Warp Processors
(a.k.a. Self-Improving Configurable IC Platforms)

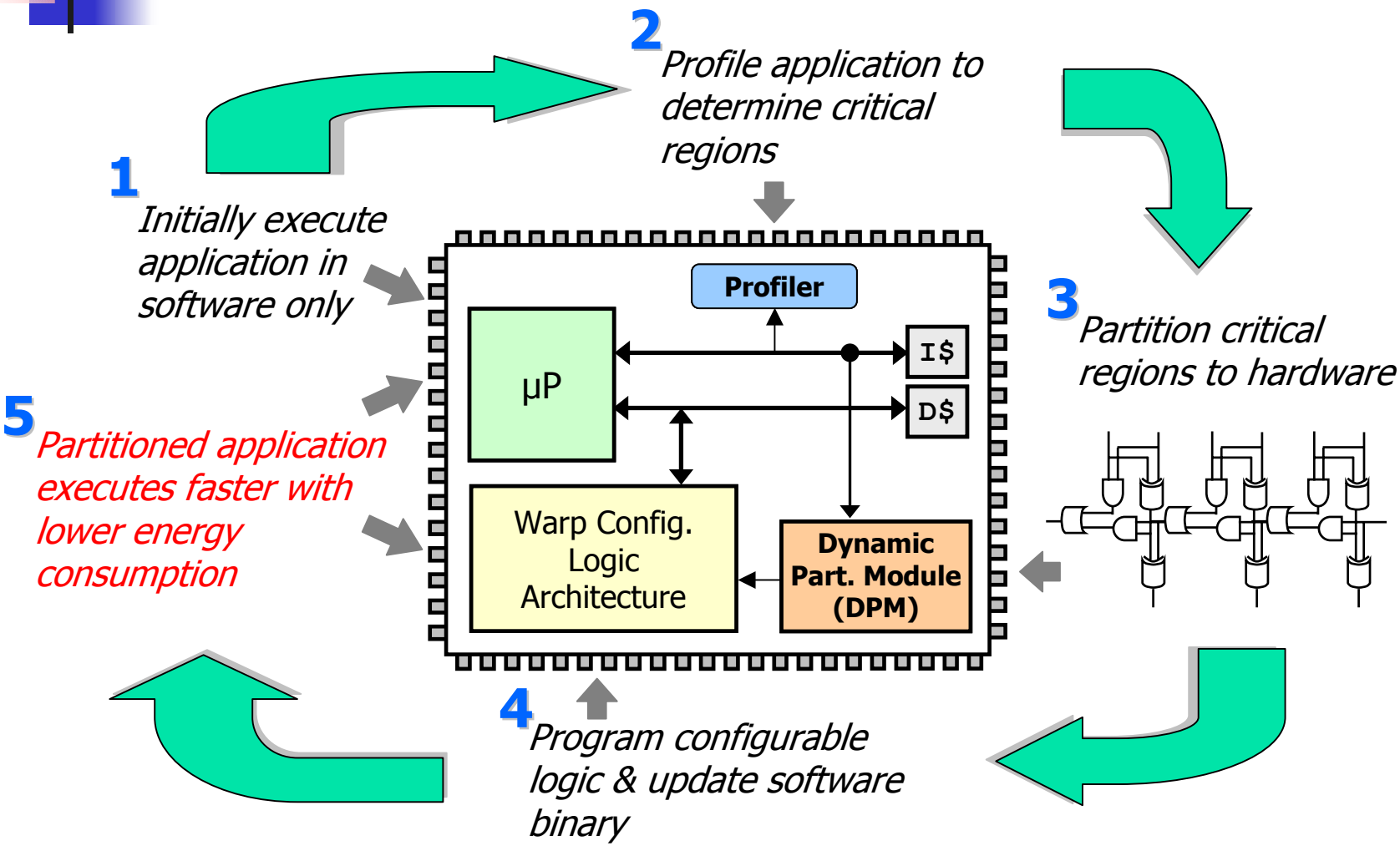
Frank Vahid
Department of Computer Science and Engineering
University of California, Riverside
Faculty member, Center for Embedded Computer Systems, UC Irvine

Ph.D. students: Roman Lysecky (grad. June 2004), Greg Stitt (grad. June 2005)
UCR collaborators: Prof. Walid Najjar, Prof. Sheldon Tan

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Introduction

Warp Processors – Dynamic HW/SW Partitioning



Introduction

Previous Dynamic Optimizations -- Translation

- Dynamic Binary Translation
  - Modern Pentium processors
    - Dynamically translate instructions onto underlying RISC architecture
  - Transmeta Crusoe & Efficeon
    - Dynamic code morphing
    - Translate x86 instructions to underlying VLIW processor
  - Just In Time (JIT) Compilation
    - Interpreted languages
    - Recompile code to native instructions
    - Java, Python, etc.

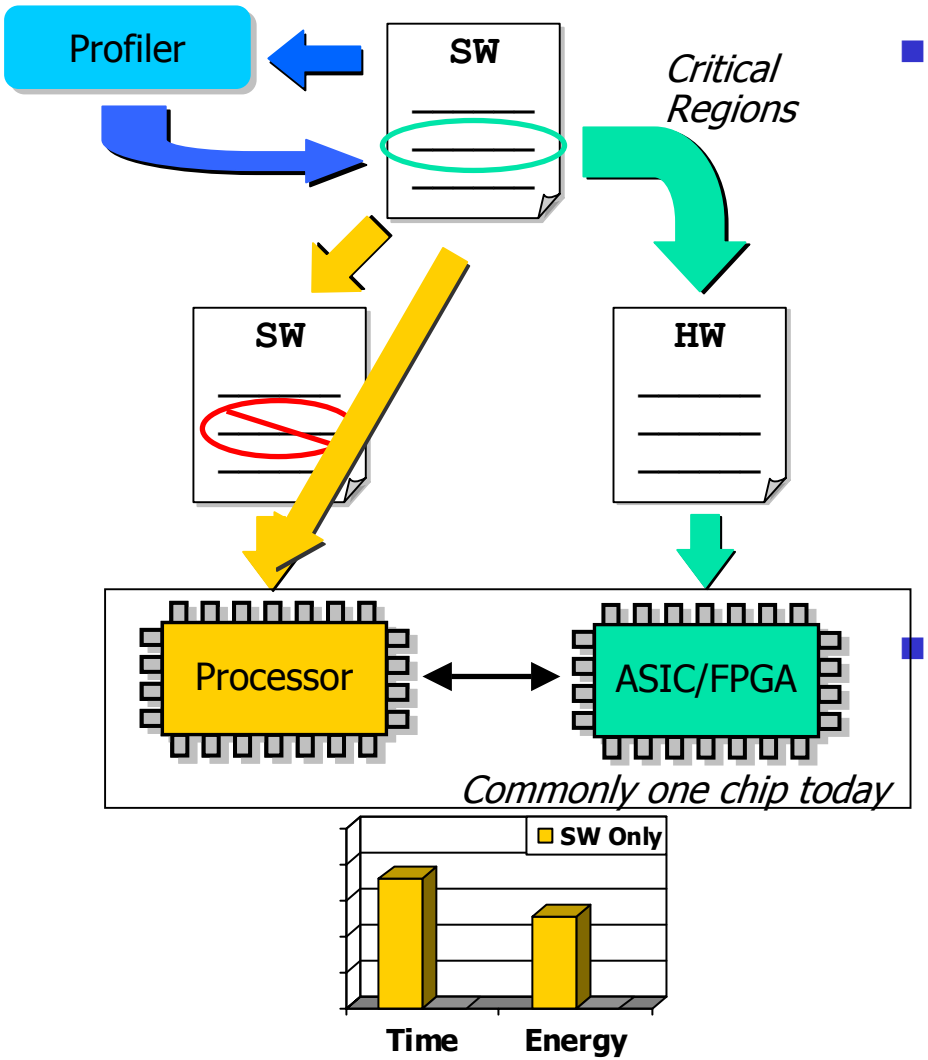
Introduction

Previous Dynamic Optimization -- Recompilation

- Dynamic optimizations are increasingly common
  - Dynamically recompile binary during execution
  - Dynamo *[Bala, et al., 2000]* - Dynamic software optimizations
    - Identify frequently executed code segments *(hotpaths)*
    - Recompile with higher optimization
  - BOA *[Gschwind, et al., 2000]* - Dynamic optimizer for Power PC
- Advantages
  - Transparent optimizations
    - No designer effort
    - No tool restrictions
  - Adapts to actual usage
  - Speedups of up to 20%-30% -- **1.3X**

Introduction

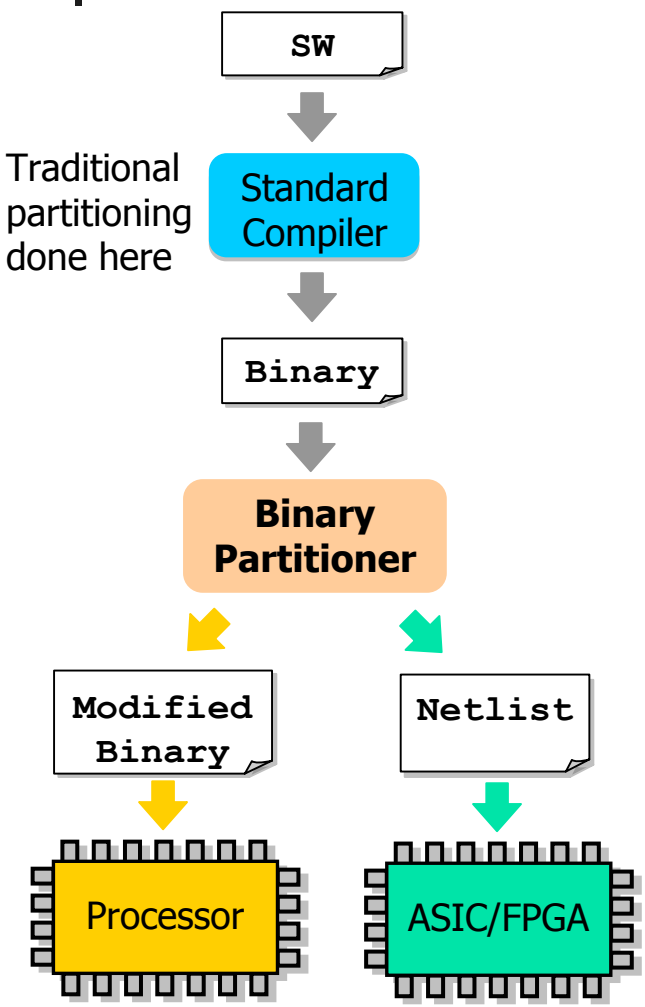
Hardware/Software Partitioning



- Benefits
  - Speedups of 2X to **10X** typical
  - Speedups of **800X** possible
  - Far more potential than dynamic SW optimizations (1.3X)
  - Energy reductions of 25% to 95% typical
- But can hw/sw partitioning be done dynamically?

Introduction

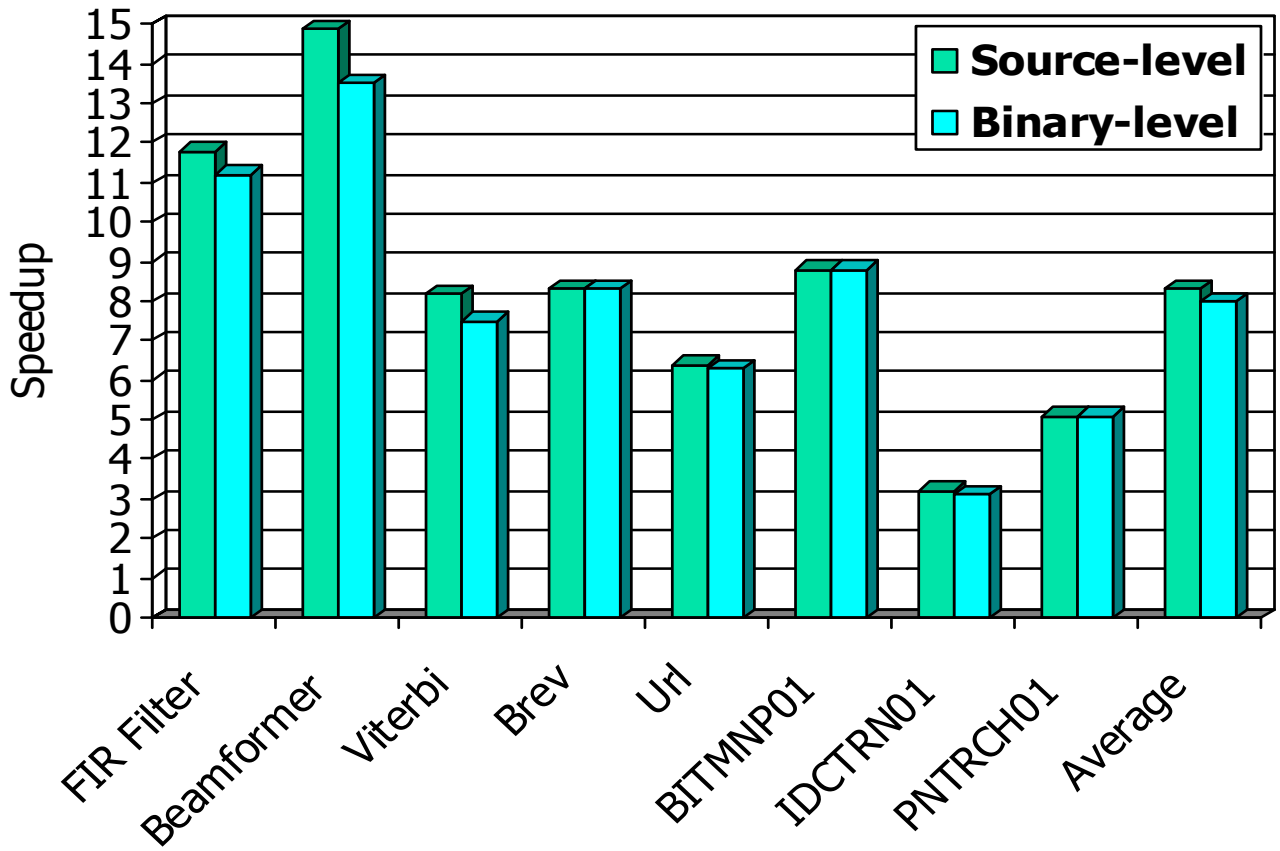
Binary-Level Hardware/Software Partitioning



- Can hw/sw partitioning be done dynamically?
  - Enabler – ***binary-level* partitioning**
  - *[Stitt & Vahid, ICCAD'02]*
  - Partition starting from SW binary
  - Can be desktop based
- Advantages
  - Any compiler, any language, multiple sources, assembly/object support, legacy code support
- Disadvantage
  - Loses high-level information
    - Quality loss?

