. Now we’ll load the other database.

**Click on the “ADB” icon on the menu bar.**

A “Read ADB files” menu will pop up.

**Double-click on “DerivedContext.adb” in the file list.**

This will read in a previously saved database. This database has the timing constraints generated by “do\_derive\_context”, the old way. (Also called the characterize way). Further the design was synthesized the same bottom up manner as the one in Step 5.

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

The timing for the worst path in the Budget\_0 is shown in the report. Notice that the path has a negative slack and hence is violating the constraints. Fill in the slack number on the line below.

Budget\_0 slack time \_\_\_\_\_ ns.

Notice that all of the negative slack in BudgetTop got applied to the submodule Budget\_0 making it overconstrained.

**Click on the “Cancel” button in the timing report window that had popped up.**

#### **Step 4. Generate timing constraints using Ambit’s Time Budgeting**

Now that we looked at the constraints generated by the “characterize” way, let’s see how Ambit’s Time Budgeting applies the slack to lower level modules.

**From the main menu bar, click-select the “Tcl/TimeBudget” menu item.**

This will invoke a TCL script in the current directory and will generate timing constraints for submodules A\_BudgetTop\_1 and Budget\_0 using time budgeting. The command used to generate the constraints is “do\_time\_budget”.

**Right-click on the module BudgetTop in module hierarchy and select “Set current and top”.**

This will make BudgetTop the current and top timing module.

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

The timing for the worst path is shown in the report. Notice that the path has a negative slack and hence is violating the constraints. Fill in the slack number on the line below.

BudgetTop slack time \_\_\_\_\_ ns.

Notice that this slack and the timing report is same as the one generated for BudgetTop in step 3. Thus, the timing constraints at the top level still have not changed, as expected.

**Click on the “Cancel” button in the timing report window that had popped up.**

**Right-click on the module A\_BudgetTop\_1 in module hierarchy and select “Set current and top”.**

This will make A\_BudgetTop\_1 the current and top timing module.

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

The timing for the worst path in the A\_BudgetTop\_1 is shown in the report. Notice that the path has a negative slack and hence is violating the constraints. Fill in the slack number on the line below.

A\_BudgetTop\_1 slack time \_\_\_\_\_ ns.

Notice that only some portion of the negative slack in BudgetTop got applied to the submodule A\_BudgetTop\_1. This distribution of slack over submodules is done by time budgeting, making the constraints realistic.

**Click on the “Cancel” button in the timing report window that had popped up.**

**Right-click on the module Budget\_0 in module hierarchy and select “Set current and top”.**

This will make Budget\_0 the current and top timing module.

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

The timing for the worst path is shown in the report. Notice that the path starts from address[1], goes through instance A, goes through instance B and ends at the output “match”.

**Click on the “Cancel” button in the timing report window that had popped up.**

So far, we have mapped the design, applied constraints and have seen that there are paths in this design that are not meeting the set timing goals. The worst path in the design is distributed over multiple instances in the hierarchy.

Such violated paths spanning multiple modules are common in designs. On a large design, if the user follows the bottom up synthesis methodology, timing constraints for each of the submodules have to be used to perform synthesis of respective modules. The old-fashioned way of creating the constraints has been “characterize” type command. Characterize however is known to Overconstrain the individual modules by counting the slack twice. Thus “characterize” generates unrealistic constraints and users have to manually set the time budgets.

Ambit’s time budgeting on the other hand employs an intelligent algorithm and applies constraints that are realistic.

### **Step 3. Generate timing constraints using the “characterize” way**

**From the main menu bar, click-select the “Tcl/DeriveContext” menu item.**

This will invoke a TCL script in the current directory and will generate timing constraints for submodules A\_BudgetTop\_1 and Budget\_0. The command used to generate the constraints is “do\_derive\_context” and results in constraints similar to “characterize”

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

The timing for the worst path is shown in the report. Notice that the path has a negative slack and hence is violating the constraints. Fill in the slack number on the line below.

BudgetTop slack time \_\_\_\_\_ ns.

**Click on the “Cancel” button in the timing report window that had popped up.**

**Right-click on the module A\_BudgetTop\_1 in module hierarchy and select “Set current and top”.**

This will make A\_BudgetTop\_1 the current and top timing module.

**From the main menu bar, click-select the “Reports/Report Timing” menu item.**

**Click the “Report” button at the bottom of the panel that pops up.**

The timing for the worst path in the A\_BudgetTop\_1 is shown in the report. Notice that the path has a negative slack and hence is violating the constraints. Fill in the slack number on the line below.

A\_BudgetTop\_1 slack time \_\_\_\_\_ ns.

Notice that all of the negative slack in BudgetTop got applied to the submodule A\_BudgetTop\_1 making it overconstrained.

**Click on the “Cancel” button in the timing report window that had popped up.**

**Right-click on the module Budget\_0 in module hierarchy and select “Set current and top”.**

This will make Budget\_0 the current and top timing module.

# Time Budgeting Demonstration

This step by step demonstration illustrates to the user advantages of the Automatic Time Budgeting feature of BuildGates. Automatic Time Budgeting results in shorter compile times, reduced number of synthesis iterations and smaller area.

The steps followed in the demonstration are

- 1) Read in the libraries and the RTL of a design.
- 2) Map the RTL into a technology, apply constraints
- 3) Generate timing constraints using the “characterize design” way
- 4) Generate timing constraints using Ambit’s Time Budgeting
- 5) Synthesize the design bottom-up using Time Budgeted constraints
- 6) See the advantages of Time Budgeting over “characterize”

## Step 1. Read in the libraries and the RTL of a design.

**Click on the “HDL” icon on the menu bar.**

A “Read HDL files” menu will pop up.

**Select either Verilog or VHDL button at the bottom of the panel.**

The HDL files for that language will appear in the “Files” list.

**Double-click on BudgetA.v or BudgetA.vhd, if you’re using VHDL, in the “Files” list.**

This transfers the files to the lower file list.

**Click on the “OK” button at the bottom of the “Read HDL files” panel.**

This step will read in the RTL file into BuildGates.

## Step 2. Map the RTL into a technology, apply constraints

**From the main menu bar, click-select the “Tools/Build Generic” menu item.**

This step will first link the whole design, will report the case statement statistics and register inferences found in the RTL. The design is now synthesized into a technology independent netlist form. The hierarchy tree of the design will pop-up in the Module Hierarchy Viewer.

**From the main menu bar, click-select the “Tcl/Map” menu item.**

This will invoke a TCL script in the current directory and will uniquify and map the design to lca300k library.

**Double-click on the “BudgetTop” item at the root of the hierarchy tree.**

This shows the schematic of the design in the schematic viewer.

**From the main menu bar, click-select the “Tcl/TimingSetup” menu item.**

This will invoke a TCL script in the current directory and will apply timing constraints such as clock constraints, data arrival times and data required times to the design.

**Right mouse-clicking inside the schematic viewer, select “Worst Path”.**

The worst path in the design is highlighted.