VPR Version 5.0

Jason Luu, Peter Jamieson, Ian Kuon and Jonathan Rose
Motivation

- Vpack and VPR are two CAD tools widely used in the exploration of FPGA architecture and CAD
  - Among the key outcomes of Vaughn Betz’ Ph.D. thesis, 1999
  - Downloaded > 1000 times by Universities
  - > 200 times by Companies (stats as of 2003)
  - Ubiquitous: VPR often an uncited noun in papers

- The goals of VPR
  - Flexible description/exploration of FPGA architecture
  - Platform for CAD Tool algorithm exploration/development
  - Packing, Placement and Routing of FPGAs
The Architecture Exploration Loop

1. FPGA Architecture
2. Circuits
3. CAD Flow
4. Quality of Architecture
What does VPR currently do?

- Models homogeneous array of soft logic
- Bi-directional routing
- Extensive flexibility and generation of different routing architectures
- Global Router
- Nice graphics
- First-in-class modeling of speed and area
- Timing analysis
Goals for Update & Renewed Effort

- Model modern features in real architectures
- Enable exploration in future IC technologies
- Create a full flow to enable use of larger benchmark circuits
  - Currently architecture research based on VPR is stuck using small benchmarks - larger ones have heterogeneity – RAM, multipliers, etc.
  - Need a complete flow that understands HDL and heterogeneity.

- Detailed Exploration of Coarse-Grained Architectures
  - With mixture of fine-grained, or not;
New Features of VPR 5.0
Four New Features

1. Single Driver Routing Architecture
   - Unidirectional/Direct Drive; dominates
   - [Virtex98] [Lewis03] [Lemieux04]

2. Heterogeneity
   - Can model different blocks types

3. Wide Selection of Architecture Files
   - Transistor-Level Design Optimized;
   - different Area/Speed Trade-offs
   - IC process down to 22nm, based on PTM

4. Regression Test Suite
   - To maintain robustness
Single Driver Routing Architecture
Single Driver vs. Multiple Driver Routing

- **Bi-directional routing**
  - Tri-state buffers and pass transistors

- **Single-driver routing**
  - One driver, fan-in to multiplexer
Single Driver Routing Architecture

- Single Driver Dominates Multi-Driver
  - Lewis et al. 2003: single-driver dominates multi-driver
  - Lemieux et al. 2004: 25% area improvement, 9% delay improvement vs. multi-driver

- Used in industry for years
- Important that whole research community uses!
Switch Pattern Generation

- Routing Architecture Generation requires high quality patterns of switches
- Problem: Achieve best routability with given number of switches, and meet architecture specification
- Issue in past but now more restrictions with single-drivers
Switch Pattern Generation Issues

1. Meeting the parameter spec – Fs, Fc, varying L
   - Deal with quantization and generate good architectures

2. Switch box patterns
   - Tileable (one same switch box design throughout)?
   - Tileability vs mux balancing, are these two conflicting?
   - What is an appropriate pattern for single-driver architectures?

3. Routability – What definition to use?
   - For fixed W, highest % circuits route
   - Average lowest W for a set of circuits with the pattern algorithm
Simple Experiment: Single vs. Multi Driver

- Repeat experiment done in both industry and academia but with this publicly available VPR 5.0
- Compared two FPGAs:
  1. All multi-driver, length 4 tracks,
  2. All single-driver, length 4 tracks,

- Measured Minimum Channel Width
Simple Experiment: Single vs. Multi Driver

- Average 11% minimum channel width increase!
- Lewis et al. 10%, Lemieux 0%

![Bar Chart]

Circuit: alu, apex2, apex4, bigkey, clima, des, diffeq, dsip, elliptic, ex1010, ex5p, frisc, misex3, pdc, s298, s38417, s38548, s38584, s38941, bes, aids, tsg
Heterogenous Logic Blocks
### Heterogeneity

- VPR 5.0 supports heterogeneous blocks
  - Examples: Multipliers, Block RAM, Crossbars