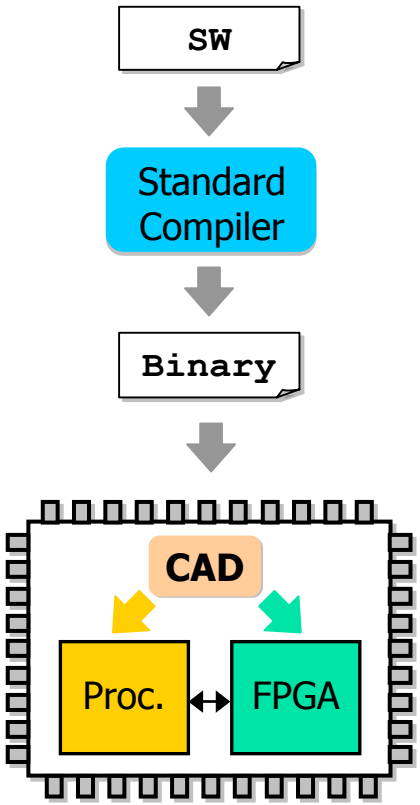
Introduction

Binary-Level Hardware/Software Partitioning



Introduction

*Binary Partitioning Enables **Dynamic** Partitioning*

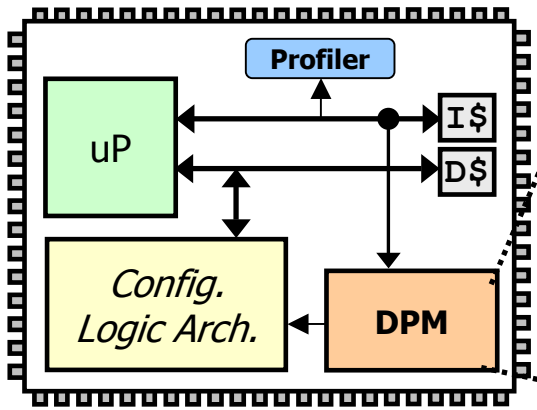


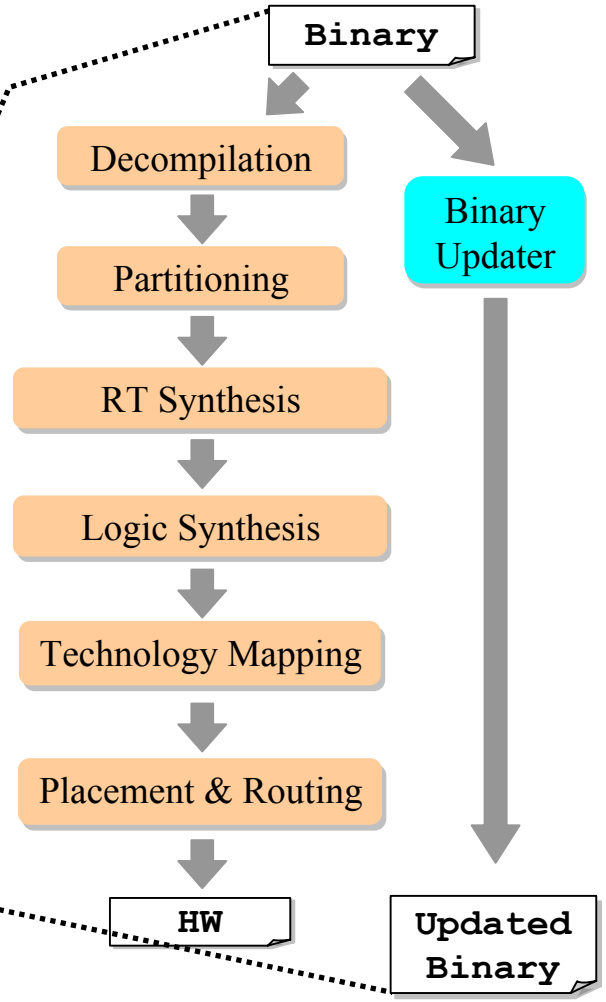
- Dynamic HW/SW Partitioning
  - Embed partitioning CAD tools on-chip
  - Feasible in era of billion-transistor chips
- Advantages
  - No special desktop tools
  - Completely transparent
  - Avoid complexities of supporting different FPGA types
- Complements other approaches
  - Desktop CAD best from purely technical perspective
  - Dynamic opens additional market segments (i.e., ***all*** software developers) that otherwise might not use desktop CAD

Warp Processors

Tools & Requirements

- Warp Processor Architecture
  - On-chip profiling architecture
  - Configurable logic architecture
  - Dynamic partitioning module

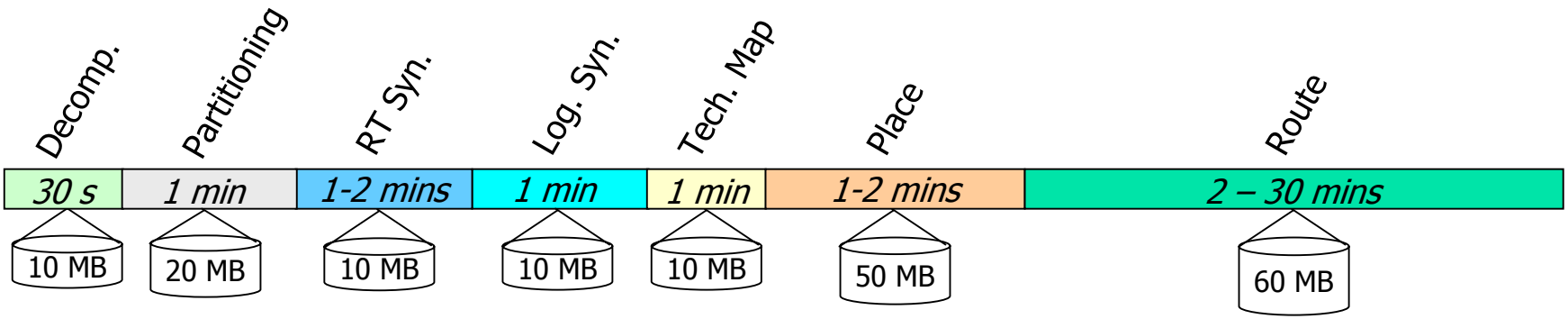




Warp Processors

All that CAD on-chip?

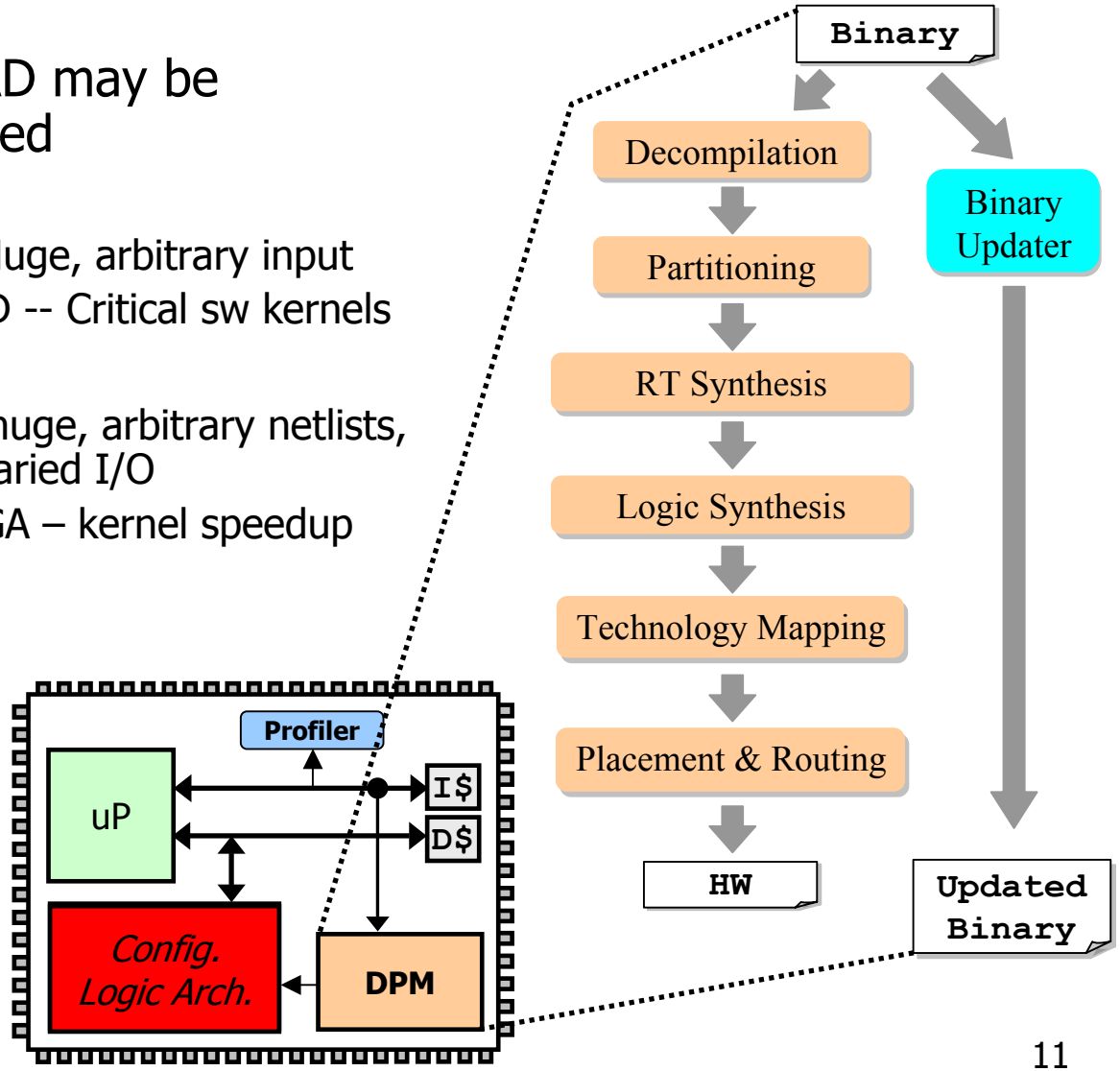
- CAD people may first think dynamic HW/SW partitioning is *"absurd"*
  - *Those CAD tools are complex*
    - *Require long execution times on powerful desktop workstations*
    - *Require very large memory resources*
    - *Usually require GBytes of hard drive space*
    - *Costs of complete CAD tools package can exceed $1 million*
    - *All that on-chip?*



