The VTR Project: Architecture and CAD for FPGAs from Verilog to Routing

J.Rose, J. Luu, C-W Yu, O. Densmore, J. Goeders,
A. Somerville, K. Kent, P. Jamieson and J. Anderson

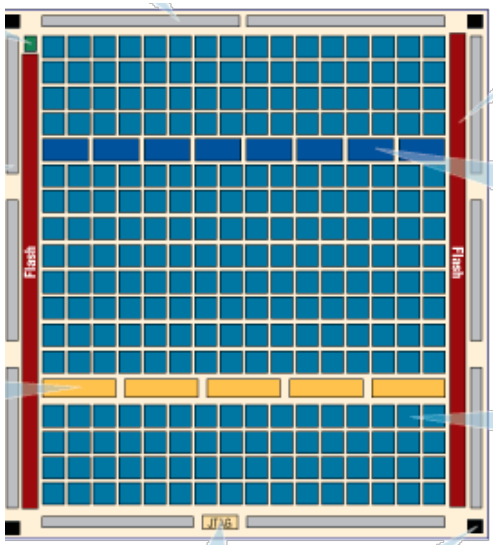
University of Toronto,
University of New Brunswick
City University of Hong Kong
Miami University

Verilog-To-Routing (VTR) Project Goal

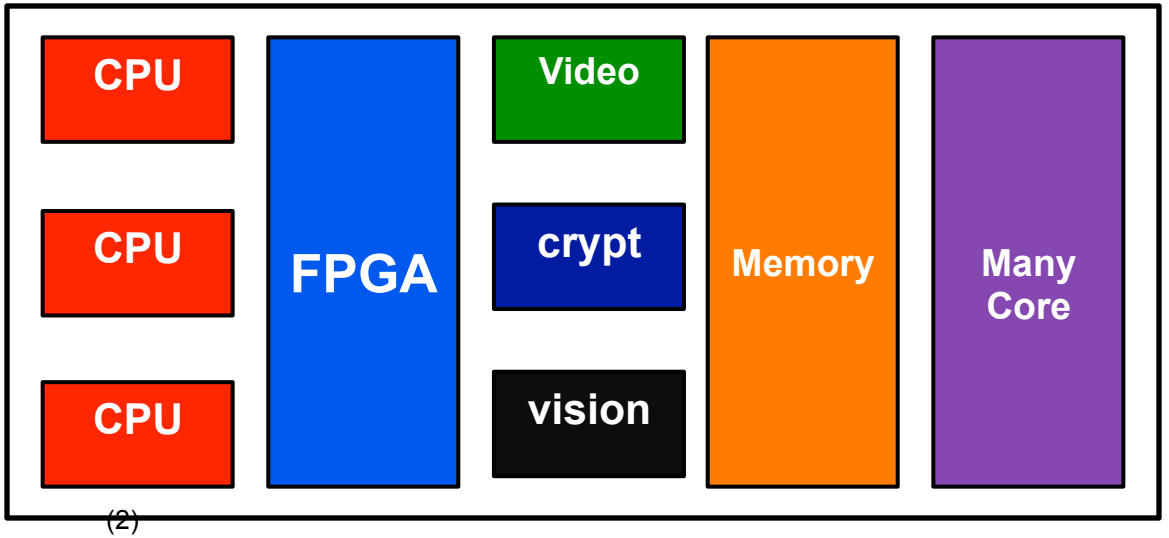
Goal is to Enable the Exploration of:

1. Hypothetical FPGA Architectures and
2. New CAD Algorithms for FPGAs

Pure-Play FPGA



FPGA Embedded in SoC



Objectives: A 'Super-VPR'

1. Like VPR, but a larger CAD flow, starting at **HDL** level
   - Serve as a basis for even longer flow, above HDL synthesis

2. Accurately model sophisticated, modern, hypothetical FPGA architectures
   - With heterogeneity, hierarchy, modes, complex clocking
   - Coarse Grained architectures
   - Energy, speed and area modeling, based on electrical design
   - Model variation, reliability and other process effects

Objectives, continued

3. Provide platform for FPGA CAD algorithm research
   - A complete flow that you can easily change part of
   - Be a basis of comparison

4. Be **Robust**
   - quality software that can be used for many purposes, reliably

5. Provide large realistic benchmarks that work in flow
   - As important as the software itself

This is Scary Big

- Big FPGA companies have 100s of people doing this
  - Small ones have a dozen or more
- Commercial flows don't support the exploration of hypothetical FPGA architectures
  - Described using an Architecture Description Language
  - Harder than targeting known devices!
  - Sotware must work on all architectures that can be conceived and described

This is too big for a single academic enterprise

VTR: A World-Wide Collaboration

- **Toronto**: Jason Luu, Jason Anderson, Vaughn Betz, Opal Densmore, Cong Wang, Peter Milankov
- **New Brunswick:** Ken Kent, Ash Furrow, Paddy O'Brien, Joey Libby, Shubham Jain, Konstantin Nasartschuk, Andrew Somerville
- **Penn:** Rafi Rubin
- **Miami Ohio:** Peter Jamieson
- **City University of Hong Kong:** Chi Wai Yu
- **UBC:** Jeff Goeders, Eddie Hung
- **Texas Instruments:** Joyce Kwong, Jeff Rudolph
- **Altera:** David Grant and Mark Jarvin