- Architectural Specification of Hard Block
  - Column based
  - Each block has parameterized (multi-row) height
  - Transparent routing
  - Can specify all input-output timing paths of block
  - Allow combinational or registered outputs
  - All other parameters same as soft cluster
Heterogeneity Example Now Working
Heterogeneous Packers

- Illuminates need for packers for any new hard block
  - Packers: **Logical to Physical Translation**
  - Need timing and other optimizations

- For example
  - Current flow has simple packer at front end synthesis
  - Treats black boxes as primary I/O’s, not aware of depth

(18)
New FPGA Architecture Input Format

- Key to architecture exploration: a language to describe FPGA architecture
- Architecture specification intrinsically hierarchical
- VPR 5.0 uses XML to leverage its inherent hierarchy
  - Parsers easy to get; old VPR parser kinda rough
  - Easy to extend language
Sample 1: Spec of Single Driver Length 4

<segmentlist>
  <segment type="unidir" length="4" freq="1"
    Rmetal="44.06455" Cmetal="1.72786e-13">  
    <mux name="normal" />
    <sb type="pattern">1 0 1 0 1</sb>
    <cb type="pattern">1 0 0 1</cb>
  </segment>
  <segment ...> ... </segment> ...
</segmentlist>
Sample 2: Heterogenous Block

<type name=".mult" height="2">
  <subblocks max_subblocks="1" max_subblock_inputs="8"
    max_subblock_outputs="8">
    <timing>
      [Timing Matrix]
    </timing>
    <fc_in type="frac">0.25</fc_in>
    <fc_out type="full" />
    <pinclasses>
      <class type="in">[pin numbers]</class>
      <class type="out">[pin numbers]</class>
      <class type="global">[pin numbers]</class>
    </pinclasses>
  </subblocks>
</type>
Wide Selection of Optimized Architecture Files
Architecture Files

- Timing and Area for routing and logic needed for architecture file

- Architecture (L, Fs, N, I, etc) can vary speed and area since transistor level implementation may change
Optimized Timing and Area Models

- Betz took ~ 2 months to create “the” area-delay optimized architecture file in 350nm; so did Ahmed
- Previously, NDA’s prevented release of accurate timing models

Goal:
- Provide publishable optimized timing and area models for a large number of designs
Creating Timing and Area Models

- Use predictive technology models (PTM) from Yu Cao at Arizona State
  - 180 nm to 22 nm CMOS
  - Not as accurate as foundry models but publishable!

- Developed custom automatic transistor sizing tool
  - Easily create optimized designs for a range of architectures
  - Adds new dimension for exploration
    - Circuit Design Objective
      - Area, Delay, or Area^kDelay^n

(25)
Experiments Possible with New Models

- **Technology scaling**

- **Trade-offs between design objective and process technologies**
Architecture Repository

- Releasing repository of selected designs with
  - Varied architecture parameters (K, N, L, …)
  - Varied design objective (areadelay, delay, …)
  - Varied technology (22 nm CMOS → 180 nm CMOS)

- Timing and area provided in the new VPR architecture format
Robustness
Robustness

- Regression test infrastructure for VPR
  - Scripts to run a suite of tests on VPR
  - Each test runs a test script on list of circuits and architectures
  - Extracts results with regular expressions and compares with golden data allowing for specifiable range of deviation
Regression Tests:

- Check-in regression: Quick tests, varying coverage
- N-K sweep: All combinations of $K = 2..7$, $N = 1..12$
- QoR: Quality of Results over 20 largest MCNC circuits
- Architecture sweep: Sweep randomly generated architectures
- Options sweep: Sweep all options
- Useful check for correctness!
New: Full CAD Flow from HDL
Now Support a Full CAD Flow:

- Heterogeneous block inclusion on FPGA architectures requires upstream CAD support

- In this package we support designs created in:
  - Verilog HDL (subset)
  - BLIF (subset)
Our Full Flow

- **HDL Front End** (ODIN/Toronto)
- **Logic Synthesis** (ABC/Berkeley)
- **Packing** (Vpack/Toronto)
- **Placement & Routing** (VPR/Toronto)