Warp Processors

Tools & Requirements

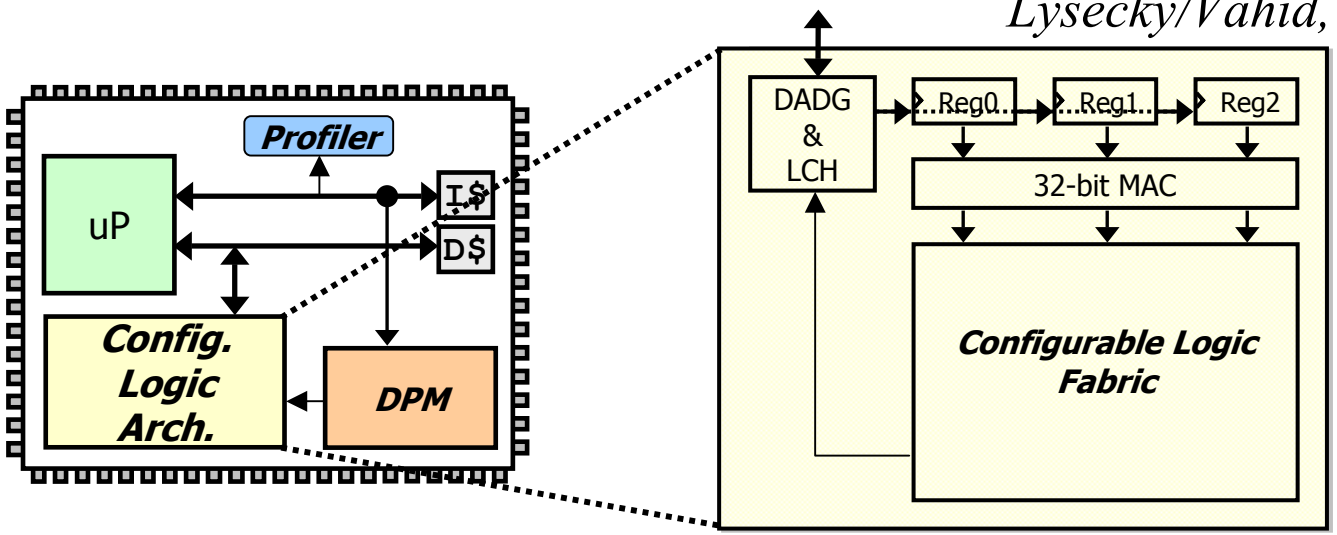
- But, in fact, on-chip CAD may be practical since specialized
  - CAD
    - Traditional CAD -- Huge, arbitrary input
    - Warp Processor CAD -- Critical sw kernels
  - FPGA
    - Traditional FPGA – huge, arbitrary netlists, ASIC prototyping, varied I/O
    - Warp Processor FPGA – kernel speedup
- Careful simultaneous design of FPGA and CAD
  - FPGA features evaluated for impact on CAD
  - CAD influences FPGA features
  - Add architecture features for kernels



Warp Processors
Configurable Logic Architecture

- Loop support hardware
  - Data address generators (DADG) and loop control hardware (LCH), found in digital signal processors – fast loop execution
  - Supports memory accesses with regular access pattern
  - Synthesis of FSM not required for many critical loops
- 32-bit fast Multiply-Accumulate (MAC) unit

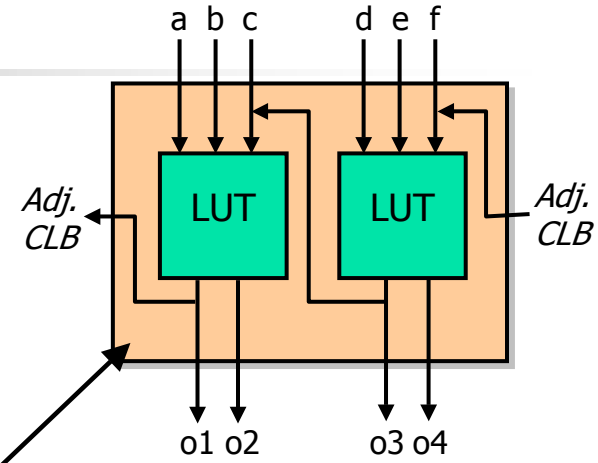
Lysecky/Vahid, DATE'04

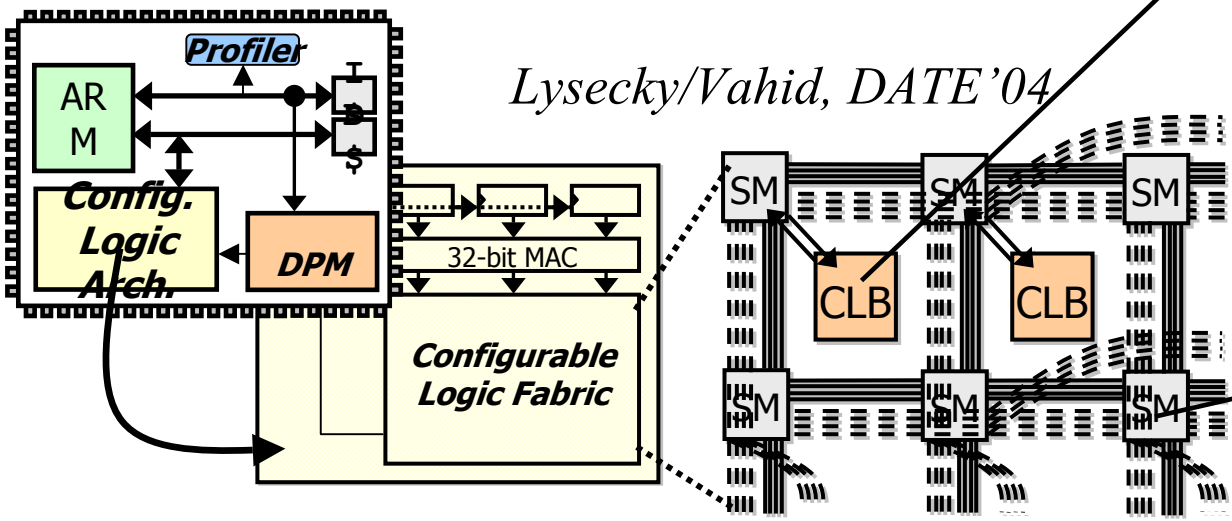


Warp Processors

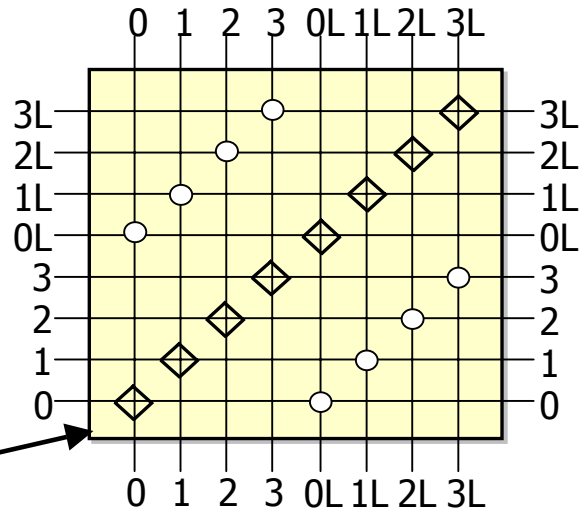
Configurable Logic Fabric

- Simple fabric: array of configurable logic blocks (CLBs) surrounded by switch matrices (SMs)
- Simple CLB: Two 3-input 2-output LUTs
  - carry-chain support
- Simple switch matrices: 4-short, 4-long channels
- Designed for simple fast CAD





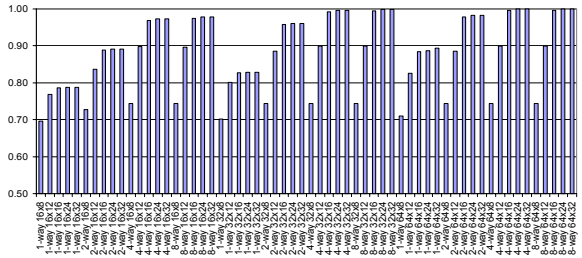
Lysecky/Vahid, DATE'04

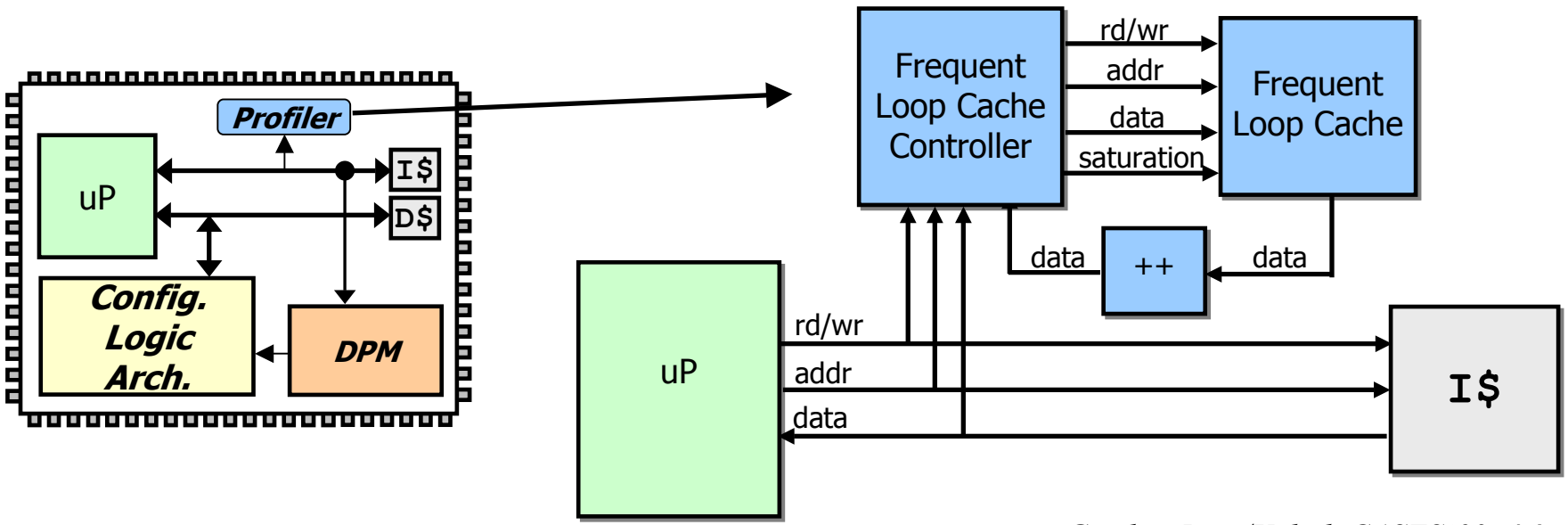


Warp Processors

Profiler

- Non-intrusive on-chip loop profiler
  - Gordon-Ross/Vahid CASES'03, to appear in "best of MICRO/CASES" issue of IEEE Trans. on Computers.
  - Provides relative frequency of top 16 loops
  - Small cache (16 entries), only 2,300 gates
    - Less than 1% power overhead when active

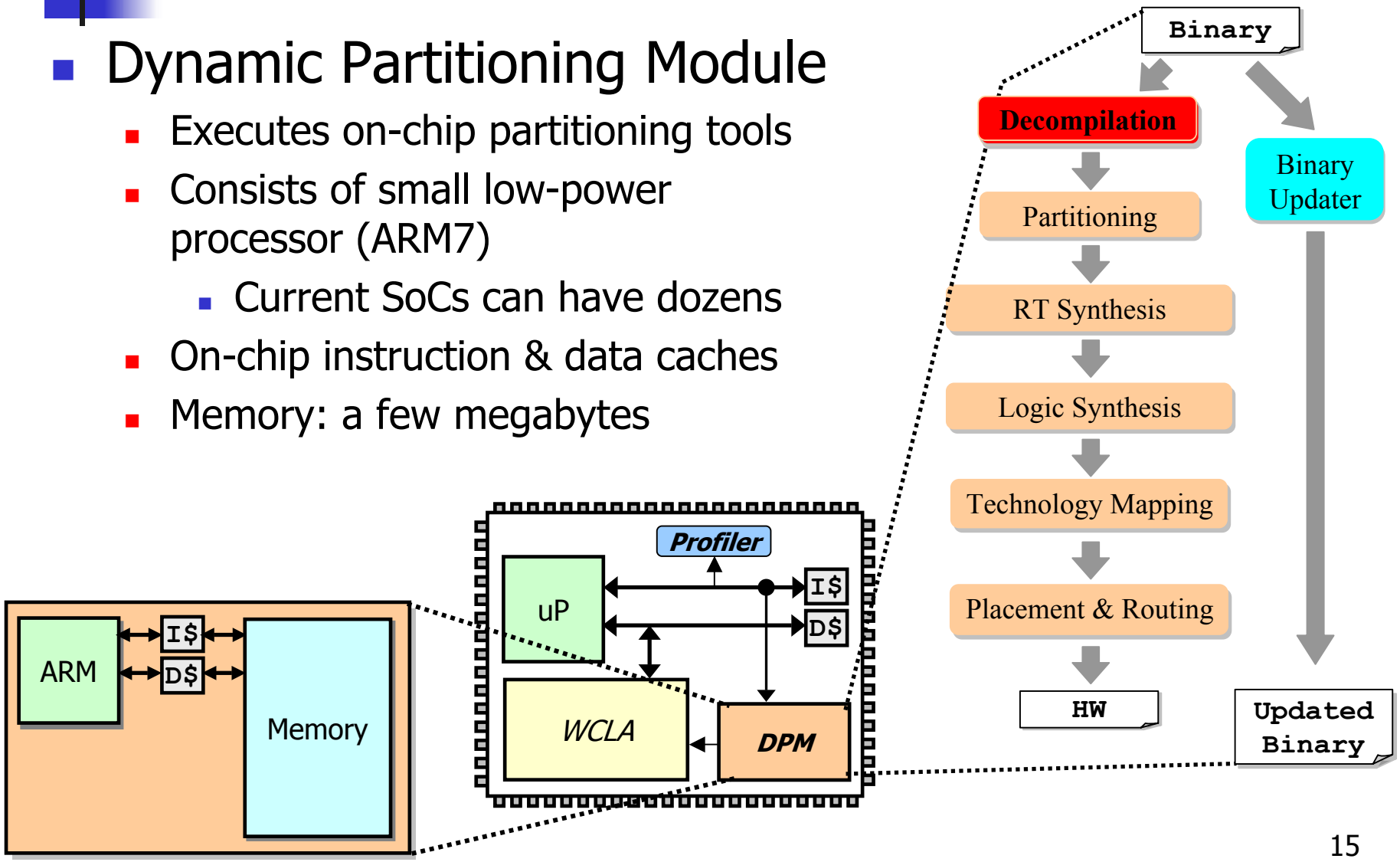




Warp Processors

Dynamic Partitioning Module (DPM)