Key: Weekly Meeting on G+ Hangout

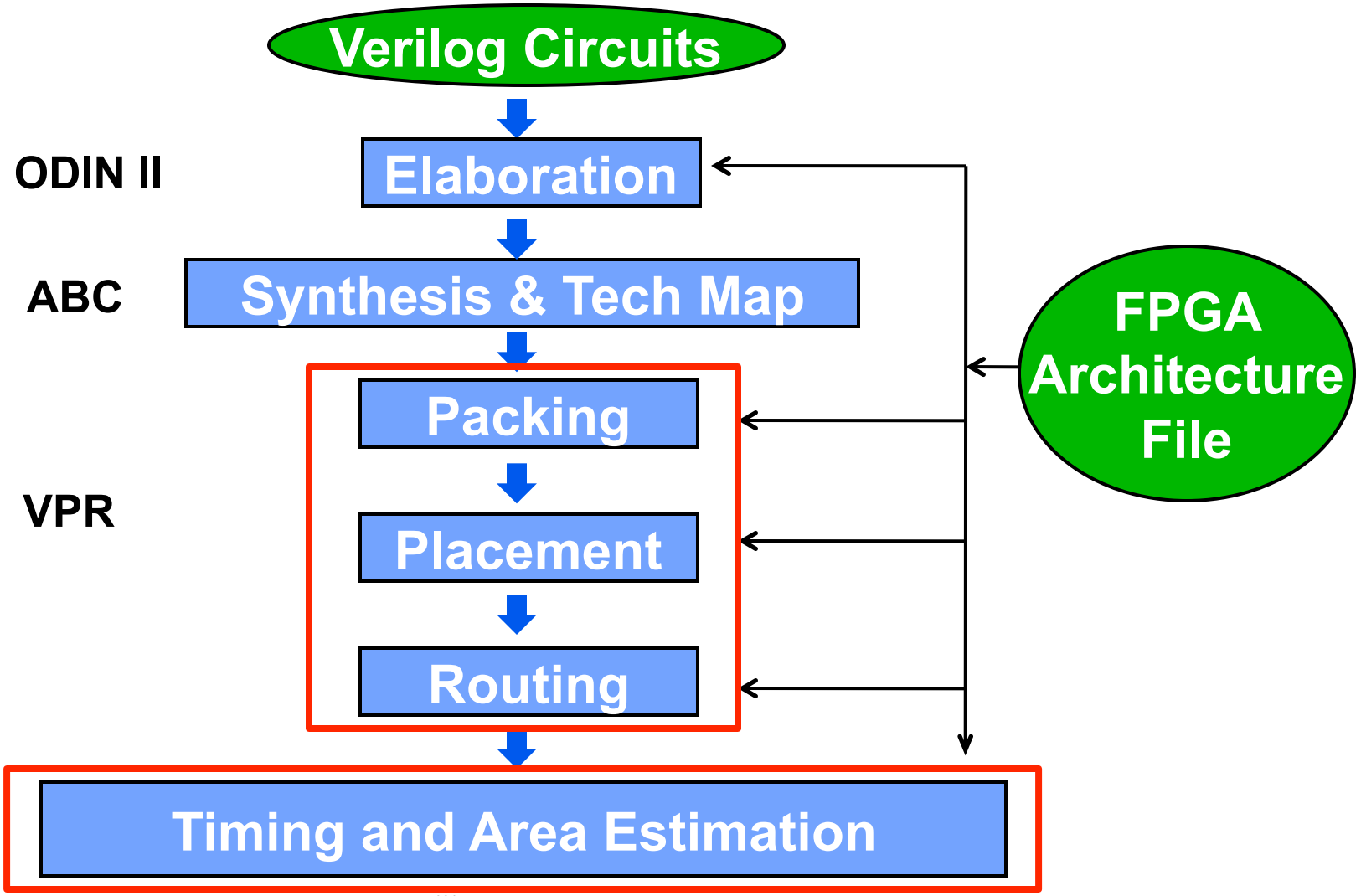
Jonathan Rose - Jan 26, 2012 (edited) - Hangout - Limited

Jonathan Rose hung out with 5 people.

Jonathan Rose - Feb 2, 2012 (edited) - Hangout - Limited

Jonathan Rose hung out with 5 people.

Big Picture of The Flow



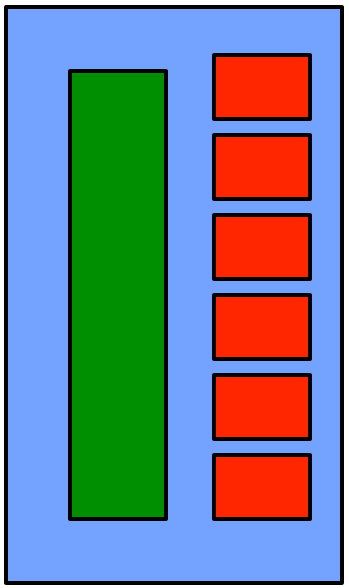
Previous Version of VTR: 1.0 Alpha

- Released 1 Year Ago
  - ODIN II, ABC, VPR 6.0 ‘alpha’
  - Contains VPR with merged packer
- New Logic Block architecture description capability
  - Far more complex logic block architectures can be described
  - And associated packer

Context: Prior Logic Blocks Were

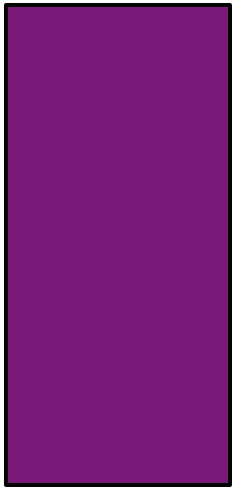
Either:

1. A fully connected cluster of Basic Logic Elements:



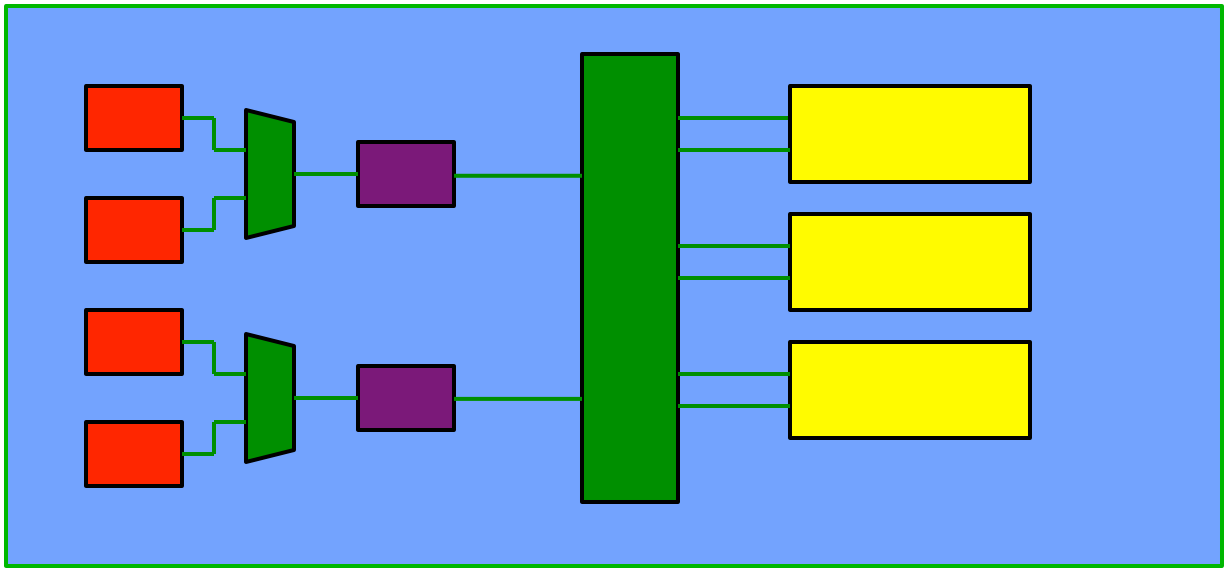
or

2. A monolithic block that had to be completely generated by the up-stream tools
   - A limited form of heterogeneity

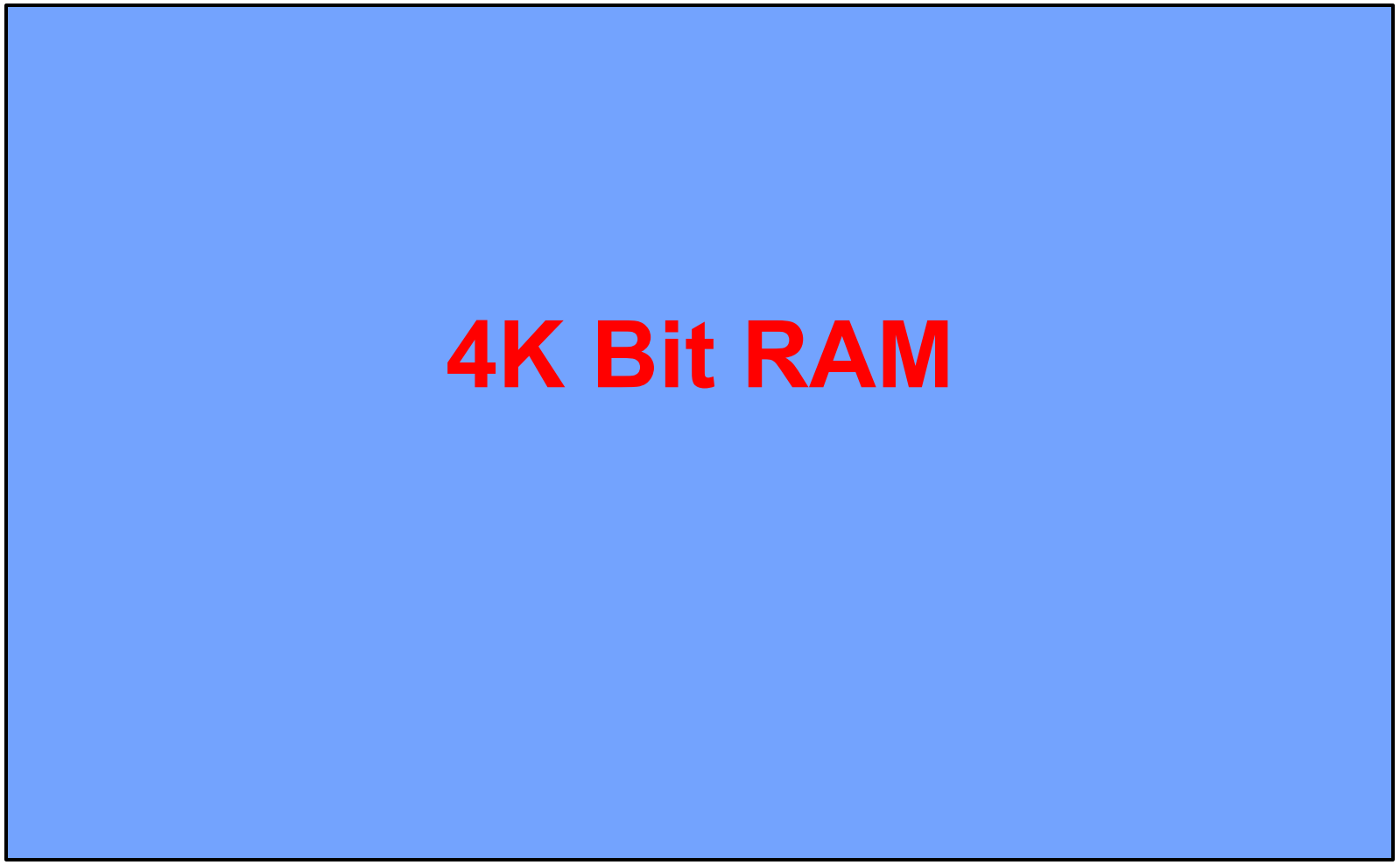


With VPR 6.0 we added Complex Blocks

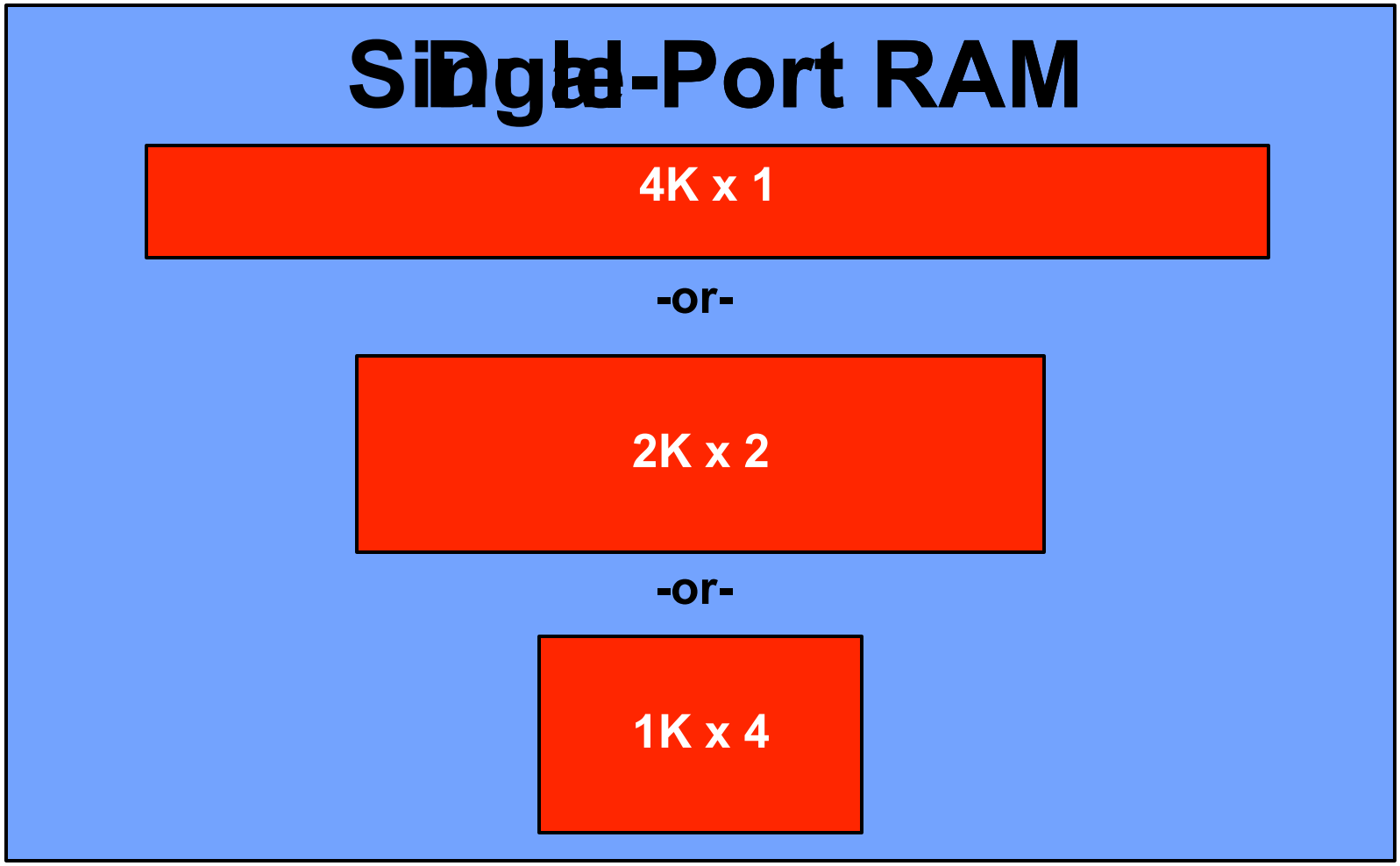
- Can describe essentially arbitrary netlist of primitives within the block
  - Primitives can be connected directly, through muxes or crossbars
  - Encased in any number of levels of hierarchy
  - Each level of hierarchy can be set to different modes, which would be set by configuration bits



