University of Pennsylvania
Department of Electrical and Systems Engineering
Electronic Design Automation

ESE535, Spring 2011 Assignment #5 Monday, March 14

Due: Part A: Monday, March 21, beginning of class.
Due: Part B: Monday, March 28, beginning of class.

Resources You are free to use any books, articles, notes, or papers as references. Provide citations in your writeup as appropriate.

Collaboration You may give tutorial assistance on using OS, compiler, and debugging tools. All code development should be done independently. You may **not** share code or show each other code solutions. All writeups must be the work of the individual.

Writeup Turn-in assignments on blackboard. See details on course web page. No handwriting or hand-drawn figures. See details below on what you need to turn in and the format.

Figure 1 shows the basic routing architecture. Routes that do not change from horizontal to vertical routing tracks can be linked together. Routes switch from a horizontal track to a vertical track through a corner turn unit as shown. Corner turn units are fully populated (can connect any horizontal track to any vertical track at a particular PE). Figure 2 (at end) shows a larger mesh of PEs.

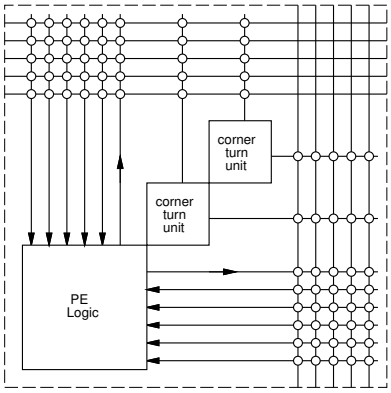

Figure 1: PE with Two Corner Turn Units

Assignment 5 Task Develop and implement an algorithm to route your placed circuit netlist from assignment 4 onto a time-multiplexed, single corner-turn routing network minimizing the maximun number of corner turns units.

- routed design should evaluate in the lower bound makespan implied by placement and this corner-turn timing model

- each route should take at most one corner turn
  - intra-cluster routes do not require a corner turn
  - if the source and sink are in the same row or column, no corner turn is needed
- fanout is allowed
  - the same corner turn unit may be used in the a single timestep for multiple different sinks from the same source
  - a source may use multiple different corner turns to reach its different sinks
- each PE contains a number of corner turn units (the thing you are trying to minimize)

$$ct_{max} = \max_{\text{PEs } j}(ct_j) \tag{1}$$

- each corner turn unit can provide:
  - one corner turn in a timestep
  - at most `cts_per_cluster` corner turns across all timesteps[1]

  Therefore, the number of corner turn units needed at a PE to support a route is:

$$ct_{instant_j} = \max_{\text{timesteps } t}(\text{corner turns made at PE } j \text{ in timestep } t) \tag{2}$$

$$ct_{total_j} = \left\lceil \left(\frac{1}{\texttt{cts\_per\_cluster}} \right) \sum_{\text{timesteps } t} (\text{corner turns made at PE } j) \right\rceil \tag{3}$$

$$ct_j = \max(ct_{instant_j}, ct_{total_j}) \tag{4}$$

We are making two potentially artificial restrictions to simplify the problem and the support code:

- routes may not take multiple corner turns
- a signal is launched from the PE when it is evaluated on the PE; it may not be launched multiple times (on multiple different cycles). You may choose the timestep you schedule the source LUT or IO to evaluate on the PE, but then you must route all of its sinks starting at that time.

Two "unrealistic" assumptions we are making:

- no explicit bound on channel width—there is no limit to the number of straight-through routing connections used instantaneously in a channel
- no cluster input banwdith limitation—there is no bound on the number of signals that arrive at a PE on a give cycle

[1] `cts_per_cluster` is named to match `luts_per_cluster` but might be more accurately called "corner turns per corner turn unit".

In practice, we would also want to place limits on both of these. We are ignoring them now to keep the problem simple assuming the corner turns are the dominant effect.

See [1] for a description of an architecture proposed with corner-turn routing, including a description of their routing solution. We add time-multiplexed reuse of the corner turn, ignore details of channel width and straight-through connections, and make the single corner-turn restriction strict.

For assignment 6 and 7, you may elect to remove the unrealistic, simplifying assumptions and restrictions we are accepting for this assignment.

