Limitations of Incremental Signal-Tracing for FPGA Debug
IncTrace — v1.0

Eddie Hung, Steven J. E. Wilton
Department of Electrical and Computer Engineering
University of British Columbia
{eddieh,stevew}@ece.ubc.ca

August 2012

This is a supporting document for the IncTrace source code that implements incremental-signal tracing on the VPR6 FPGA CAD tool (part of the Verilog-To-Routing project [3]), as described in our paper [2].

In this implementation, incremental-tracing can be used to incrementally connect (i.e. without moving any of the placed blocks, nor ripping up any of the existing routing) a set of nets specified by the user to an input pin of a free memory-block, under the assumption that these unoccupied memories can be transformed into a trace-buffer with no overhead. For more details regarding this work, please consult our paper.

The included patch is designed to be applied to `vtr_release_1.0_full.tar.gz` available from `http://code.google.com/p/vtr-verilog-to-routing/`. The md5sum of this file is:

```
f5e513d0cb98ec8ab90da654b6728ace        vtr_release_1.0_full.tar.gz
```

After extracting the tarball, place the patch file into the `vtr_release` sub-directory. To apply the patch, issue the following command from this same location:

```
vtr_release$ patch -p0 < IncTrace_v1.0.patch
```

After building VPR, the following new switches will now be available from the command-line:

`--inc_trace`	Enable incremental-tracing.
`--inc_nets <comma-separated-int>`	Trace the (local) nets specified in this comma-separated list; e.g. `--inc_nets 1,2,3,4,5`
`--inc_swap_bles`	Enable swapping the order of BLEs within the logic cluster to improve tracing flexibility. See paper for more details [2].

Note: Incremental-tracing does not support the minimum channel-width binary-search feature within VPR, hence, a fixed channel width must always be specified using the `--route_chan_width <int>` switch.

Furthermore, the following switch has been modified:

```
--router_algorithm [breadth_first | timing_driven | directed_search | read_route]
```

The `read_route` option (in combination with the `--route` and `--route_file` switches) allows users to read into VPR a previously-routed solution for incremental-tracing.
Note: This patched version VPR can only read in `.route` files that were previously generated by a patched VPR instance: this is because new routing files contain additional information to allow its reconstruction.
Note: In order to read a routing file, its channel width must be specified!

Net Selection

In this modified version of VPR, after the packing stage, VPR will now write out a `.netinfo` file. This file is in CSV format, and specifies one signal per line, with each entry separated by a comma: local net number, net name, global net number, block number.

Nets to be traced (using `--inc_nets`) must be specified using this local net number. Local nets that do not have a block number (last column) cannot be traced. We have observed that these nets exist only between an LUT output and a FF input; their values can be observed by tracing the FF output instead.

Example `mkPktMerge.netinfo` excerpt:

```
0,top^EN_iport1_put,0,11
1,top^EN_iport0_put,1,10
2,n2223,2,4
3,n2225,3,7
4,top^EN_oport_get,4,18
5,n2255,5,3
6,n2278,6,6
7,top^FF_NODE 2324,7,8
8,top^FF_NODE 2323,8,5
9,top.arSRLFIFO_b+fi1.generic_fifo_sc_b+fifo_1^FF_NODE 867,9,3
```

One solution for selecting influential signals in a circuit is described in [1].

Usage Examples

Only pack the circuit (source netlist `mkPktMerge.blif` must be present in the working directory) in order to dump out the signals into the `.netinfo` CSV file:

```
./vpr sample_arch.xml mkPktMerge --pack
```

Perform packing, placement, and routing (using a fixed channel-width of 50 tracks) before incrementally tracing the local nets: 0, 1, 2 and 3:

```
./vpr sample_arch.xml mkPktMerge --route_chan_width 50 --inc_trace --inc_nets 0,1,2,3
```

Read in a previous packing (by default, VPR will use `mkPktMerge.net` as the `--net_file`, but re-place and re-route (with channel-width of 60) before incrementally-tracing the local nets 20–29 with LE swapping:

```
./vpr sample_arch.xml mkPktMerge --place --route --route_chan_width 60 --inc_trace
   --inc_nets 20,21,22,23,24,25,26,27,28,29 --inc_swap_bles
```

Read in previous packing, placement and routing (with channel-width of 50) before incrementally-tracing the local nets 30, 31, 32, 33, with LE swapping:

```
./vpr sample_arch.xml mkPktMerge --route --router_algorithm read_route
   --route_chan_width 50 --inc_trace --inc_nets 30,31,32,33 --inc_swap_bles
```

Source Code

This section briefly describes the changes that were made to implement incremental signal-tracing inside VPR. For further information about the following VPR functions, please consult the unofficial, automatically generated documentation (using Doxygen) located at: `http://ece.ubc.ca/~eddieh`.