Also Memories and Multipliers & Modes

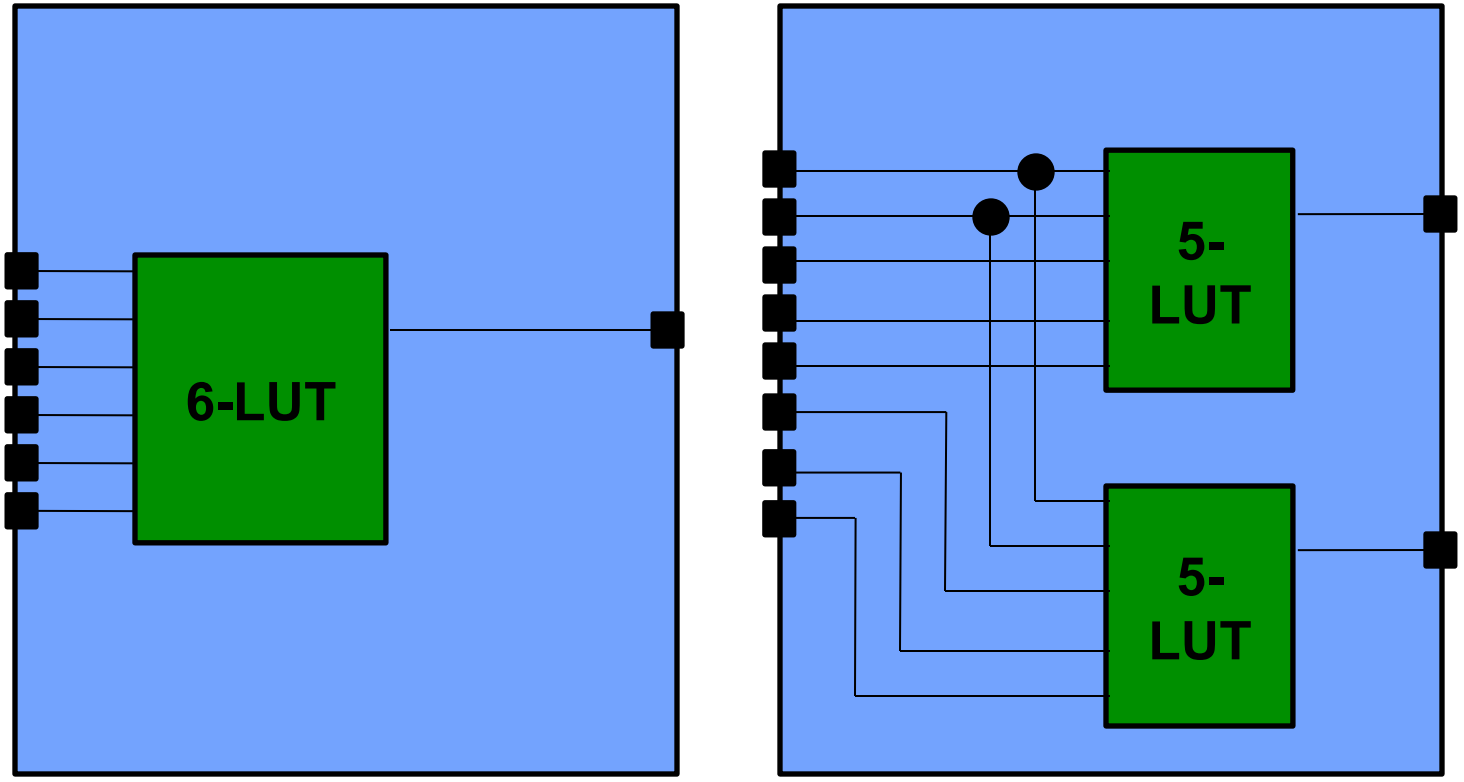


Also Memories and Multipliers & Modes



Allows Fracturable LUTs

Either One 6-LUT: - or - Two 5-LUTs with sharing:

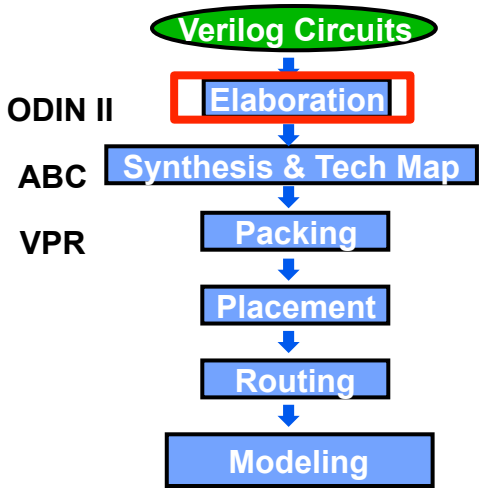


Gave Rise to New & Hard Packing Problem

- **Given:** the netlist of logical primitives and the Complex block which can receive those primitives
- **Determine:** where to pack the logical primitives so that:
  - Everything can be connected within and without the block
  - So that optimization goals of area & speed are met
- Last year we presented a basic packing algorithm that could handle the hierarchy, modes and interconnect
  - But it wasn't timing driven
  - Nor was that version of VPR, because timing analysis was broken by the complexity of the block

