BFS on FPGA

Nina Engelhardt

August 25, 2015

Breadth First Search

- Find minimal spanning tree
- Equivalent to shortest path from root to all nodes, with unweighed edges
- For correct parallel implementation, synchronization barrier between “levels” is important

Vertex-Centric Algorithms

Distributed, synchronous execution in phases separated by global barriers:

1. Node receives messages from previous phase
2. Node updates internal status based on data received
3. Node sends messages with new data to neighbors for next phase

Nodes are *active* if they receive at least one message in a phase. Computation terminates when no active nodes remain.

Vertex-Centric Algorithms

Salient features:

- Need to store all messages between phases! Potentially as many as there are edges in the graph.
- Need to be able to determine activity/detect termination.

Implementation

- Split nodes across multiple processing elements
- NoC interconnect to transfer and store messages

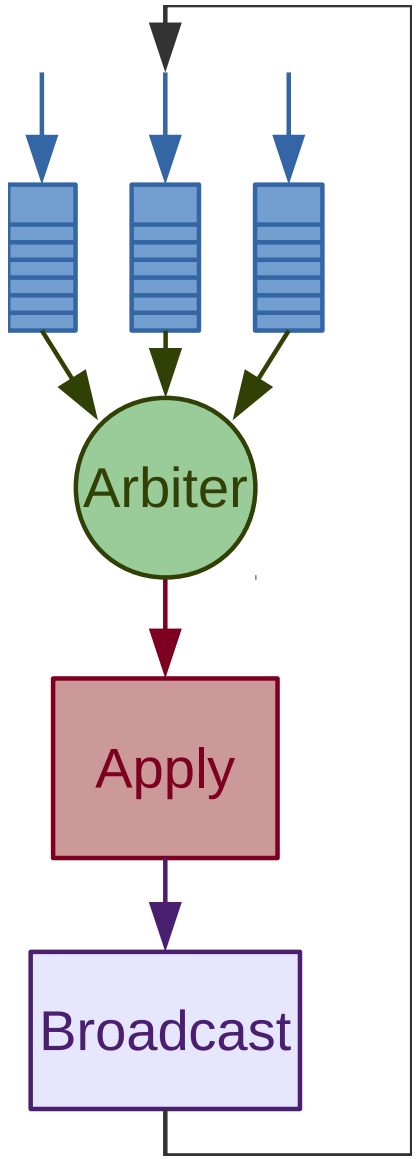
Implementation: NoC

- Currently, just $N \times N$ FIFO queues linking all PEs point-to-point (including self-link)
- "Floating barrier": assuming in-order delivery of messages, PE signals end of phase by sending a barrier marker
- Other PEs will not consume messages in queue behind barrier marker until next phase
- PE passes to next phase when barrier marker received from all PEs

Implementation: PE

Components:

- Incoming message queues
- Apply module
- Update broadcast module



Implementation: Incoming Messages

- N queues buffering messages from each PE
- Round-robin consume message from a queue where non-barrier message at front
- If all queues show barrier, consume all barrier markers and signal barrier to Apply module
- Two barriers in a row = no active nodes in this PE during last phase

Implementation: Apply

- Contains node-associated data storage
- Algorithm-specific. In BFS:
  - message contains ID of destination node and sender node
  - check destination node storage. If not yet visited ($= 0$), store parent information (sender node ID) and output ID of updated node
- Barrier messages are passed along to output in order (increase internal counter)

Implementation: Broadcast

- Receives message payload to broadcast and sending node ID (in BFS, both are the same)
- Looks up neighbors of sending node in edge storage
- Sends message to all neighbors
- For barriers, sends barrier marker to all PEs (including self)

Issues

Things to work on near term:

- Deadlock if not enough queue space (buffer after apply most space-effective)
- Number of PEs vs. PE size - more nodes per PE mean less area overhead, but also less parallelism
- Simulation only works up to 64 PEs