Timing The corner turn includes a mandatory register. It will take one or more clock cycles to reach the corner turn form the sink, then one or more clock cycles to reach the sink.

$$T_{\Delta x} = \lceil (x_{node} - x_{ct}) \times \texttt{inter\_cluster\_per\_manhattan\_hop\_delay} \rceil \quad (5)$$

$$T_{\Delta y} = \lceil (y_{node} - y_{ct}) \times \texttt{inter\_cluster\_per\_manhattan\_hop\_delay} \rceil \quad (6)$$

$$T_{interconnect\_delay} = T_{\Delta x} + T_{\Delta y} \quad (7)$$

If the source evaluates at time T_{src} then:

- A vertical first route uses the corner turn on cycle $T_{src} + T_{\Delta y}$.
- A horizontal first route uses the corner turn on cycle $T_{src} + T_{\Delta x}$.
- The earliest the sink (successor) can evaluate is cycle $T_{src} + T_{\Delta x} + T_{\Delta y}$.
  originally was incorrectly stated as: $T_{src} + T_{\Delta x} + T_{\Delta y} + 1$.

Code Base Changes:

Pickup the code in `assign5.tar` from `~ese535/spring2011/assign5.tar` on `eniac`. Changes have been made to `mesh.c`, `sched_main.c`, and `check_precedence.c`. The following have been added:

- `check_routed.c` – this code checks that all nets have been routed
- `greedy_vfirst_route.c` – this is a dumb router that always routes first vertically then horizontally
- `your_route.c` – this is where you put your router code

You will be comparing the result of your router with the provided `greedy_vfirst_route` on your placement from assignment 4. The new `sched_main` is setup to run the `greedy_vfirst_route` on your placement from assignment 4 as option 4, then run your router on your placement from assignment 4 as option 5. You should update the option 4 and 5 code as necessary to properly call your placer and router.

Part A turnin: A single PDF that includes:

1. Ordering
   - How might the order of net routing impact the behavior of the router?
   - In what order is `greedy_vfirst_route` routing the nets?
   - In what order did [1] route the nets and why?
   - What other factors might be important in route ordering?
2. Consider a LUT, A, whose ASAP time is 4 in the PE at (7,7) with a single successor B whose ALAP time is 8 in the PE at (10,4). Assume inter_cluster_per_manhattan_hop_delay = 0.2. List all combinations of timesteps for A and corner turn locations and associated corner turn timesteps that could be used to route the A to B link.
3. Merge your assignment 4 placement with the new `sched_main` including the call to the provided greedy vertical first router and complete the following comparison table with the costs in `test/Makefile`:

Design	ct_{max}		
	ASAP	assign4	Δ_A%
example	2		
e64	6		
s1423	4		
alu4-em4	10		
frisc-em4	7		
s298-em4	4		

Δ_A% is the percent improvement of your assignment 4 placement algorithm over ASAP when using the greedy vertical first router baseline:

$$\Delta\% \text{ improve} = \frac{ASAP\_vfirst\_ct_{max} - Assign4\_vfirst\_ct_{max}}{ASAP\_vfirst\_ct_{max}}$$

Part B Turnin: You will need to upload two files. We have created separate assignments on blackboard so that you only need to submit a single file to each assignment.

- **assign5b-writeup**: a single PDF with:
  1. Description of your algorithm and the code that supports it.
     (a) A high-level, English text description of your strategy. Use references if appropriate. The description should relate your solution to quesions 1 and 2 on Part A.
     (b) Define any cost functions used by your algorithm.
        i. For your algorithm, is it important to compute the changes to these cost functions based on an incremental move or potential assignment?
        ii. How does the cost function reward an incremental move or assignment toward the overall goal of maximum corner turn unit minimization?
        iii. How expensive (in runtime) is it to compute the change in cost for an incremental move or assignment?
     (c) Provide pseudocode for your algorithm.