The New Release: VTR 1.0 Full

ODIN II: Verilog Elaboration



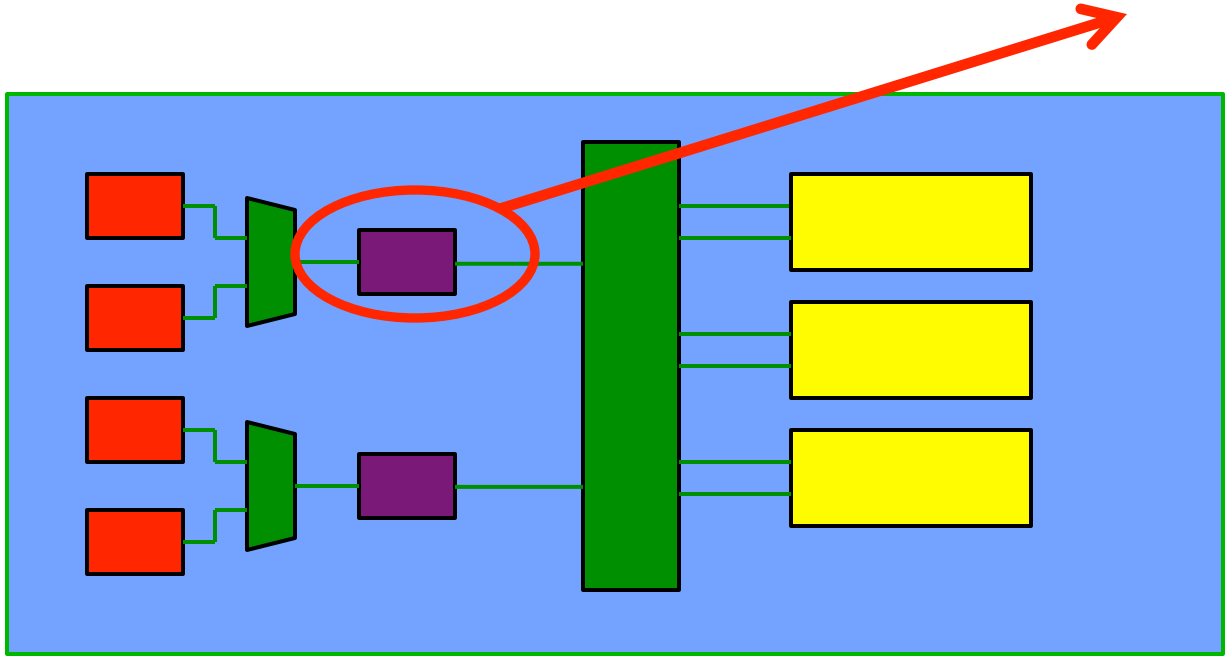
ODIN II – Verilog Elaboration

- Input:
  - Verilog HDL-level circuit
  - FPGA Architecture Description File
    - Describes any non-standard logical primitives
- Output: BLIF netlist
  - Soft Logic LUTs
  - Properly split Multipliers & Memories
  - Instantiated primitives that are parts of new Complex Blocks

Architecture File & ODIN II

- ODIN II reads the same architecture file as VPR
  - There is a separate section that describes the primitves

- For example, the purple block below, call it a '**Flurch**'



The Primitive is Declared in Arch File

- Like this:

```
<model name="Flurch">
      <input_ports>
          <port name="fin1"/>
          <port name="fin2"/>
      </input_ports>
      <output_ports>
          <port name="fout1"/>
          <port name="fout2"/>
      </output_ports>
  </model>
```

Also Describe the Physical Structure

```
<pb_type name=”Flurch_in_Block"
blif_model=".subckt Flurch” num_pb="1">

   <input name=”fin1" num_pins=“1” />
   <input name=”fin2" num_pins=“1” />
   <output name=”fout1" num_pins=“1” />
   <output name=”fout2" num_pins=“1” />

</pb_type>
```

Instantiation in User's Verilog Code

```
...
     input d1,d2;
     wire fd1, fd2;
...
     Flurch My_Flurch(d1, d2, fd1, fd2);


...
```

ODIN II Produces output BLIF

```
.model Test_Flurch
.inputs d1, d2
.outputs fd1, fd2

.subckt Flurch fin1=d1 fin2=d2 fout1=fd1
fout2=fd2

.names top.flurch+My_Flurch^fout1 top^fd1 …
1 1
```

New Features of ODIN II

- Simulation engine for Verification of tool flow
  - ODIN now contains a built-in simulator that can check if output is correct
  - Check Verilog vs. BLIF produced by other tool
  - Ultimately will use (not yet) to check logical correctness of downstream flow
- Can accept test vectors or generate random test vectors

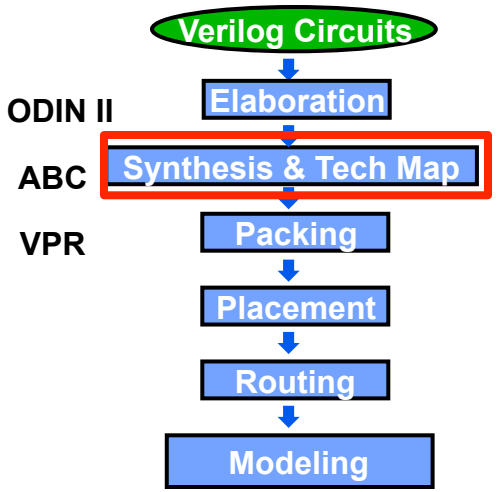
New Features of ODIN II

1. Enhanced Language Coverage:
   - Support for generic hard blocks

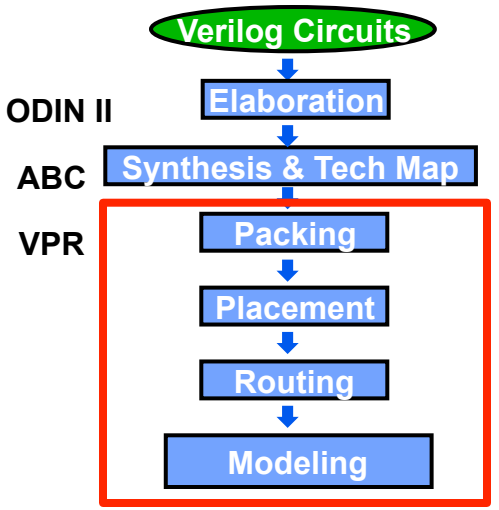
2. Macro pre-processor now permits:
   - **`ifdef, define, else, endif and include`**
   - Has improved ability to port new benchmarks to ODIN II