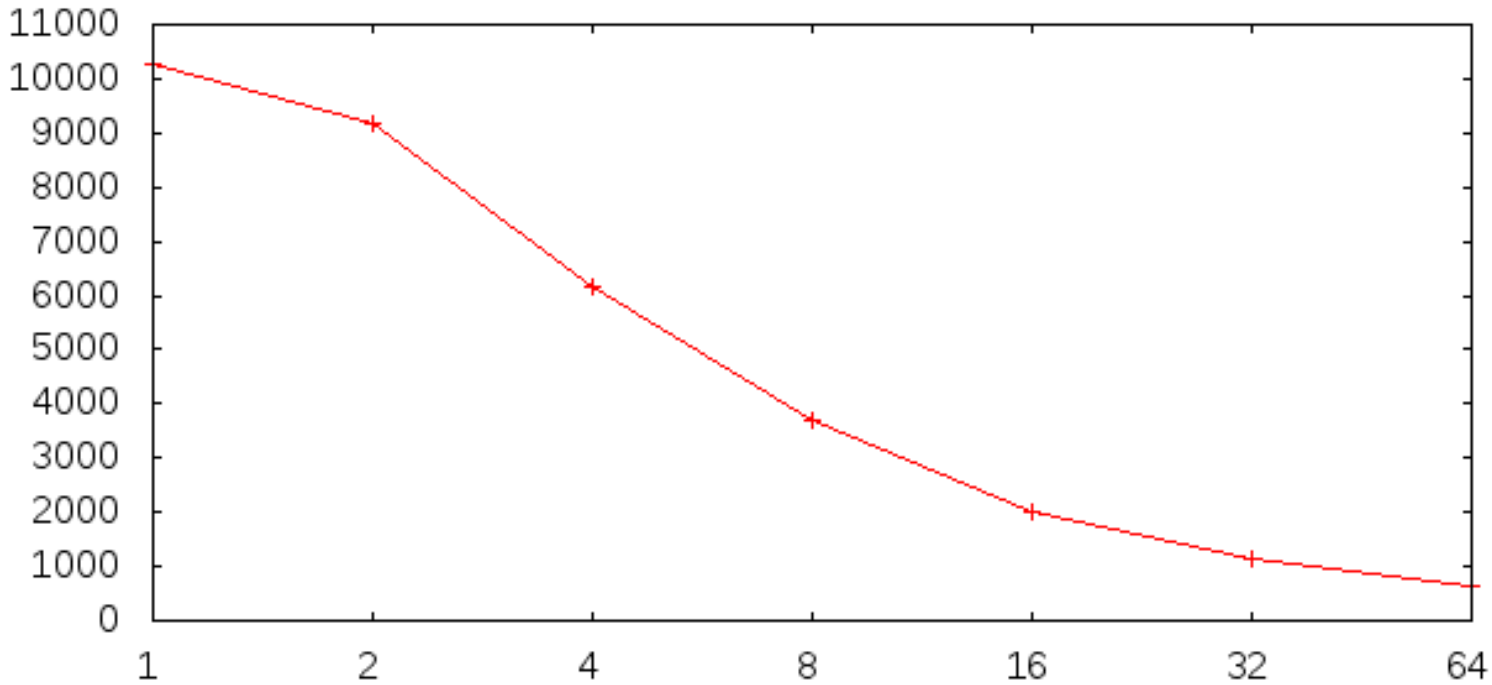
Issues

Things to work on long term:

- NoC: $N \times N$ FIFO approach not scalable
- How to partition nodes across PEs (currently filled in order)
- How to deal with nodes with many neighbors (split single node across multiple PE?)

Some simulation results

GRCite graph has 1953 nodes and 8307 edges.[1]



[1] Due to clustering of high-arity nodes, could not reduce edge storage size going from 2 to 4 PEs.

Graph density effect on runtime

Random connected graph (different one each time) with 32000 nodes and n edges:

- $n = 34000$: 3445 cycles taken.
- $n = 64000$: 5314 cycles taken.
- $n = 128000$: 9470 cycles taken.
- $n = 192000$: 13711 cycles taken.