Verilog to Routing (VTR): A Flexible Open-Source CAD Flow to Explore and Target Diverse FPGA Architectures

Vaughn Betz

With thanks to all VTR contributors, and particularly Kevin Murray for assistance in creating this presentation
Agenda

• FPGAs: Basics & Utility
• Flexible CAD: Motivation and Challenges
• Verilog to Routing: Recent Progress
  • Architecture Generality
  • AIR: Faster, More Adaptive Routing
  • VTR 8.0: QoR Improvements
  • Reinforcement Learning for Adaptive Placement
• Summary
FPGA Basics & Utility
FPGA Basics

Programmable Hardware

Can implement arbitrary circuits

- Look-Up Table (LUT) implements logic
- Flip-Flop (FF) stores sequential state
FPGA Basics

Logic Block (LB)

- Group of Logic Elements (LEs)
FPGA Basics

Device

- Grid of Function Blocks
- Interconnect

Routing Switch
FPGA Basics

Mapping Applications to an FPGA

• Modern FPGAs are large (>10M LEs)

• Use Computer Aided Design (CAD) flow!
  • Designer: provides *behavioural* specification
  • Algorithms & tools: figure out the *details*
Why Use an FPGA?

1. Flexible I/O
   • No off-the-shelf chip with right I/O

2. Power Efficiency
   • CPU/DSP/GPU performance per watt too low

   45 nm CPU energy breakdown [1]

   - 25 pJ
   - 6 pJ
   - Control
   - 70 pJ

   - Cache Access
   - Register File Access
   - Add

   - 0.03 pJ for 8-bit int add
   - 0.90 pJ for 32-bit fp add

3. Programmability
   • Dedicated ASIC not practical (e.g. changing workload)

Widely Used Across Many Domains

CAD flow key to success
Flexible, Open Source CAD: Motivation and Challenges
I: FPGA Architecture Research

Challenges:

• Need representative benchmarks

• Describe FPGA Architecture

• Need high quality CAD flow

Benchmarks

FPGA Architecture #1

Need CAD Flow for Architecture #1

Speed/Area/Power

FPGA Architecture #2

Need CAD Flow for Architecture #2

Speed/Area/Power
Why New FPGA Architectures?

1. Process Technology Changes
   - Wire resistance, power limits, interposers, ...

2. Application Changes
   - New standards, faster processing: 4G → 5G, ...
   - Deep Learning → new compute demands
   - Embedded FPGAs

3. Innovative Architecture Ideas!
II: FPGA CAD Research

New idea for CAD algorithm

Need infrastructure!
• Do not build it all
• Focus on your contribution

Common framework:
• Best practise
• Compare results

Flexible CAD: more impact
• VTR enhancements help Symbiflow (QuickLogic EOS S3, Xilinx Artix7), start-ups, architecture research, …
III: Program Novel Spatial Compute Architectures

• Have an architecture? Build it!
• FPGA/FPGA+??/?
• Need a CAD flow!

ReImagine: FPGA + Imager (MIT Lincoln Lab) [1]

OpenFPGA (U of Utah) [2]
PRGA (Princeton) [3]

VTR: Verilog to Routing
Verilog to Routing (VTR) Project

Open Source FPGA CAD Flow

Approach:
- Architecture Agnostic
- Data-driven FPGA Architecture description
VTR Challenges

- Explore larger design space

Architecture
Generality

Quality
Run-time

- Accurate architecture conclusions
- Reduces designer effort
- Can take hours to days (for large designs)

Can we have all three?
Outline

1. Architecture
   - Verilog
     - ODIN II
       - ABC
       - Netlist
     - VPR
       - Pack
       - Place
       - Route
       - Analysis
     - STA
     - QoR

1. VTR 8 Capabilities & New Features
2. AIR: Adaptive Incremental Router
3. VTR 8 QoR, Run-time & CAD Enhancements
4. Reinforcement Learning Enhanced Placement
VTR 8 Capabilities & New Features

1. AIR: Adaptive Incremental Router
2. VTR 8 QoR, Run-time & CAD Enhancements
3. Reinforcement Learning Enhanced Placement

Block Modeling

• Supports both soft & hard blocks
• Arbitrary:
  • Netlist Primitives
  • Hierarchy
  • Connectivity
General Device Grids

Perimeter

Single

Fill

Column

Region
Arbitrary Grid Example

- IOs
- Logic Block
- DSP
- Block RAM
- NoC Router
- Hard Block

New!
Detailed Routing Architecture

- Direct Block-Block Connection
- Wire-Wire Connections (switch-block)
- Block Input Connection
- Routing Wires
- Block Output Connection
Customizable Routing Architectures

Switch-block Locations

- All
- External
- External Full Internal Straight

Wire Connectivity

Non-Configurable Edges
Fully Custom Routing Architectures

- Load RR graph from file
- VPR or non-VPR generated
- Full control of routing network
- Allows freezing RR graph (e.g. taped-out)
Analysis & Modeling