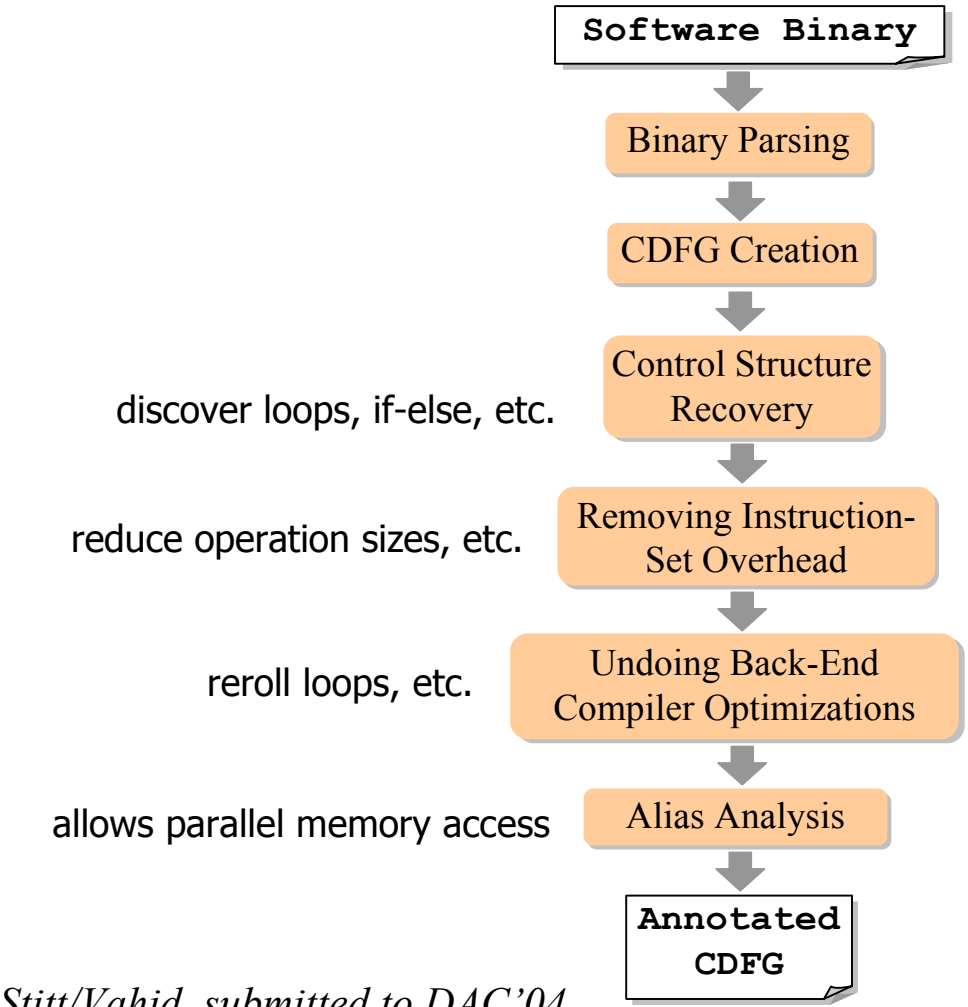
- Dynamic Partitioning Module
  - Executes on-chip partitioning tools
  - Consists of small low-power processor (ARM7)
    - Current SoCs can have dozens
  - On-chip instruction & data caches
  - Memory: a few megabytes



Warp Processors

Decompilation

- Goal: recover high-level information lost during compilation
  - Otherwise, synthesis results will be poor
- Utilize sophisticated decompilation methods
  - Developed over past decades for binary translation
  - Indirect jumps hamper CDFG recovery
    - But not too common in critical loops (function pointers, switch statements)



Stitt/Vahid, submitted to DAC'04

Warp Processors

Decompilation Results

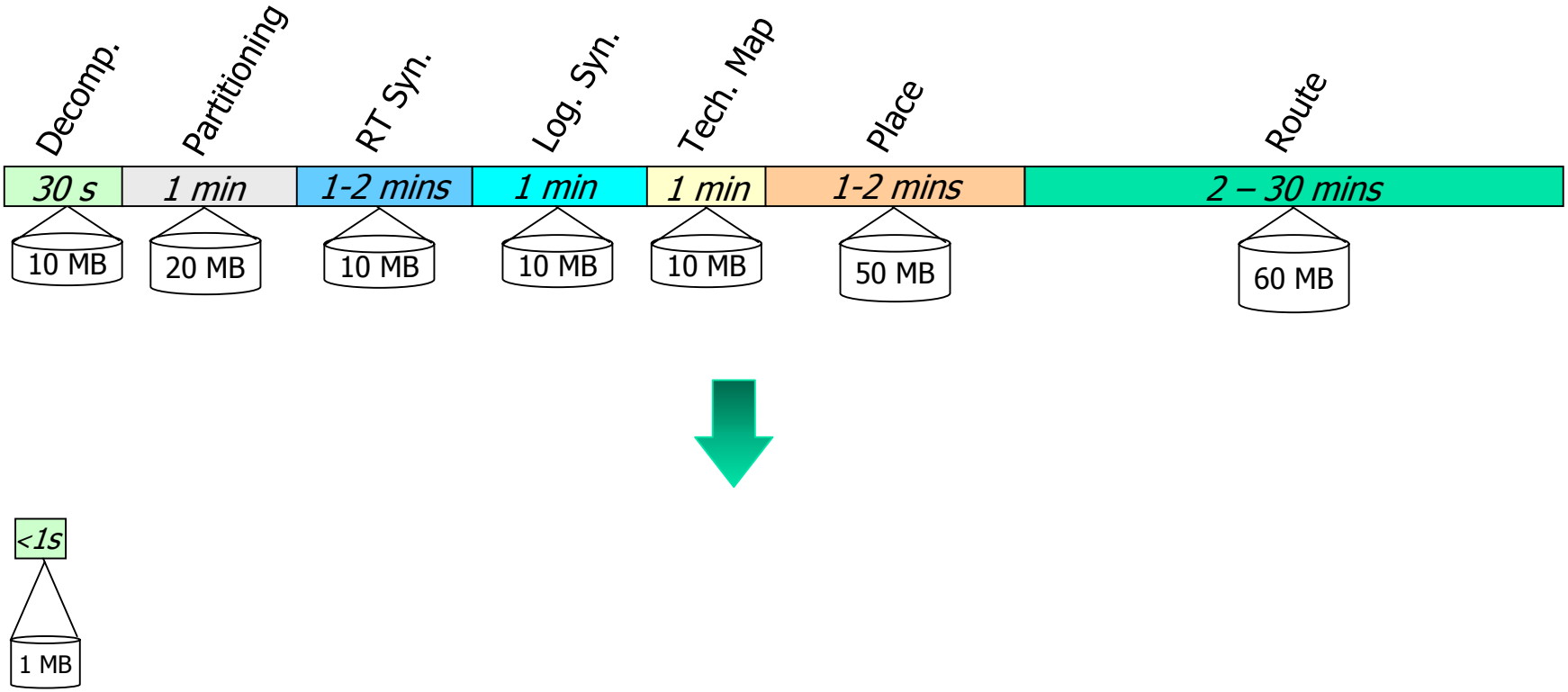
- In most situations, we can recover all high-level information
  - Recovery success for a dozen benchmarks, using several different compilers and optimization levels:

Decompilation Process	Success Rate
CDFG Recovery	88%
Loop Identification	100%
Loop Type Recovery	100%
If Statement Recovery	100%
Array Recovery	100%
Unrolled Loop Identification	100%
Loop Rerolling	100%

Stitt/Vahid, submitted to DAC'04

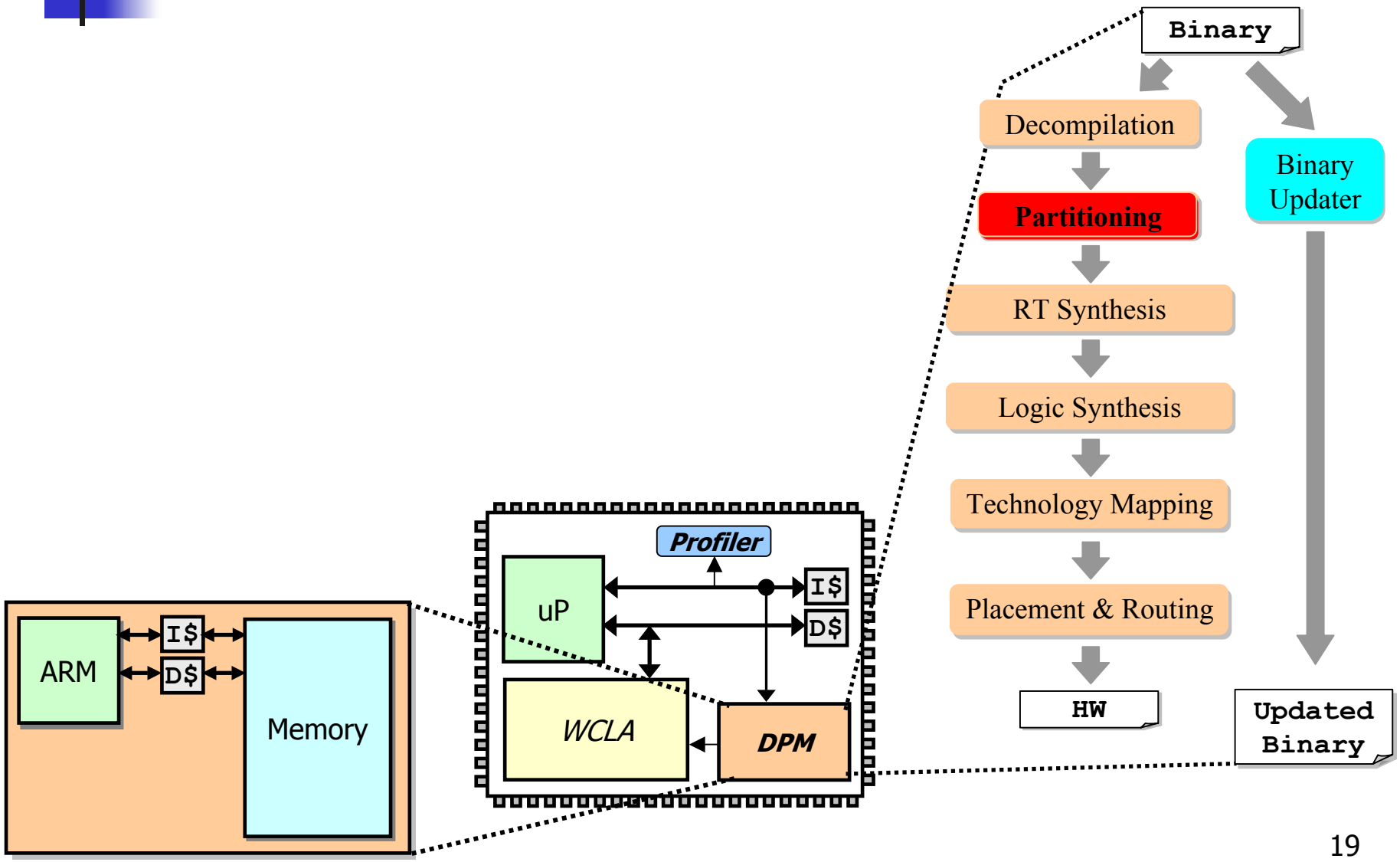
Warp Processors

Execution Time and Memory Requirements



Warp Processors

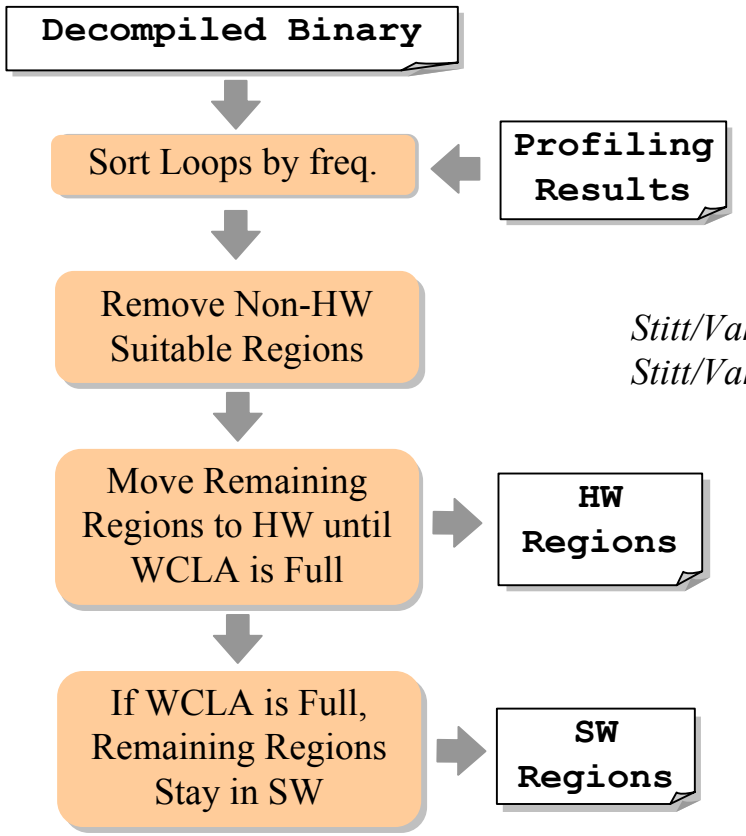
Dynamic Partitioning Module (DPM)