3. Ongoing bug fixing in response to benchmark processing

ABC: Logic Synthesis

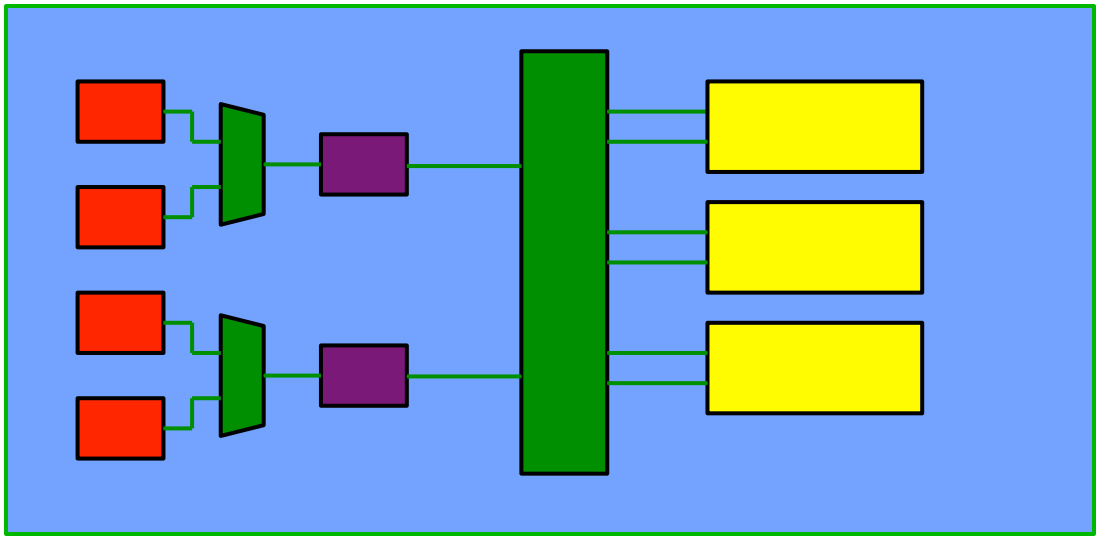


VPR: Packing, Placement & Routing



VPR 6.0 Full Key Feature: Timing-Driven!

- Can now do timing analysis inside complex blocks
  - Needed to build the timing graph of arbitrary structure inside the logic block
  - New version of architecture file now includes ability to specify timing of primitives that make up blocks
  - Specify delay, set-up, clk-to-q, and max operating frequency



Timing-Driven Packer

- Previous version of packer was only area-driven
- New algorithm is now timing-driven
  - as good as old timing-driven packer on classical LUT-cluster architectures
  - Can do much more complex architectures

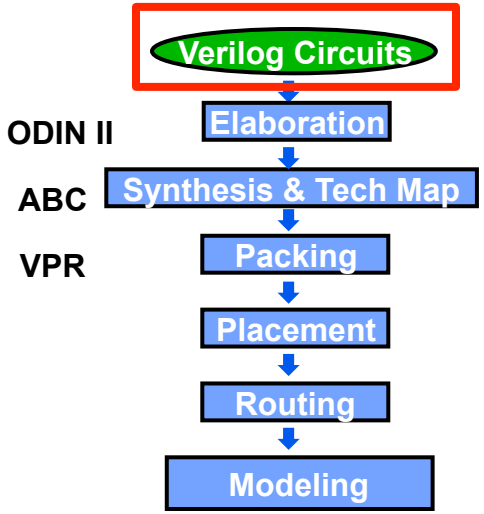
- Basic algorithm is greedy: select candidate & test
- Test for feasibility of primitive in block includes **routing**
- Packer is very general, but quite slow
  - working on speed now

Timing-Driven Placement & Routing

- Returned with the advent of the timing analysis, no other change needed

- Numerous other enhancements
  - Sped up routing again, still needs work
  - Cleaned up memory usage

Benchmark Circuits



Benchmarking Goal: Go larger!

- One crucial reason for adding ability to process memories and multipliers is that all modern large circuits include these structures
- With this ability, we can now include much larger circuits in our flow
- **Key Realization**: Benchmarks must be released, like software, alongside the version of the software that they work with!

Benchmark Processing: Large Circuits

One of the new circuits:

- 'MCML'
  - Previous UofT research project on Photo-Dynamic Therapy for Cancer Treatment
  - 95,890 6-LUTs
  - 53,280 Flip-Flops
  - 6,343 x 10 Fracturable 6-LUT Clusters
  - 10 Memories
  - 30 Multipliers

- This and one other circuits took 1 person-summer
  - Verified through simulation after porting
  - Needed to develop/port math cores

Benchmarks Run Through Flow

- Previously released circuits have been updated to work with new flow, now 19 total.
  - Added new circuits as above

- Example run through 40nm CMOS FPGA architecture:
  - Fracturable 6-input LUTs as basic soft logic primitive
  - Soft cluster contains 10 Fracturable 6-LUTs
  - 144K bit Dual Port RAM
  - 36x36 Multipliers, Fracturable to 18x18 or 9x9

Benchmarks – Size Data

Circuit	# Inputs	# Outputs	#6-LUTs	# Flip-Flops	# Memories	# Mults	Min W
bgm	257	32	42065	6100	0	11	152
blob_merge	36	100	6002	445	0	0	92
boundtop	271	192	2907	1667	1	0	72
ch_intrinsics	99	130	416	233	1	0	50
diffeq1	162	96	470	193	0	5	64
diffeq2	66	96	299	96	0	5	60
LU8PEEng	114	102	18950	6627	9	8	132
LU32PEEng	114	102	67543	20893	9	32	208
mcml	36	33	95890	53280	10	30	166
mkDelayWorker32B	511	553	5069	2459	9	0	92
mkPktMerge	311	156	234	36	3	0	48
mkSMAdapter4B	195	205	1731	983	3	0	74
or1200	367	394	2447	612	2	1	86
raygentop	239	305	2070	1423	1	18	78
sha	38	36	1140	431	0	0	58
stereovision0	157	197	11494	13405	0	0	78
stereovision1	132	145	10020	11789	0	152	120
stereovision2	149	182	29977	18416	0	564	196
stereovision3	10	30	205	102	0	0	34