After packing, VPR calls the `place_and_route()` function. Inside this function, if incremental signal-tracing is requested and the channel width has been fixed, then after a successful placement and routing of the original user-circuit, VPR will begin incremental-tracing by first calling `inc_parse_trace_string()` and then entering the `inc_trace()` function. This function, along with other incremental signal-tracing source code, is located in the `SRC/inc` sub-directory of VPR. After incremental tracing is complete, VPR continues as normal beginning with the timing analysis stage.

`inc_parse_trace_string()` parses the command-line string into integers, and transforms local nets (nets that exist entirely within a logic cluster) into global nets that can be routed on the global interconnect, via the `inc_vpack_to_clb()` function.

`inc_trace()` is responsible for setting up the tracing environment, such as resetting data structures, and marking all free memory blocks as potential trace-buffer targets (`mark_targets()`). Once these tasks are complete, it calls `inc_trace_nets()`.

`inc_trace_nets()` is where the bulk of the work happens, and is very similar to the original VPR routine for breadth-first search: `try_breadth_first_route()`. Here, the function will iteratively execute the Pathfinder negotiated congestion routing algorithm until a valid solution is found, or it exceeds the maximum number of iterations. In each iteration, for each of the nets that are to be traced, it will rip up any existing incremental-traces (note: the original routing is not ripped up), attempt to re-route a new trace using a breadth-first strategy (calling `inc_route_net()` to do so). At the end of each iteration, a check is performed to see if the solution is valid (`inc_feasible_routing()`) if no incremental resources are being overused.

`inc_route_net()` performs the breadth-first incremental-routing for each trace, and is based on the original VPR routine: `breadth_first_route_net()`. The key difference here is that it performs breadth-first expansion from *all* points of the existing user net, searching until *any* trace-pin sink is found. For local nets (nets that exist entirely within a logic cluster), it expands from all local OPINs within the same

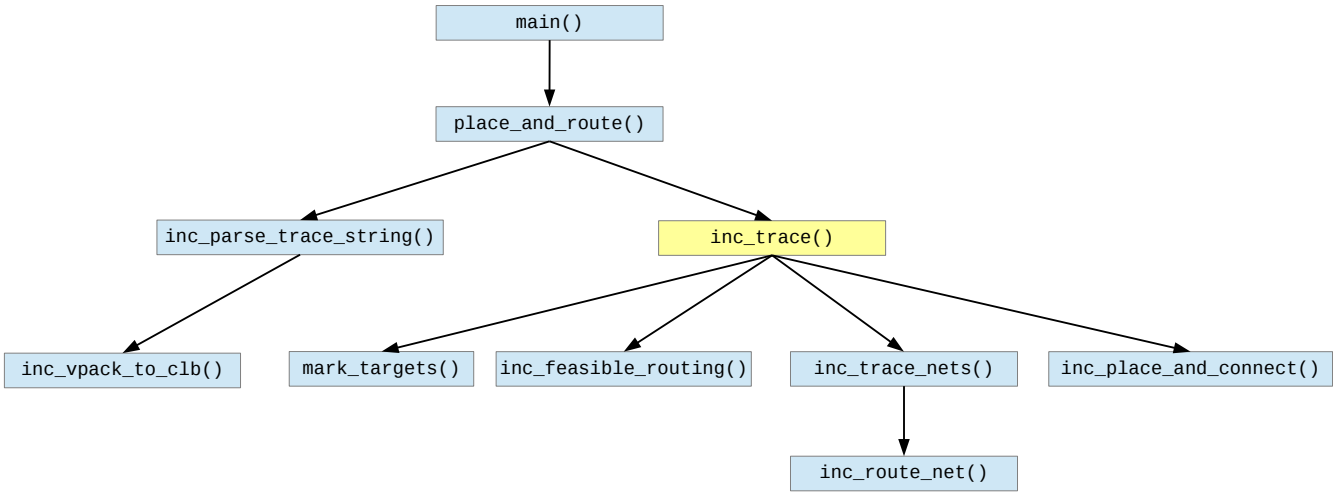

Figure 1: Caller-graph for main Incremental Signal-Tracing functions.

cluster. In addition, the expand neighbours function (`inc_breadth_first_expand_neighbours()`) will only add routing resources onto the priority queue only if there it contains spare capacity.

After a valid solution is found, `inc_place_and_connect()` is called to place a trace-buffer into the free memory location (if one hasn't been placed already) and then to fully connect this trace net to it. After this, `inc_trace_nets()` will return and VPR will resume.

Figure 1 illustrates the caller-graph for these main functions.

References

[1] Eddie Hung and Steven J. E. Wilton. Scalable Signal Selection for Post-Silicon Debug. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

[2] Eddie Hung and Steven J. E. Wilton. Limitations of Incremental Signal-Tracing for FPGA Debug. In *FPL 2012, International Conference on Field Programmable Logic and Applications; Oslo, Norway*, August 2012.

[3] Jonathan Rose, Jason Luu, Chi Wai Yu, Opal Densmore, Jeff Goeders, Andrew Somerville, Kenneth B. Kent, Peter Jamieson, and Jason Anderson. The VTR Project: Architecture and CAD for FPGAs from Verilog to Routing. In *Proc. of the 20th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 77–86, 2012.