Warp Processors

Binary HW/SW Partitioning

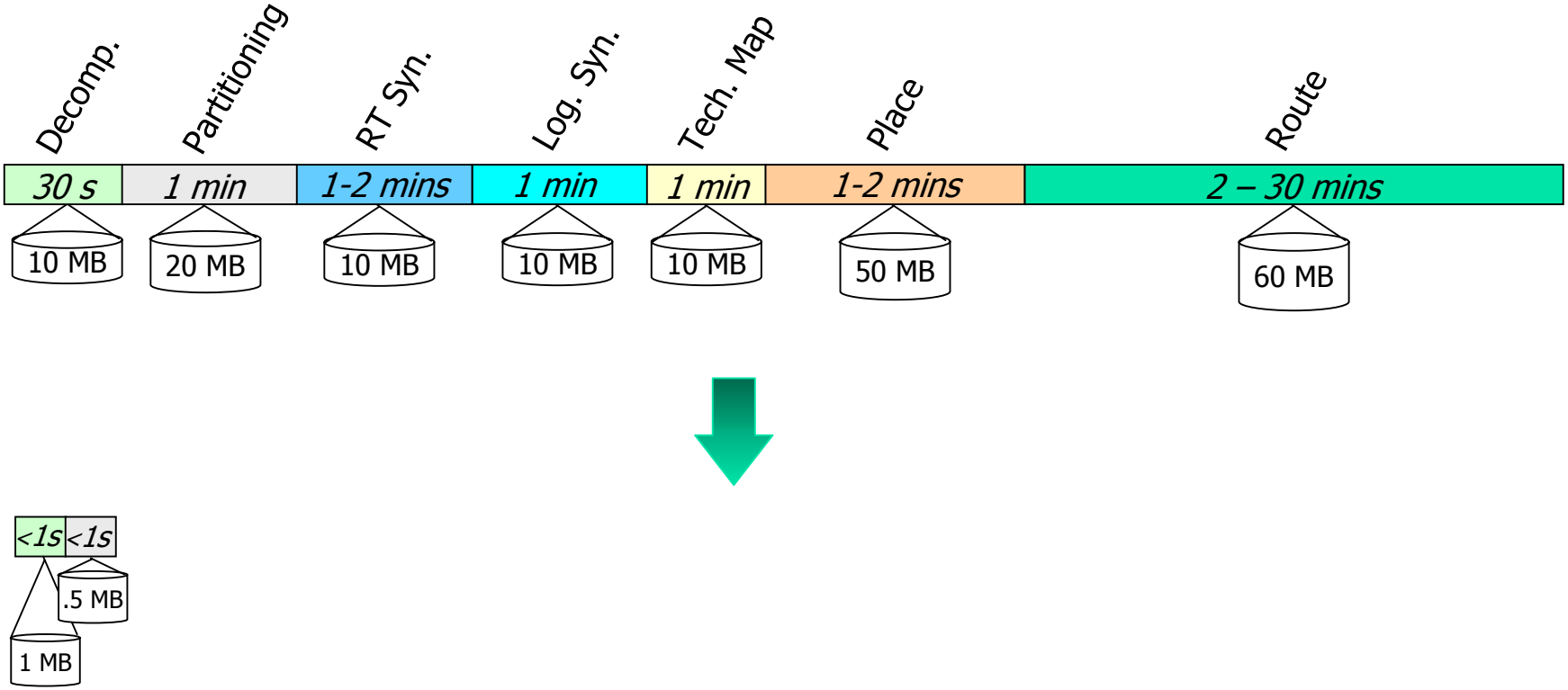
Simple partitioning algorithm -- move most frequent loops to hardware
Usually one 2-3 critical loops comprise most execution



Stitt/Vahid, ICCAD'02
Stitt/Vahid, submitted to DAC'04

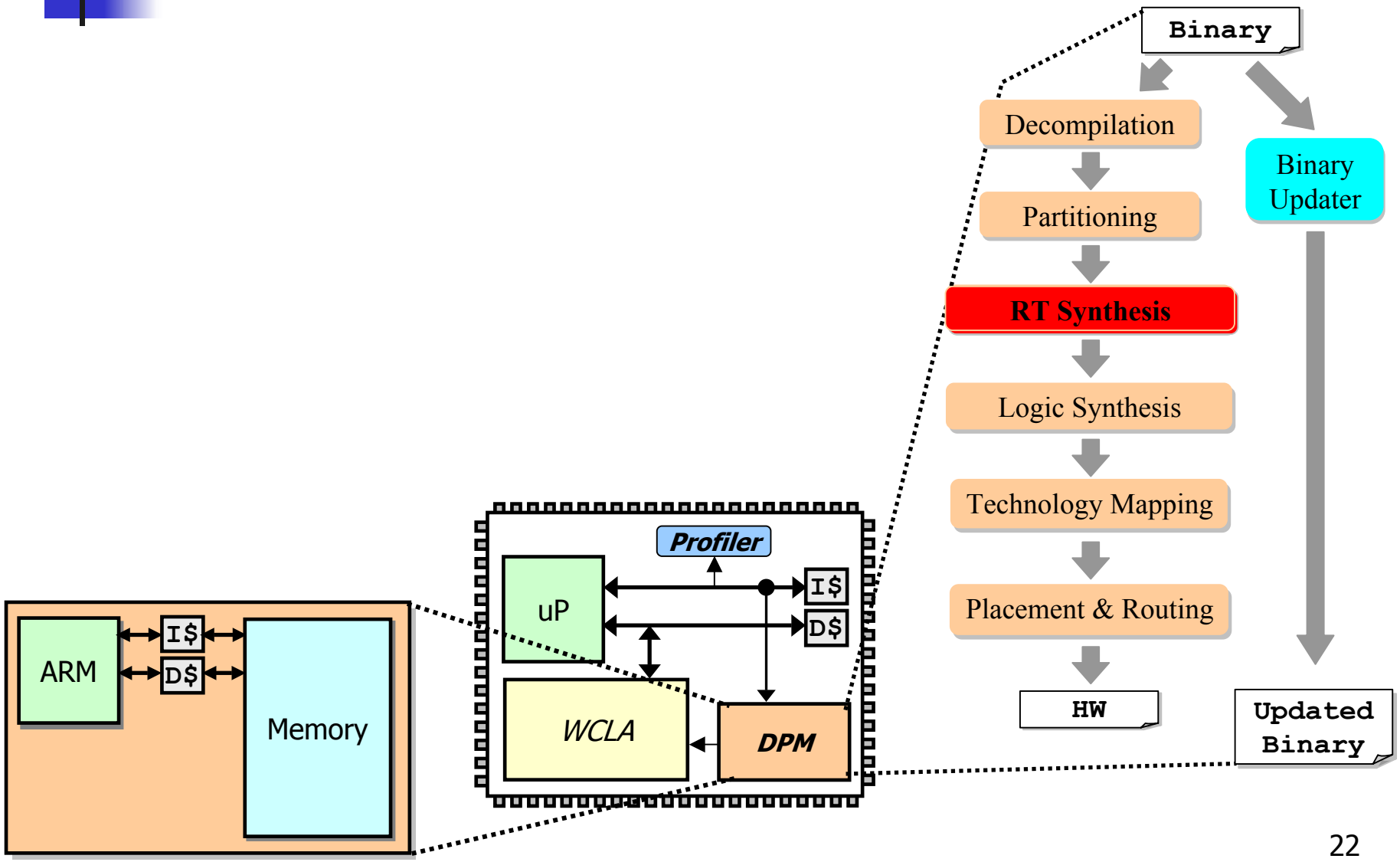
Warp Processors

Execution Time and Memory Requirements



Warp Processors

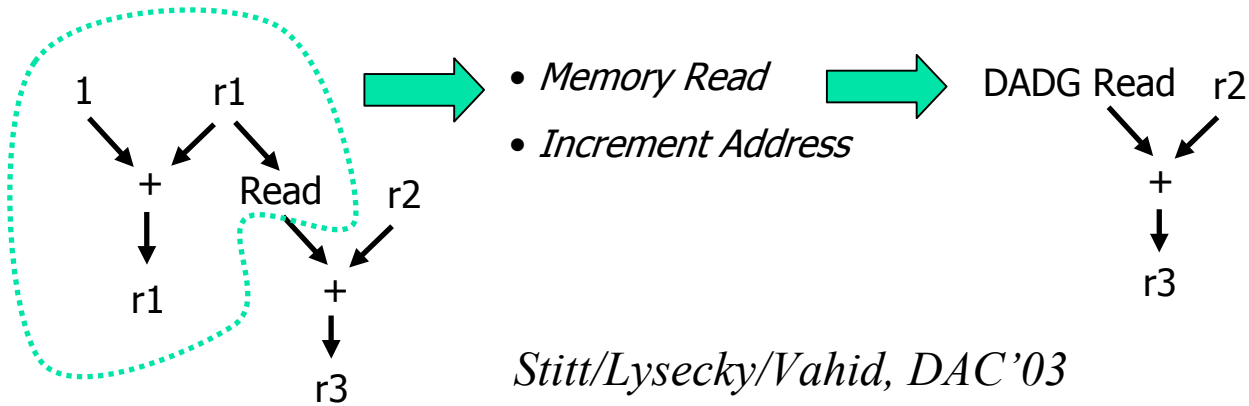
Dynamic Partitioning Module (DPM)



Warp Processors

RT Synthesis

- Converts decompiled CDFG to Boolean expressions
- Maps memory accesses to our data address generator architecture
  - Detects read/write, memory access pattern, memory read/write ordering
- Optimizes dataflow graph
  - Removes address calculations and loop counter/exit conditions
  - Loop control handled by Loop Control Hardware

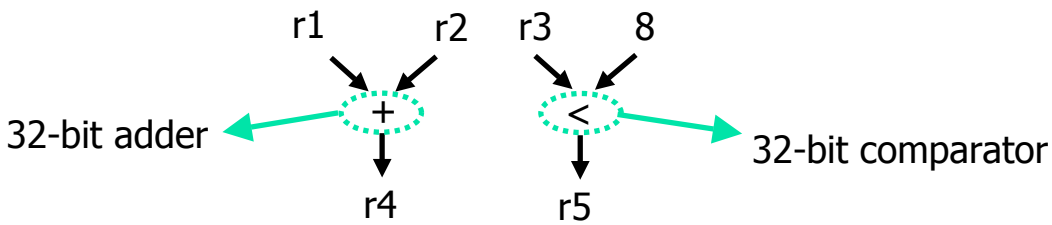


Stitt/Lysecky/Vahid, DAC'03

Warp Processors

RT Synthesis

- Maps dataflow operations to hardware components
  - We currently support adders, comparators, shifters, Boolean logic, and multipliers
- Creates Boolean expression for each output bit of dataflow graph



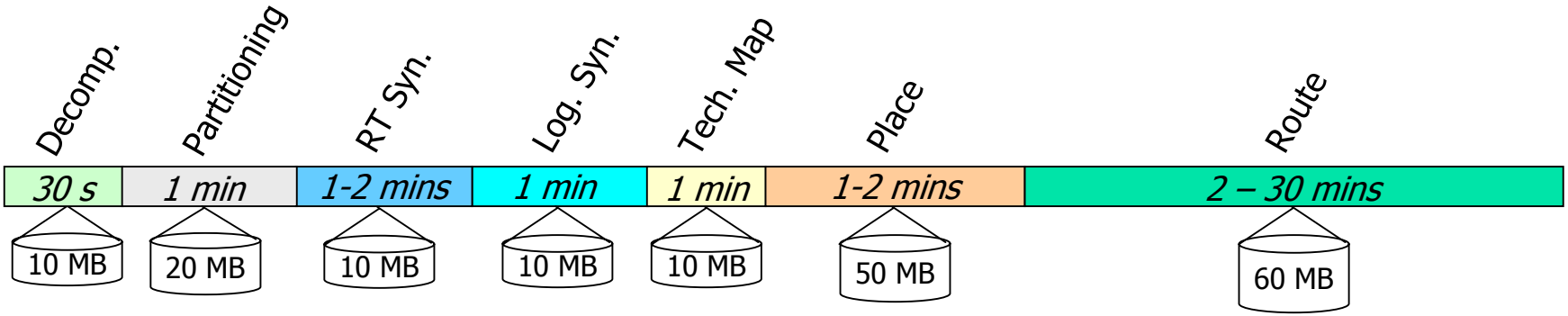
r4[0]=r1[0] xor r2[0], carry[0]=r1[0] and r2[0]

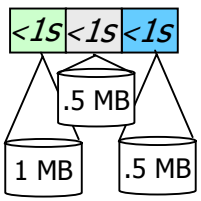
r4[1]=(r1[1] xor r2[1]) xor carry[0], carry[1]= …….