• Run analysis independent of optimization
• Improved visualizations & reporting
• Improved area models
• Full featured timing analysis
• Min & max timing models
• Multi-clock primitives

Multi-clock Primitives
Routing Utilization
Outline

1. VTR 8 Capabilities & New Features
   - AIR: Adaptive Incremental Router

2. VTR 8 QoR, Run-time & CAD Enhancements
   - Reinforcement Learning Enhanced Placement

FPGA Routing Problem

- Model interconnect network as Routing Resource (RR) graph:
  - Nodes: Conductors (wires/pins)
  - Edges: Configurable switches

- FPGA Routing Problem:
  - Find embedding of netlist in RR graph
  - Requires finding many non-overlapping trees in RR graph
  - Minimize timing and wiring (power)
  - Reasonable run-time

- Usually solved as single combined global/detailed routing stage
Negotiated Congestion Routing

- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other
AIR: Adaptive Incremental Router

• Negotiated congestion router

• Adaptive Routing

• Lazy Routing

Improve quality & robustness

Avoid unnecessary work!
Connection Router: Architecture Adaptation

\[ \text{TotalCost}(s, n, t) = \text{PrevCost}(s, n) + \text{NodeCost}(n) + \text{ExpectedCost}(n, t) \]

\[ \text{NodeCost}(n) = (1 - \gamma) \cdot b(n) \cdot c(n) + \gamma \cdot \text{Delay}(n) \]

AIR uses adaptive:
- Lookahead: \textit{ExpectedCost}
- Base Costs: \textit{b(n)} (see paper)
Adaptive Lookahead

• Observation: Lookahead should adapt to routing architecture

• Challenges:
  • Many FPGA Architectures
  • Huge Graph
  • Reasonable build time
  • Fast evaluation

• Approach:
  • Profile graph with undirected Djikstra flood & automatically data reduce
  • Adapts to graph
  • More accurate → faster for equivalent/better quality
Net Router: Lazy Routing

AIR Routes Nets Lazily:
• Efficiently handle High-Fanout nets
• Incrementally re-route congested nets
High-Fanout (HF) Routing

- Observation: Most routing of HF net irrelevant for a particular sink
  - But still consider as potential branch points (expensive)

- Optimization:
  - Only consider branch points spatially *nearby* target
  - Don’t even look at others
  - Limit search to region near target
HF Router Expansion

Traditional

Spatial Lookup
**Incremental Routing**

• Observation: Most net connections routed legally

• Optimization:
  • Rip-up illegal sub-trees
  • Re-route *only* those sub-tree connections

• Challenge: May degrade critical path
  • Fix: Also rip-up delay sub-optimal connections
Titan Flow:
- Synth. & Tech-Map with Quartus
- Pack/Place/Route with Quartus or VPR

Titan Benchmarks:
- Target Stratix IV
- 23 designs, 90K-1.9M primitives

• Academic Routers:
  - VPR 7
  - CRoute
  - AIR

• Commercial Routers:
  - Intel Quartus 18.0
AIR Outperforms Academic Routers

- **Route Time**
  - VPR 7: 0.2
  - CRoute: 0.3
  - AIR: 0.1
  - > 6.6x run-time reduction
  - Incr. Routing
  - HF Routing
  - Lookahead

- **Routed CPD**
  - VPR 7: 0.9
  - CRoute: 0.8
  - AIR: 0.7
  - 19% faster critical path
  - Lookahead

- **Routed WL**
  - VPR 7: 0.9
  - CRoute: 0.8
  - AIR: 0.7
  - 15% less wiring
  - Resource Base Costs
  - Lookahead
AIR is Faster than Quartus

<table>
<thead>
<tr>
<th></th>
<th>Intel Quartus 18.0</th>
<th>VPR 8 Place + AIR Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route Time</td>
<td>0.0</td>
<td>0.2</td>
</tr>
<tr>
<td>Routed CPD</td>
<td>1.0</td>
<td>1.2</td>
</tr>
<tr>
<td>Routed WL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4x run-time reduction

Gap due to lower quality placement
AIR Effectively Resolves Congestion

- AIR Congestion Oblivious (min. delay)
- AIR Congestion Free (legal)

Critical Path not degraded
Wiring reduced
Outline

Architecture

Verilog

ODIN II

ABC

Netlist

VPR

Pack

Place

Route

Analysis

STA

QoR

1. VTR 8 Capabilities & New Features

2. AIR: Adaptive Incremental Router

3. VTR 8 QoR, Run-time & CAD Enhancements

4. Reinforcement Learning Enhanced Placement

K. E. Murray, et al., “VTR 8: High Performance CAD and Customizable FPGA Architecture Modelling”, In submission to ACM TRET, 2020
## Titan Benchmark Completion

