RunningVTR

Instructions on running VTR.

Updated May 7, 2013 by mikewangly

Running the VTR Flow

Introduction

VTR is a collection of tools that perform the full FPGA CAD flow from Verilog to routing. The stages of the flow are shown in the following diagram:

The flow consists of ODIN II (synthesis), ABC (optimization and tech mapping), and VPR (pack, place and route).

There is no single executable for the entire flow. Instead, scripts are provided to allow the user to easily run the entire tool flow. The following provides instructions on using these scripts to run VTR.

Running a Single Benchmark

The run_vtr_flow.pl script is provided to execute the VTR flow for a single benchmark and architecture (note that in the following <vtr> means the directory in which you placed the VTR source code):

```
<vtr>/vtr_flow/scripts/run_vtr_flow.pl <circuit_file> <architecture_file>
```

Circuit files are located here:

```
<vtr>/vtr_flow/benchmarks/
```

Architecture files are located here:

```
<vtr>/vtr_flow/arch/
```

The script can also be used to run a partial VTR flow. For command-line options see run_vtr_flow.pl.

Tasks

A framework is provided that manages execution of the VTR flow for multiple benchmarks. This is called a **task**. A task specifies which benchmark circuits and which architectures will be used. By default, tasks execute the run_vtr_flow.pl script for every circuit/architecture combination. There are multiple tasks already provided in the VTR release, and are located here:

```
<vtr>/vtr_flow/tasks
```

Of course, users may create their own tasks. See Creating and Modifying Tasks.

The task framework provides the additional benefits of file organization and statistics analysis. For this reason, it is suggested to use the task framework even when running a single benchmark circuit.

Executing Tasks

Tasks can be executed using the run_vtr_task.pl script:

```
<vtr>/vtr_flow/scripts/run_vtr_task.pl <task_name>
```

For command-line options see run_vtr_task.pl.

Task Output

The output of the script will include one line for each architecture/benchmark in the format:

```
<architecture>/<benchmark>...<status>
```

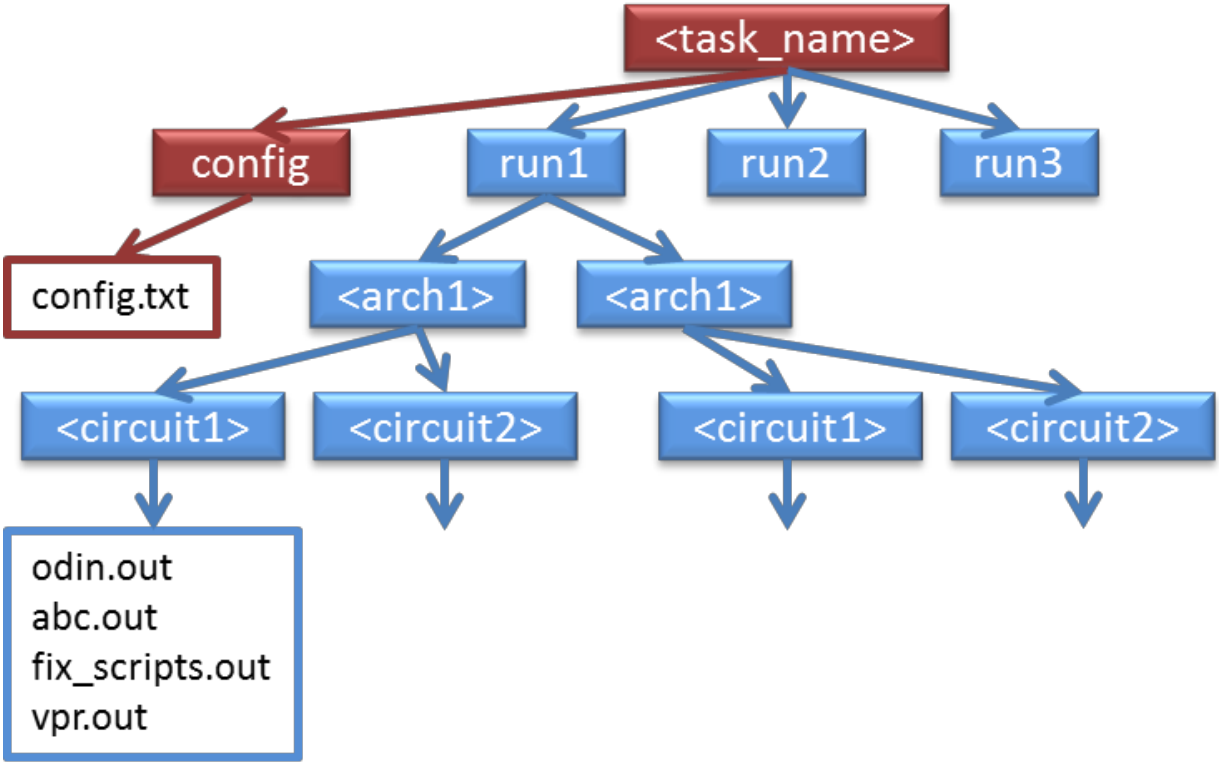
If the VTR flow succeeded, the status will be listed as 'OK'. The following is an example output for a task configured with two circuits and two architectures:

```
k4_N8_memSize16384_memData64_stratix4_based_timing_sparse/ch_intrinsics...OK
k4_N8_memSize16384_memData64_stratix4_based_timing_sparse/diffeq1...OK
k6_N10_memSize16384_memData64_stratix4_based_timing_sparse/ch_intrinsics...OK
k6_N10_memSize16384_memData64_stratix4_based_timing_sparse/diffeq1...OK
```

The output is simply a collection of the outputs from the underlying script used for the execution of each benchmark/architecture. If the user configures the task to use a different script than the default run_vtr_flow.pl, the output will be different.

Task File Organization

The below diagram illustrates the file organization for each tasks. All of the blue items are automatically created on execution of the task. Each execution of the task will create a new folder named run<#>, where # begins at 1 and increments with each execution. This maintains a history of all executions of the task. Within the 'run#' folder, folders will be created for each architecture type. Inside each architecture folder, folders will be created for each circuit. Within each circuit folder, the run_vtr_flow.pl script will be executed, which will produce output files for each stage of the VTR flow.



Extracting and Verifying Statistics

Scripts are provided to extract statistics from the results of the VTR flow. After executing a task, parse_vtr_task.pl can be used to parse and assemble statistics for all of the circuit/architecture combinations. The results can also be verified against a golden set of results.

If the task framework was not used, parse_vtr_flow.pl can be used to parse statistics for a single execution of the VTR flow.

Enter a comment: Hint: You can use Wiki Syntax.

Submit