…….

Warp Processors

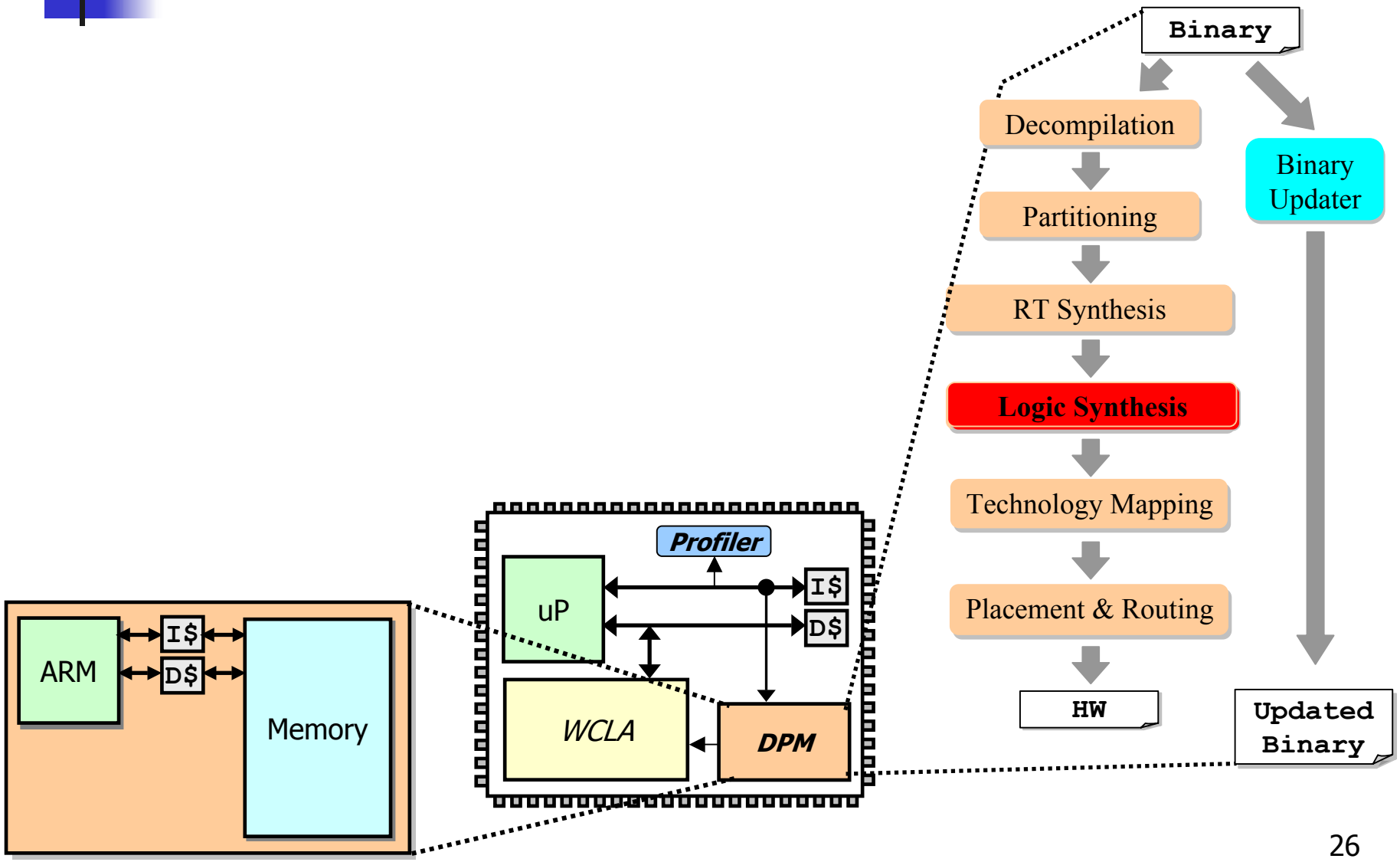
Execution Time and Memory Requirements





Warp Processors

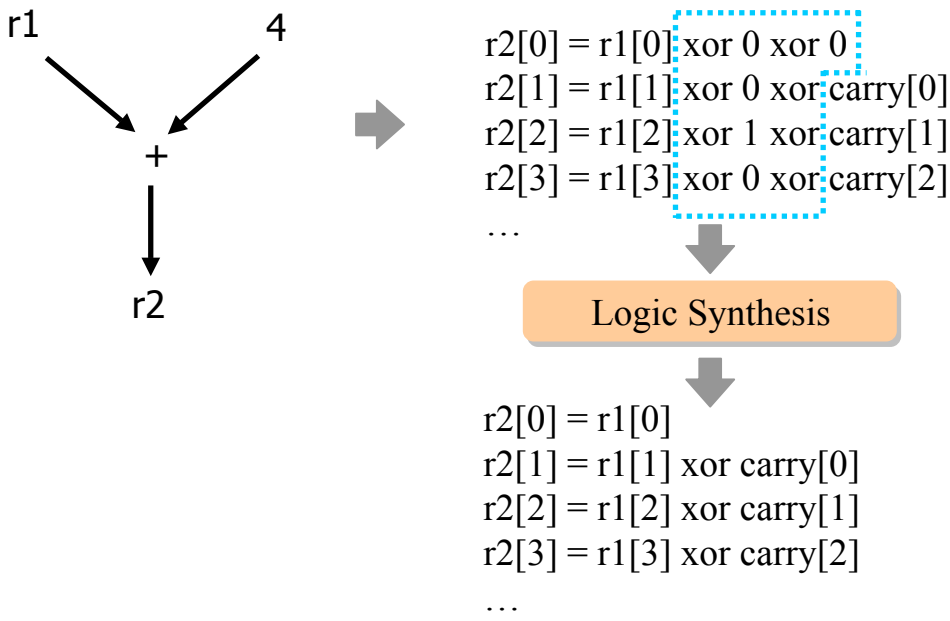
Dynamic Partitioning Module (DPM)



Warp Processors

Logic Synthesis

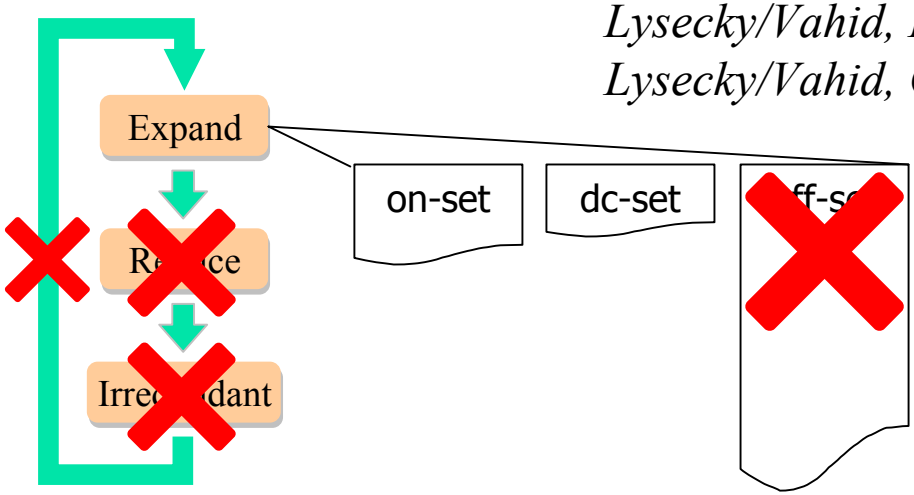
- Optimize hardware circuit created during RT synthesis
  - Large opportunity for logic minimization due to use of immediate values in the binary code
- Utilize simple two-level logic minimization approach



Warp Processors - ROCM

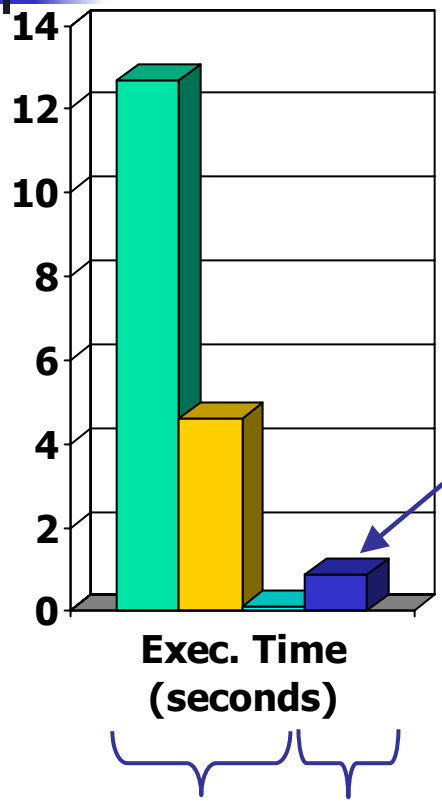
- ROCM – Riverside On-Chip Minimizer
  - Two-level minimization tool
  - Utilized a combination of approaches from Espresso-II *[Brayton, et al. 1984]* and Presto *[Svoboda & White, 1979]*
  - Eliminate the need to compute the off-set to reduce memory usage
  - Utilizes a single expand phase instead of multiple iterations
  - On average only 2% larger than optimal solution for benchmarks

Lysecky/Vahid, DAC'03
Lysecky/Vahid, CODES+ISSS'03



Warp Processors - ROCM

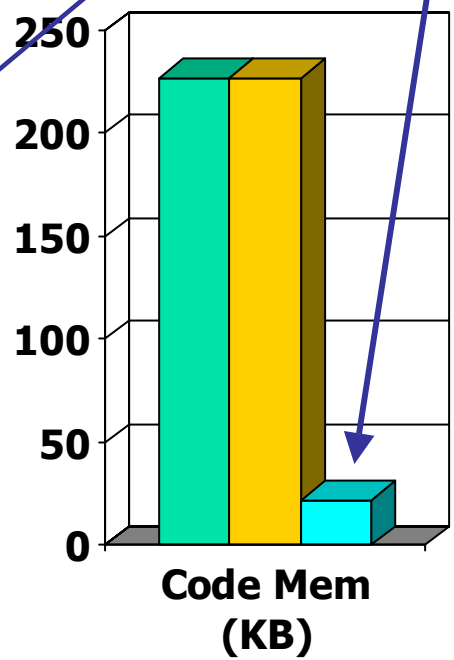
Results

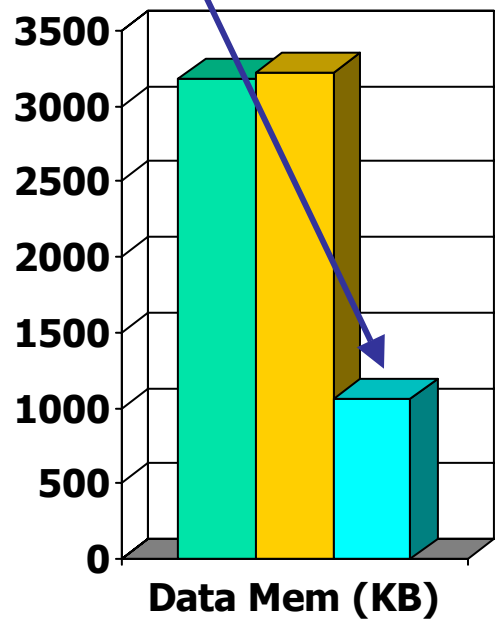


ROCM executing on 40MHz ARM7 requires less than 1 second

Small code size of only 22 kilobytes

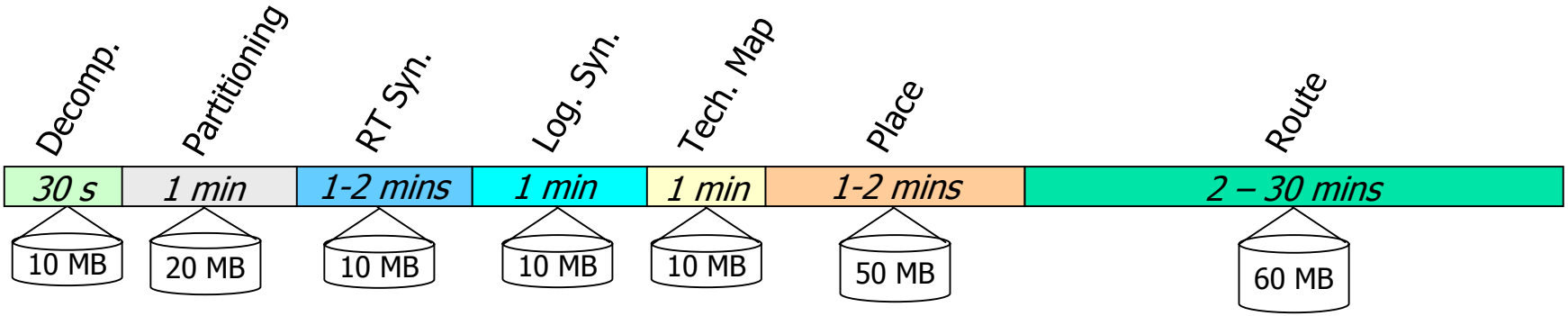
Average data memory usage of only 1 megabyte