<table>
<thead>
<tr>
<th>Tool</th>
<th>Routed Benchmarks</th>
<th>Unroutable Benchmarks</th>
<th>Device Too Small</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTR 7</td>
<td>14</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>VTR 8</td>
<td>23</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Quartus 18.0</td>
<td>20</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### VTR 8 Titan Benchmark Implementations

![Images of benchmark implementations for VTR 8](attachment:image.png)
Packing & Placement Enhancements

Avoid highly interconnected (unnatural) clusters
- Avoid packing unrelated logic
- Include more indirect (e.g. transitive) connectivity measures
- De-prioritize high-fanout nets

Improved Placement Exploration
- Better macro/carry-chain swapping
- Improved swapping of sparse blocks
Run-time & memory comparable to commercial tools
VTR 8 QoR

The chart illustrates the quality of various components in Quartus 18.0. The values for LABs, DSPs, M9Ks, M144Ks, Routed Wirelength, and Routed CPD (geomean) are shown.

- LABs: 0.95
- DSPs: 0.97
- M9Ks: 1.26
- M144Ks: Quality in the range of commercial tools
- Routed Wirelength: 1.26
- Routed CPD (geomean): 1.2

The chart indicates a high level of quality for most components, with Routed CPD (geomean) being 1.2.
Outline

1. Architecture
   - Verilog
     - ODIN II
       - ABC
         - Netlist

2. VPR
   - Pack
   - Place
   - Route
   - Analysis
   - STA

3. VTR 8 Capabilities & New Features
   - AIR: Adaptive Incremental Router
   - VTR 8 QoR, Run-time & CAD Enhancements
   - Reinforcement Learning Enhanced Placement

K. E. Murray and V. Betz, “Adaptive FPGA Placement Optimization via Reinforcement Learning”, MLCAD, 2019
Reinforcement Learning & CAD
CAD Tool Development: Human-in-the-loop

Large solution space → Use heuristics

With human-in-the-loop:

• Slow
• Simple heuristics, limited tool parameters (to keep tractable)
• Tune for average case (can’t investigate every benchmark design)
CAD Tool Development: Reinforcement Learning (RL)

With RL:

- Human \textit{out} of the loop!
- Learn better heuristics: exploit more information, more parameters
- Online adaptation $\rightarrow$ better than average case
FPGA Placement
Simulated Annealing (SA) Placement

• Modify placement by making ‘moves’
• Accept/Reject move based on:
  • Cost change
  • Temperature (hill climbing)
Move Generation
Many possible types of moves!

Simple: random swap

Smart: directed move to ‘good’ location (wirelength, timing)

Complex: Analytic

Complex: Assignment

Many considerations:
• Frequencies of different moves
• Situation dependent?
• Move ‘strength’ vs run-time

Treat as RL Problem!
**RL Move Generator**

Actions: moved different block types

Reward:
- Accepted: $-\Delta \text{cost}$
- Rejected: 0

Agent:
- Estimates value of actions
- Selects action to take
Estimating Action Values

- Values of action are not stationary!

Agent determines ‘good’ move types online!
Action Selection: Exploration vs Exploitation

- \( \varepsilon \)-greedy: Mostly greedy (exploit), occasionally random (explore)
- \( \varepsilon \): fraction of exploratory moves

![Graph showing wirelength vs place time with \( \varepsilon \) values from 0.00 to 0.90.](image)

- No exploration harms quality
- Exploit to save run-time
Quality/Run-time Comparison

- VTR Benchmarks (10K-165K primitives), 3 seeds

![Graph showing Quality/Run-time Comparison for VTR benchmarks with two lines: VTR Default and VTR 8 (hand tuned).]
Quality/Run-time Comparison

- VTR Benchmarks (10K-165K primitives), 3 seeds
Quality/Run-time Comparison

- VTR Benchmarks (10K-165K primitives), 3 seeds
Quality/Run-time Comparison

- VTR Benchmarks (10K-165K primitives), 3 seeds

![Graph showing Quality/Run-time Comparison with 50% and 20% faster labels.]
Conclusion

• RL-enhanced Simulated Annealing based FPGA Placer
• RL agent controlled move generator
• Learns on-line what types of moves are productive
• Improves run-time/quality trade-offs
  • Particularly at low run-times
• Much more to explore
  • More diverse placement perturbations / moves
  • More powerful agents
Summary
Summary

• Open-source, flexible CAD crucial for
  • Evolution/innovation in spatial hardware
  • Enabling CAD research by the community
• VTR
  • Large, very active open-source effort
    • >12,000 commits, ~100 committers
  • Linked to other open-source efforts like Symbiflow, OpenFPGA
    • Mix and match your flow
  • Flexible, with good (but can be better!) result quality
  • Contributions welcome!

Open source enables innovation across a wider community
Thanks!

Questions?
Email: vaughn@eecg.utoronto.ca

VTR: verilogtorouting.org