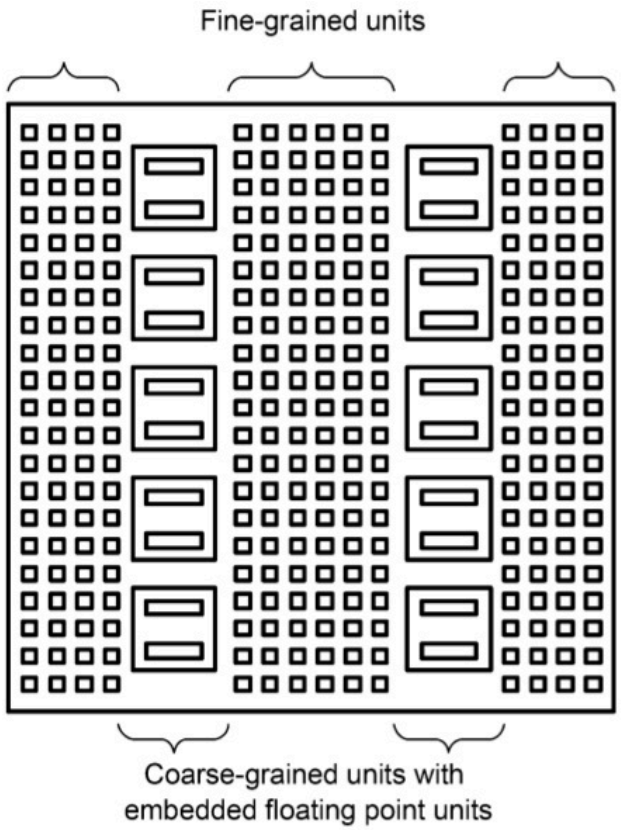
Benchmark Run: Timing vs. Non-Timing

Circuit	# CLBs	Timing-Driven Critical Path (s)	Non-Timing Critical Path (s)	Ratio
bgm	3933	1.82E-08	2.17E-08	1.19
blob_merge	510	5.13E-09	6.48E-09	1.26
boundtop	222	5.42E-09	6.77E-09	1.25
ch_intrinsics	35	2.77E-09	2.84E-09	1.02
diffeq1	37	1.33E-08	1.43E-08	1.08
diffeq2	25	1.11E-08	1.21E-08	1.09
LU8PEEng	1901	4.68E-08	5.64E-08	1.21
LU32PEEng	6790	4.67E-08	6.55E-08	1.40
mcml	6343	6.44E-08	6.78E-08	1.05
mkDelayWorker32B	405	6.42E-09	7.11E-09	1.11
mkPktMerge	16	3.69E-09	3.69E-09	1.00
mkSMAdapter4B	144	5.29E-09	6.06E-09	1.15
or1200	204	1.13E-08	1.67E-08	1.48
raygentop	168	3.88E-09	4.51E-09	1.16
sha	100	1.04E-08	1.27E-08	1.23
stereovision0	828	3.32E-09	4.70E-09	1.42
stereovision1	854	3.63E-09	5.33E-09	1.47
stereovision2	2381	1.31E-08	1.85E-08	1.41
stereovision3	15	2.97E-09	3.31E-09	1.12
Geomean				**1.21**

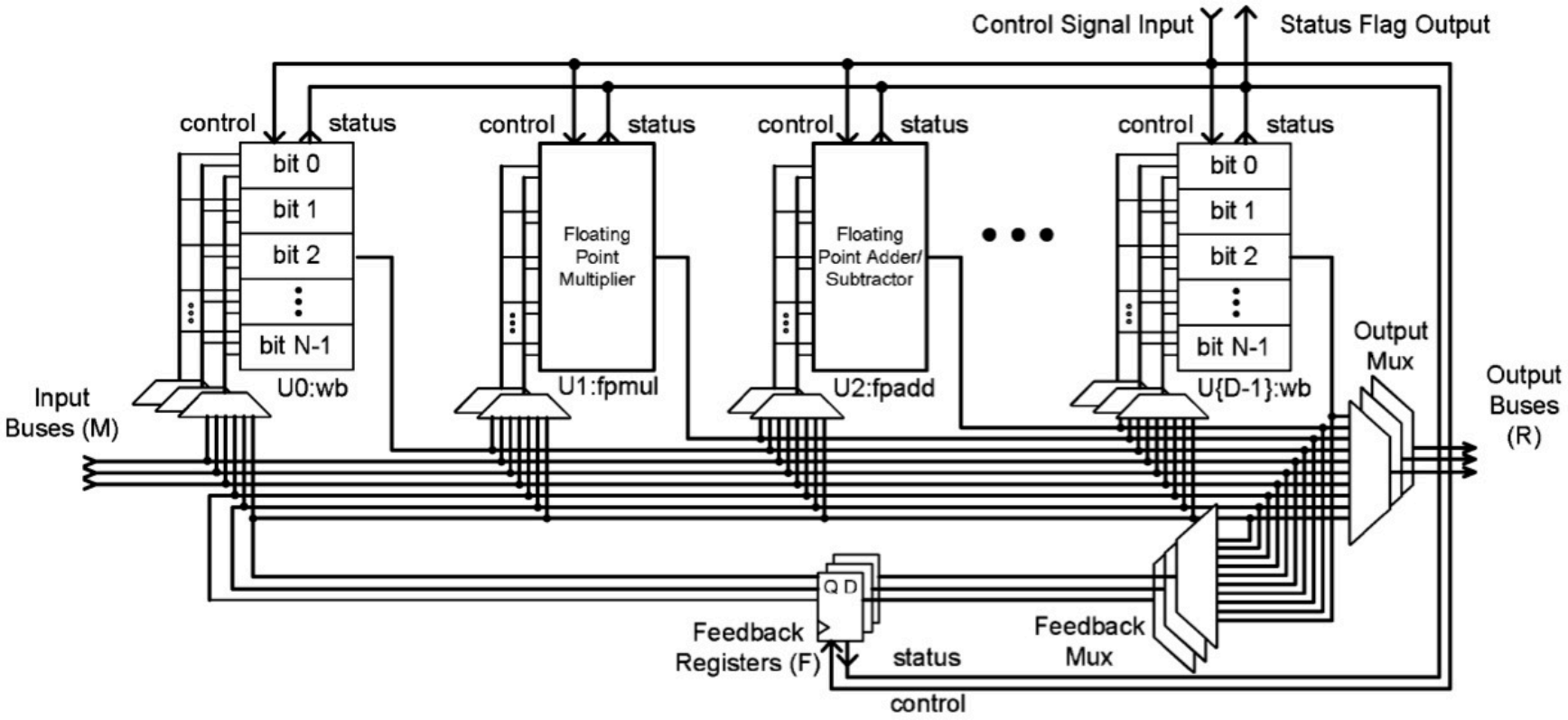
Example Use of VTR Flow in FPGA Architecture Exploration

Example Use by Architect

- Chi Wai Yu Ph.D. thesis at Imperial was to architect a floating point unit inside an FPGA:
  - He originally created a custom version of VPR to model and experiment with
- Can he use new VTR flow instead?
  - without any software modifications



The FPU Block



Success!