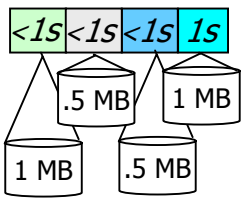




Warp Processors

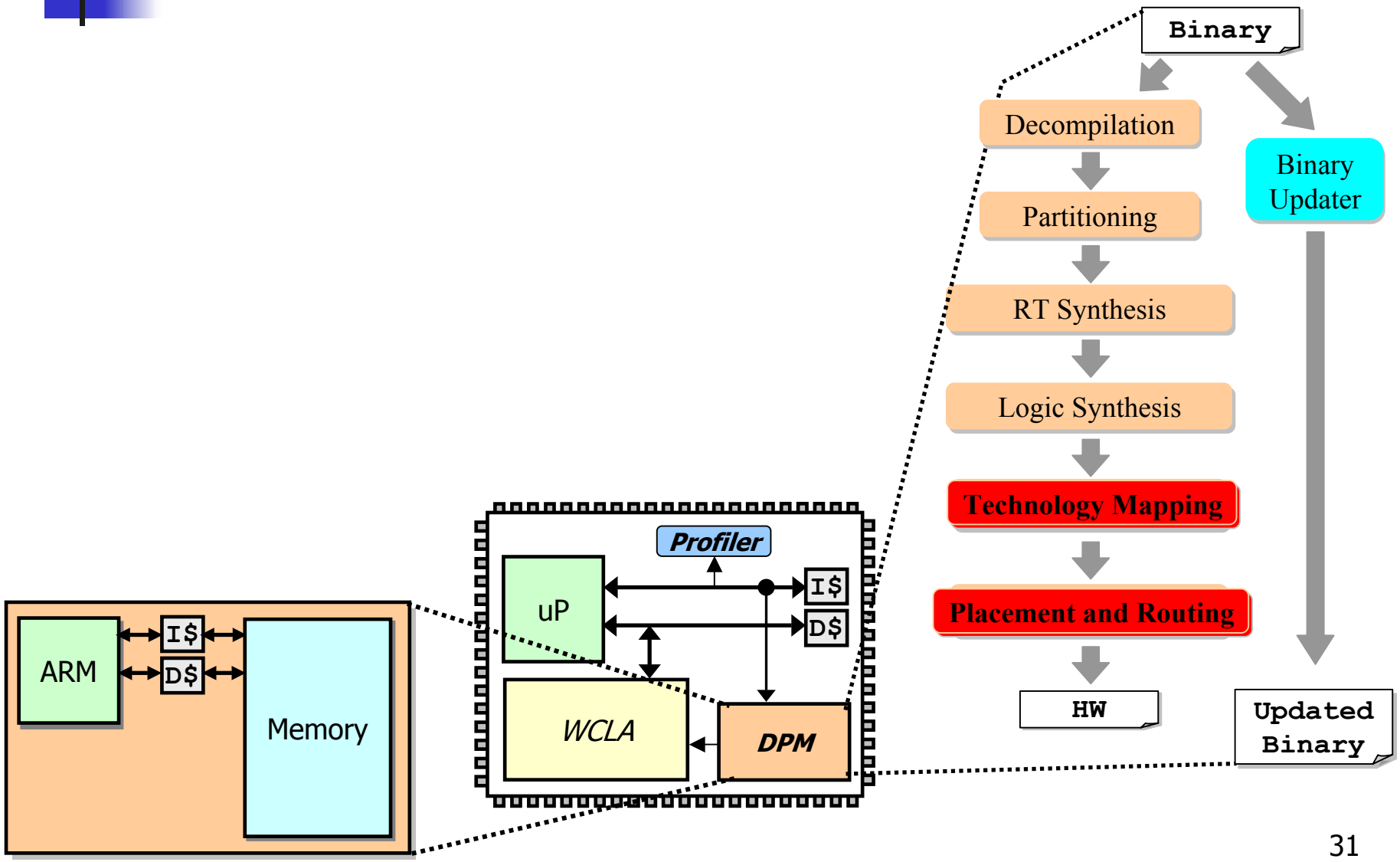
Execution Time and Memory Requirements





Warp Processors

Dynamic Partitioning Module (DPM)

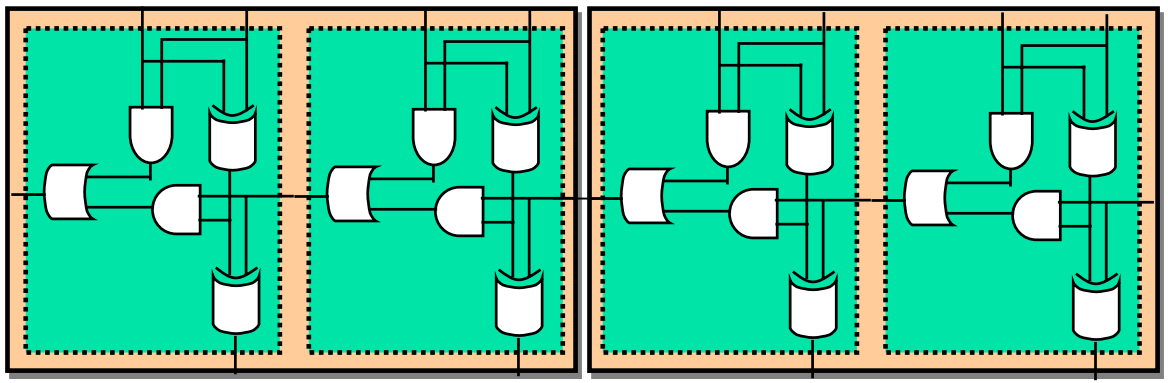


Warp Processors

Technology Mapping/Packing

- ROCPAR – Technology Mapping/Packing
  - Decompose hardware circuit into basic logic gates *(AND, OR, XOR, etc.)*
  - Traverse logic network combining nodes to form single-output LUTs
  - Combine LUTs with common inputs to form final 2-output LUTs
  - Pack LUTs in which output from one LUT is input to second LUT
  - Pack remaining LUTs into CLBs

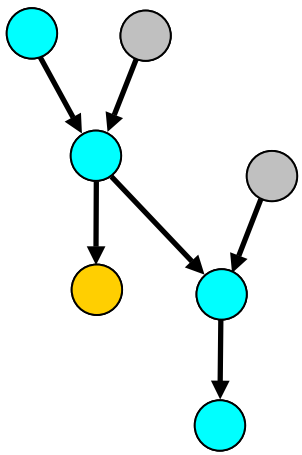
Lysecky/Vahid, DATE'04
Stitt/Lysecky/Vahid, DAC'03



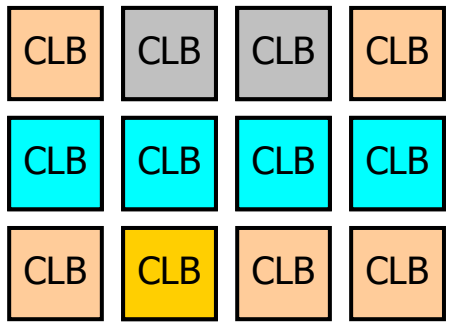
Warp Processors

Placement

- ROCPAR – Placement
  - Identify critical path, placing critical nodes in center of configurable logic fabric
  - Use dependencies between remaining CLBs to determine placement
    - Attempt to use adjacent cell routing whenever possible

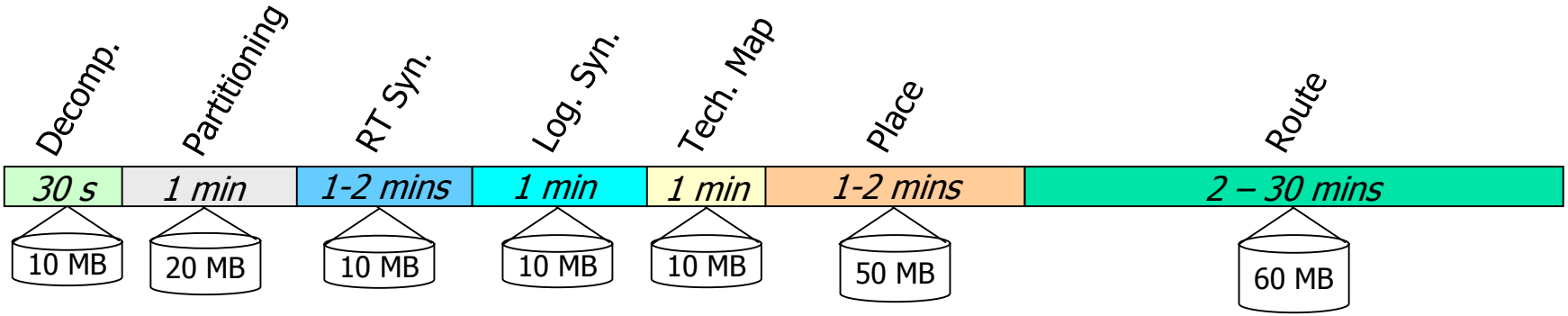


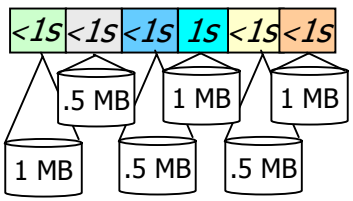
Lysecky/Vahid, DATE'04
Stitt/Lysecky/Vahid, DAC'03



Warp Processors

Execution Time and Memory Requirements

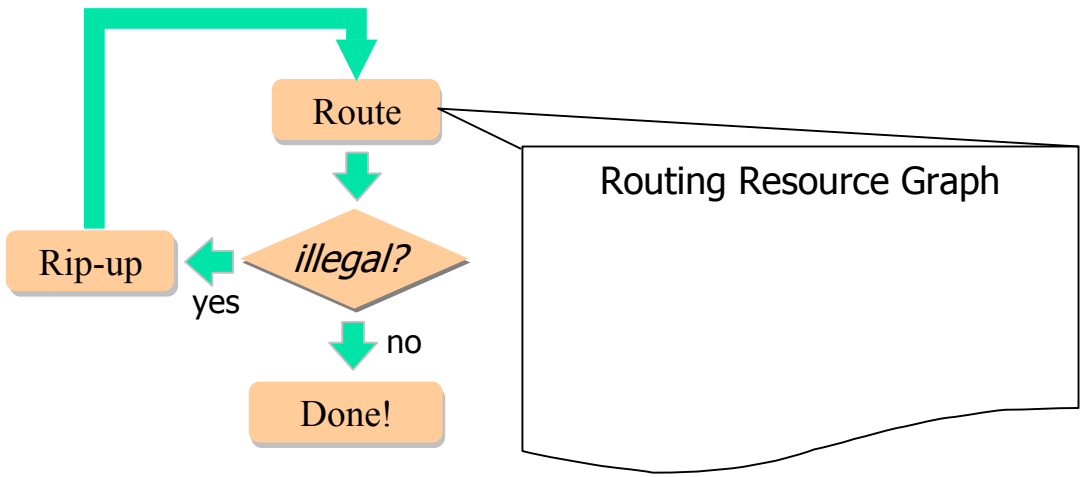




Warp Processors

Routing

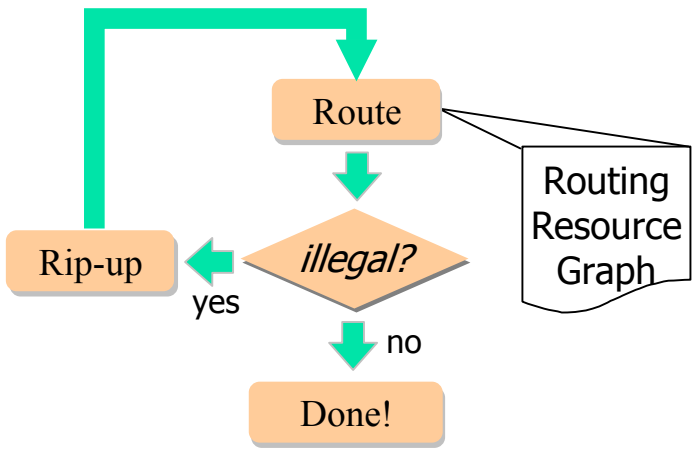
- FPGA Routing
  - Find a path within FPGA to connect source and sinks of each net
- VPR – Versatile Place and Route *[Betz, et al., 1997]*
  - Modified Pathfinder algorithm
    - Allows overuse of routing resources during each routing iteration
    - If illegal routes exists, update routing costs, rip-up all routes, and reroute
  - Increases performance over original Pathfinder algorithm
  - Routability-driven routing: Use fewest tracks possible
  - Timing-driven routing: Optimize circuit speed



Warp Processors

Routing

- Riverside On-Chip Router (ROCR)
  - Represent routing nets between CLBs as routing between SMs
    - Resource Graph
      - Nodes correspond to SMs
      - Edges correspond to short and long channels between SMs
  - Routing
    - Greedy, depth-first routing algorithm routes nets between SMs
    - Assign specific channels to each route, using Brelaz's greedy vertex coloring algorithm
  - Requires much less memory than VPR as resource graph is much smaller