- **Input:** Verilog HDL designs
- **Output:** Placed and Routed design on specified FPGA
  - explore a wide range of FPGA architectures
Heterogeneity representation in flow

- Front end synthesizer needs to identify heterogeneous blocks and potentially pack them
  - Currently identifies and packs hard multipliers
  - Output is in BLIF format with heterogeneity represented as connected black boxes

- Logic synthesis, technology mapping, and clustering representation of heterogeneous blocks is black boxes
  - Inputs to black box are like Primary Outputs
  - Outputs to black box are like Primary Inputs
module mult_design (a, b, c, o1, o2, clk)
input [1:0]a, [1:0]b, [1:0]c, clk;
output [3:0]o;

always @(posedge clk)
begin
    o0 <= a*b;
    o1 <= c*b;
end
endmodule
BLIF format

```
.model mult_design
.inputs a_0 a_1 b_0 b_1 c_0 c_1 clk
.outputs o0_0 o0_1 o0_2 o0_3 o1_0 o1_1 o1_2 o1_3
.sublk mult2 in0=a_0 in1=a_1 in2=b_0 in3=b_1 out0=o0_0
       out1=o0_1  out2=o0_2 out3=o0_3
.sublk mult2 in0=c_0 in1=c_1 in2=b_0 in3=b_1 out0=o1_0
       out1=o1_1  out2=o1_2 out3=o1_3
.end

.model mult2
.inputs in_0 in_1 in_2 in_3
.outputs out_0 out_1 out_2 out_3
.blackbox
.end
```
.global clk
.input a_0
pinlist: a_0
...
.output o_0
pinlist: o_0
...
.mult2 mult2_0
pinlist: a_0 a_1 b_0 b_1 o0_0 o0_1 o0_2 o0_3
subblock: 0 1 2 3 4 5 6 7
Timing Analysis in CAD flow

- This is not a full timing driven flow

- **Not fully timing driven:** Logic synthesis, technology mapping, and clustering
  - Paths through heterogeneous blocks are analyzed the same as going through registers

- **Timing driven:** Placement and routing
  - All paths have timing values
**Hacks in CAD flow (skip; for info only)**

Scripts fix link between tech-mapping and clustering:

1. Global script included in package to reconnect clock signals to registers (not maintained through ABC)
   - Doesn’t support multi clock designs

2. Two scripts to fix simple logic synthesis operations not performed on specific designs
   - Script fixes sv_chip0
     - LUT with two of the same input signals
   - Script fixes sv_chip1
     - Clocked constant that should have been synthesized away
Features for Future Versions of VPR and New Full Flow
Future Features

- Full Power Modeling
- Bus-Based Routing
- Complete Timing Driven Flow including Heterogeneity
- Packing Algorithms for New Heterogeneous Blocks
- Enable Research on Routing Pattern Generation for Single-Driver Architectures
- Carry Chains
- More Complete flow from Verilog -> Routing
  - Have a version of ODIN -> ABC -> VPR prototyped
More Future Features

- Selectable registered inputs and outputs for logic blocks
- Depopulation of logic clusters
- Routing rotations for LUTs
- Direct supply ratio specification
- Tileable switch block pattern
  - Mux balancing (done)
  - Tileable quantization (not done)
  - Tileable switch block (not done)
- Simulatable transistor-level output
- Show routing graph before and after placement
- Colour heterogeneous blocks
Acknowledgements

Many people have contributed to this work, and are working actively on it:

- Vaughn Betz;
- Sandy Marquardt
- Andy Ye
- Russ Tessier
- Mark Fang
- Jason Luu
- Peter Jamieson
- Ian Kuon
- Ted Campbell
- Danny Paladino
VPR development

- Andy Ye implemented single-drivers in VPR with variable L, F_{c_{out}}, F_{c_{in}}, and depopulation

- Wei Mark Fang added Fs flexibility and mux balancing
  - Mux balancing: making all resulting muxes roughly same size
Questions?

Beta Test Volunteers?