- Capturing the FPU shook out some bugs
- Were able to run original Verilog-based circuits through flow, and obtain measurements of speed and area

- Able to compare:

  FPGA with floating-point block vs. FPGA without

Speed of Soft FPU vs. Hard

Circuit	Soft Logic Delay (ns)	Hard FPU Delay (ns)	Ratio
bfly	36.1	2.99	12.1
bgm	35.7	2.99	11.9
dscg	36.6	2.99	12.2
fir	36.0	2.99	12.0
mm3	35.3	2.99	11.8
ode	34.5	2.99	11.5
syn2	37.6	2.99	12.6
syn7	50.3	2.99	16.8
GEOMEAN			**12.5**

- Original Research had ratio at 4 times
  - Soft logic FPGA had carry chains, which made soft faster
  - A key future work is to include special carry logic

Area of Soft FPU vs. Hard

Circuit	Soft Logic area (CLBs)	Hard FPU Area (Equiv CLBs)	Ratio
bfly	6405	264	24.3
bgm	16908	792	21.3
dscg	6371	440	14.5
fir	6215	352	17.7
mm3	4556	264	17.3
ode	3609	480	7.5
syn2	6553	264	24.8
syn7	39420	1584	24.9
GEOMEAN			**17.9**

- Original Research had ratio at 24 times
  - Is in the same ball park – different base FPGA

Effort to Capture New Architecture v. Old

- Original Research required roughly 1 person-year of work to modify VPR 4.3 to handle FPU
- This work required roughly 2 person-weeks, including VPR debug time.

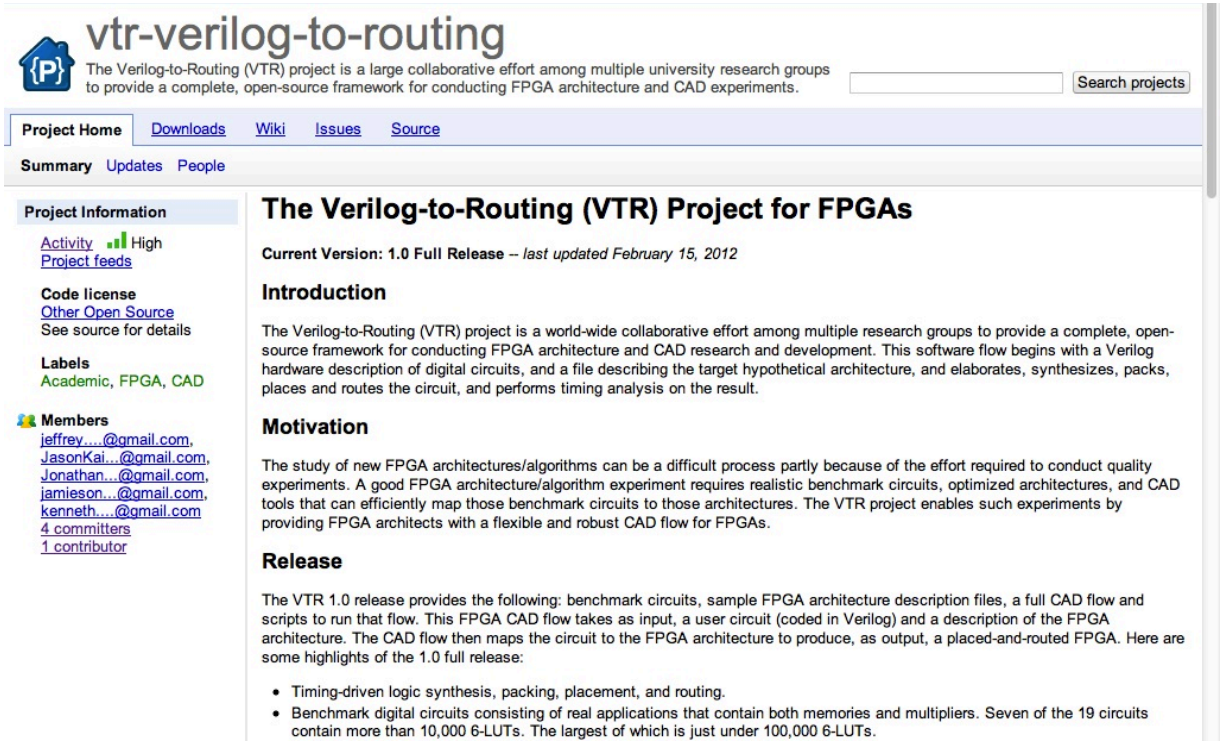
The Release: Last Week: VTR 1.0 Full

Location

- Original Location:
  - http://www.eecg.utoronto.ca/vtr
- Permanent, new Location, in Google Archive:
  - http://code.google.com/p/vtr-verilog-to-routing/

Google Archive Provides

- New Documentation and Tutorials for Whole Flow
- Issue Tracking – anyone can report a bug
- Software Development Trunk now public



License

NEW: VTR Fully Open Source

- ODIN II was already Open Source
- ABC (from Berkeley) was also Open Source

- **VPR is now, as of last week, fully Open Source**
  - No restrictions, MIT license
  - For VTR download, still do ask for name of user, but is not required to obtain software.

Future Work

Future VTR Work: Odin II

- Generate Carry Chain elements from arithmetic
- Support Verilog Parameters
- Infer Memory from 2D Arrays
- Continued improved language coverage
  - And general bug fixing

- Extend Simulation Verification to downstream flow steps

Future VTR Work: ABC

- Move to more recent version of ABC
  - Current version is old
- Find way to have it handle multiple clocks
  - Rather than remove clocks!
- Use and Test efficacy of White Blocks
  - Which permit optimization across fixed boundaries, both hard and soft

Future VTR Work: VPR Algorithms

- Better Packing Algorithms
  - Will attack pack time for common cases
  - Resort to routing only when absolutely necessary
  - Handle Carry Chains

- Placement
  - Algorithms that scale in modern world of slow processors
  - Handle Carry Chains

Future VTR Work: VPR Modeling

- Timing Analysis
  - Support multiple clocks
  - Hold time analysis
  - Support arrival time specification
- Area Modeling
  - Capture transistor-level design of complex blocks
  - Automated transistor-level optimization (sizing) of blocks
- Energy/Power Modeling

Future VTR Work: VPR Architecture

- Support Carry Chains
  - In architecture definition for intra-block and extra-block
- Bus-based routing for coarse-grained FPGAs
- Architecture of Clock spines/trees

VTR: A Long-Term Ongoing Project

- Trying to serve the community
- Weekly meeting via Google+ Video
  - Almost every Thursday at 11am Eastern Time
- **Let us know if you'd like to help!**