2. Complete a version of this comparison table with the costs in `test/Makefile`:

Design	ct_{max}				
	ASAP	assign4	assign5	Δ_{AV}%	Δ_4%
example	2				
e64	6				
s1423	4				
alu4-em4	10				
frisc-em4	7				
s298-em4	4				

Δ_{AV}% is the percent improvement of your assignment 4 and 5 algorithm over the ASAP and greedy vertical first router baseline:

$$\Delta\% \text{ improve} = \frac{ASAP\_vfirst\_ct_{max} - Assign5\_ct_{max}}{ASAP\_vfirst\_ct_{max}}$$

Δ_4% is the percent improvement of your assignment 5 routing algorithm over the greedy vertifical first router baseline using your assignment 4 placement:

$$\Delta\% \text{ improve} = \frac{Assign4\_vfirst\_ct_{max} - Assign5\_ct_{max}}{Assign4\_vfirst\_ct_{max}}$$

- **assign5b-code**: a single tar file with your code (no binary files, but in an archive like the provided support so it can be unpacked and built; this means the make file should be updated to build the application with any additional source files you may have added)
  - run `make clean` in both the code and test directories
  - use `make assign5.tar` to create the tar file
  - test that you can unpack your `assign5.tar` and build and run tests from there before you upload to blackboard

References

[1] Nicholas Weaver, John Hauser, and John Wawrzynek. The SFRA: A corner-turn FPGA architecture. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, March 2004.

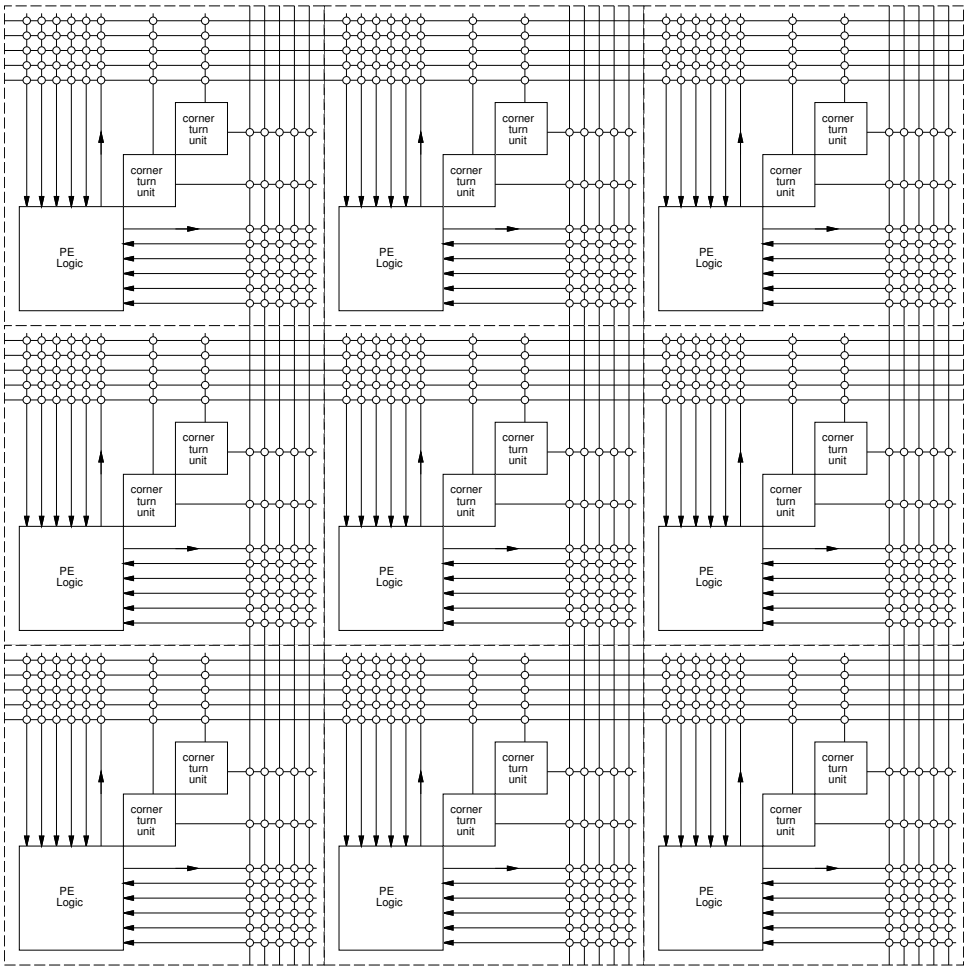

Figure 2: Corner Turn Mesh Routing Architecture