Warp Processors

Routing: Performance and Memory Usage Results

Bench-mark	VPR (RD) Time	VPR (RD) Mem	VPR (TD) Time	VPR (TD) Mem	ROCR Time	ROCR Mem
alu4	221.1	16508	8.3	12312	0.6	3484
apex2	315.6	18780	12.4	14552	4.3	3496
apex4	213.7	14332	7.8	11128	0.6	3468
bigkey	403.6	47944	13.5	37648	1.3	3512
des	376.0	52276	12.8	49980	1.0	3496
diffeq	135.5	15484	5.8	12576	0.4	3480
dsip	231.2	47796	10.4	37496	0.9	3500
e64	19.2	6296	1.0	5644	0.1	3428
elliptic	770.4	33524	33.7	26244	7.8	3572
ex5p	187.8	12612	6.3	9840	0.3	3460
frisc	865.9	33468	35.1	27112	13.8	3564
misex3	190.9	15628	6.8	11508	0.4	3468
s1423	4.1	4240	0.5	3548	0.1	3428
s298	265.0	18984	11.6	15384	0.7	3496
s38417	1428.2	57380	49.9	44180	8.7	3680
s38584.1	1110.1	51760	36.0	43700	8.8	3692
seq	291.1	17198	11.4	13800	2.2	3488
tseng	73.6	11048	3.1	8960	0.2	3464
Average	**394.6**	**26403**	**14.8**	**21423**	**2.9**	**3510**

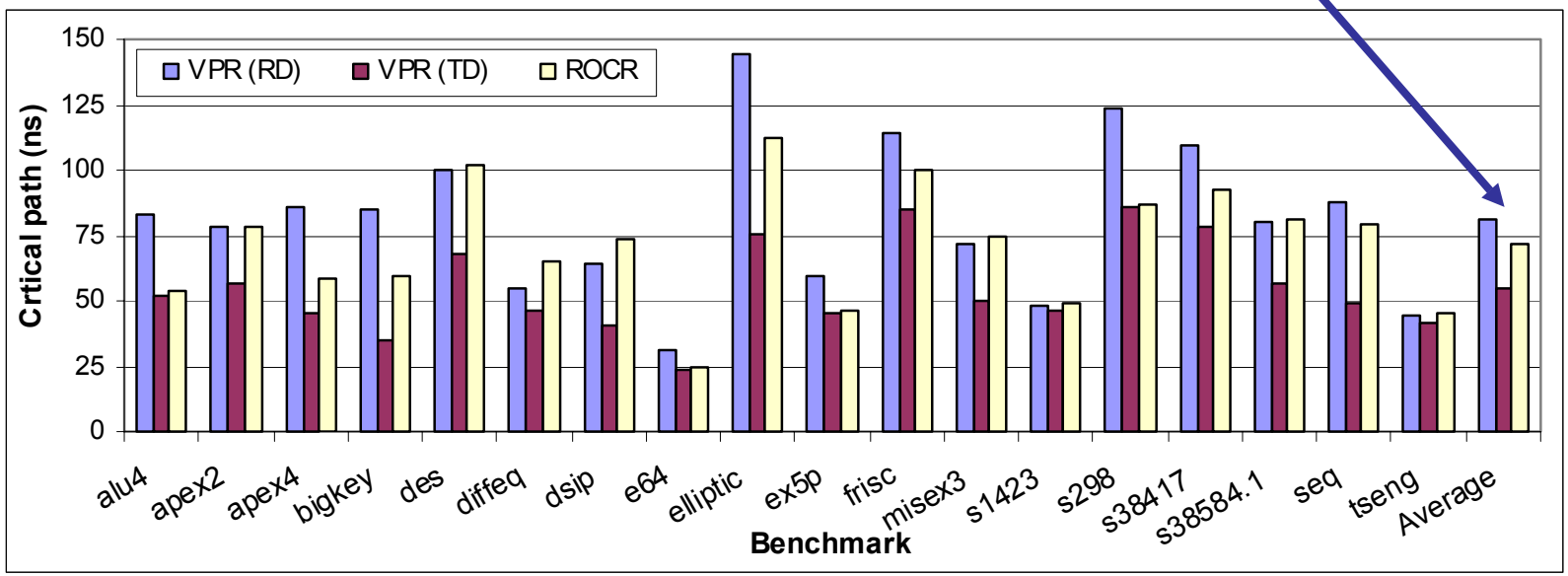
- Average 10X faster than VPR (TD)
  - Up to 21X faster for *ex5p*
- Memory usage of only 3.6 MB
  - 13X less than VPR

Warp Processors

Routing: Critical Path Results

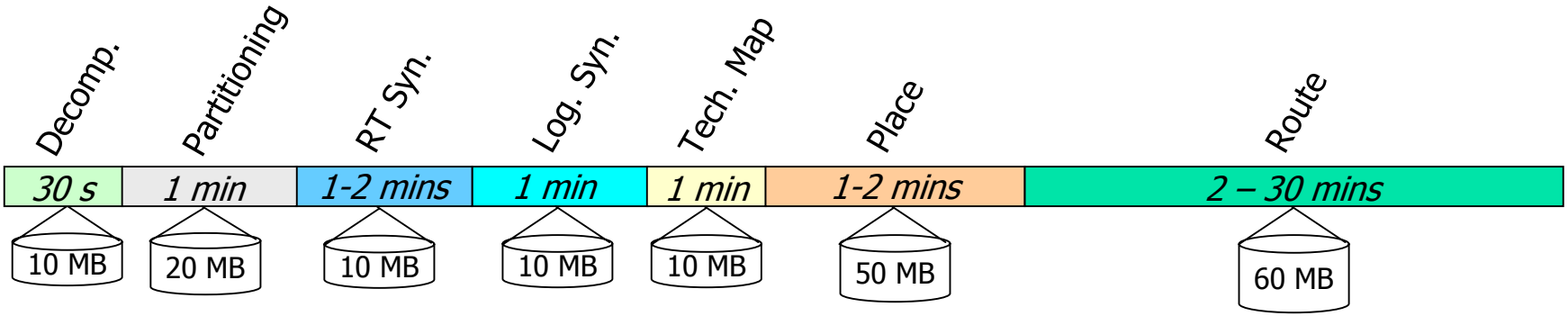
32% longer critical path than VPR (Timing Driven)

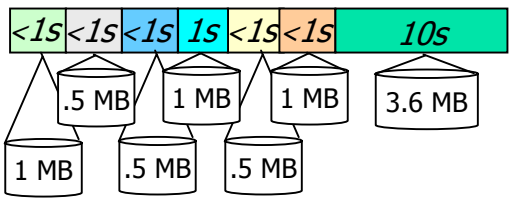
10% shorter critical path than VPR (Routability Driven)



Warp Processors

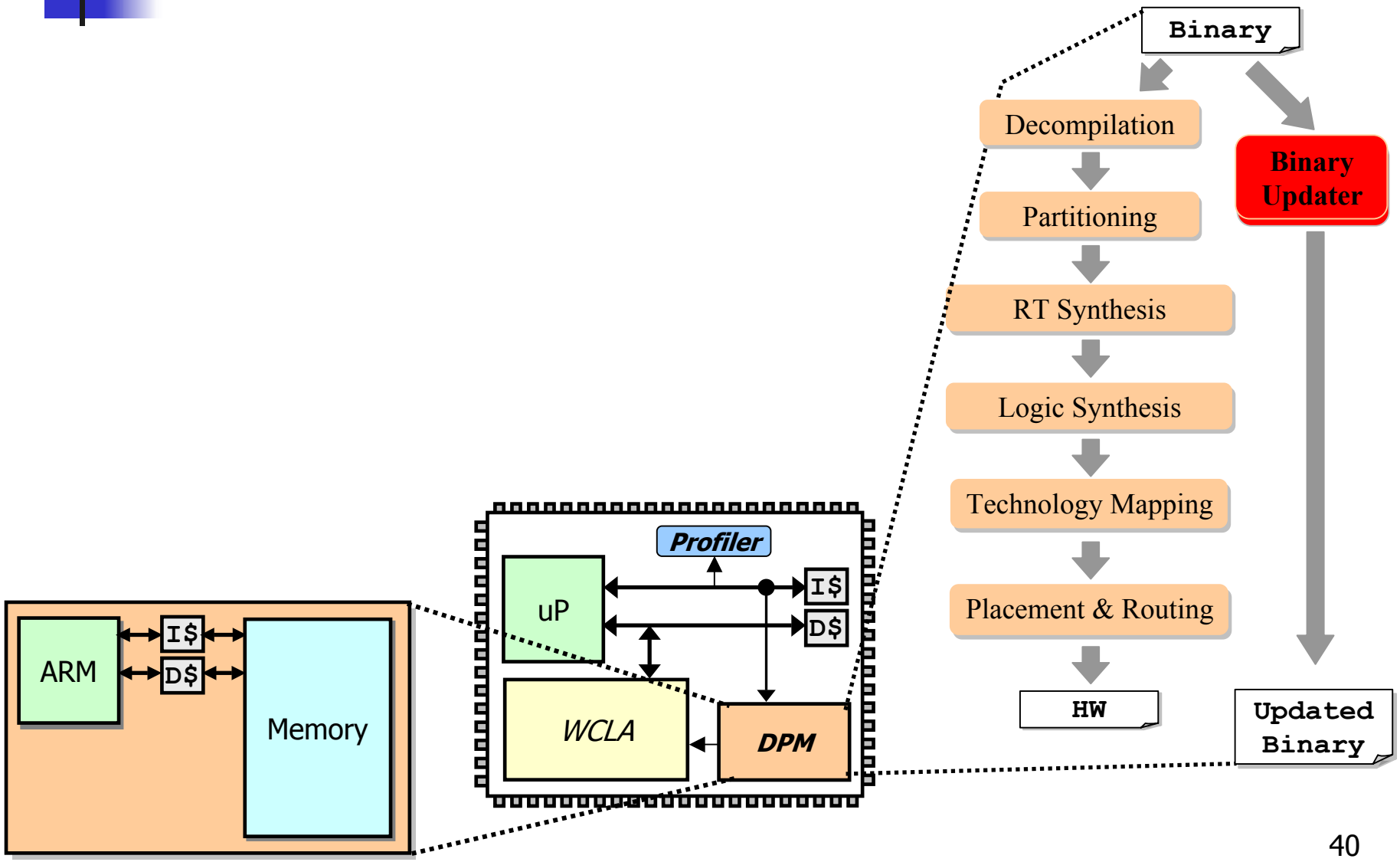
Execution Time and Memory Requirements





Warp Processors

Dynamic Partitioning Module (DPM)



Warp Processors

Binary Updater

- Binary Updater
  - Must modify binary to use hardware within WCLA
  - HW initialization function added at end of binary
  - Replace HW loops with jump to HW initialization function
    - HW initialization function jumps back to end of loop

```
..
..
..
for (i=0; i < 256; i++)
   output += input1[i]*2;
..
..
..
```

```
..
..
..
initHW();
..
..
..

initHW() {
..
}
```

Initial Overall Results

Experimental Setup

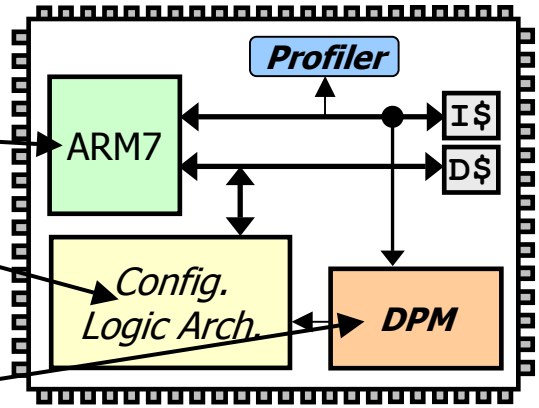
- Considered 12 embedded benchmarks from NetBench, MediaBench, EEMBC, and Powerstone
- Average of 53% of total software execution time was spent executing **single** critical loop (more speedup possible if **more loops** considered)
- On average, critical loops comprised only 1% of total program size

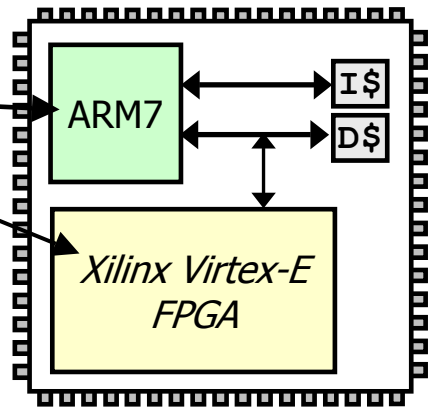
Benchmark	Total Instructions	Loop Instructions	Loop Execution Time	Loop Size%	Speedup Bound
brev	992	104	70.0%	10.5%	3.3
g3fax1	1094	6	31.4%	0.5%	1.5
g3fax2	1094	6	31.2%	0.5%	1.5
url	13526	17	79.9%	0.1%	5
logmin	8968	38	63.8%	0.4%	2.8
pktflow	15582	5	35.5%	0.0%	1.6
canrdr	15640	5	35.0%	0.0%	1.5
bitmnp	17400	165	53.7%	0.9%	2.2
tblook	15668	11	76.0%	0.1%	4.2
ttsprk	16558	11	59.3%	0.1%	2.5
matrix01	16694	22	40.6%	0.1%	1.7
idctrn01	17106	13	62.2%	0.1%	2.6
Average:			***53.2%***	***1.1%***	***3.0***

Warp Processors

Experimental Setup

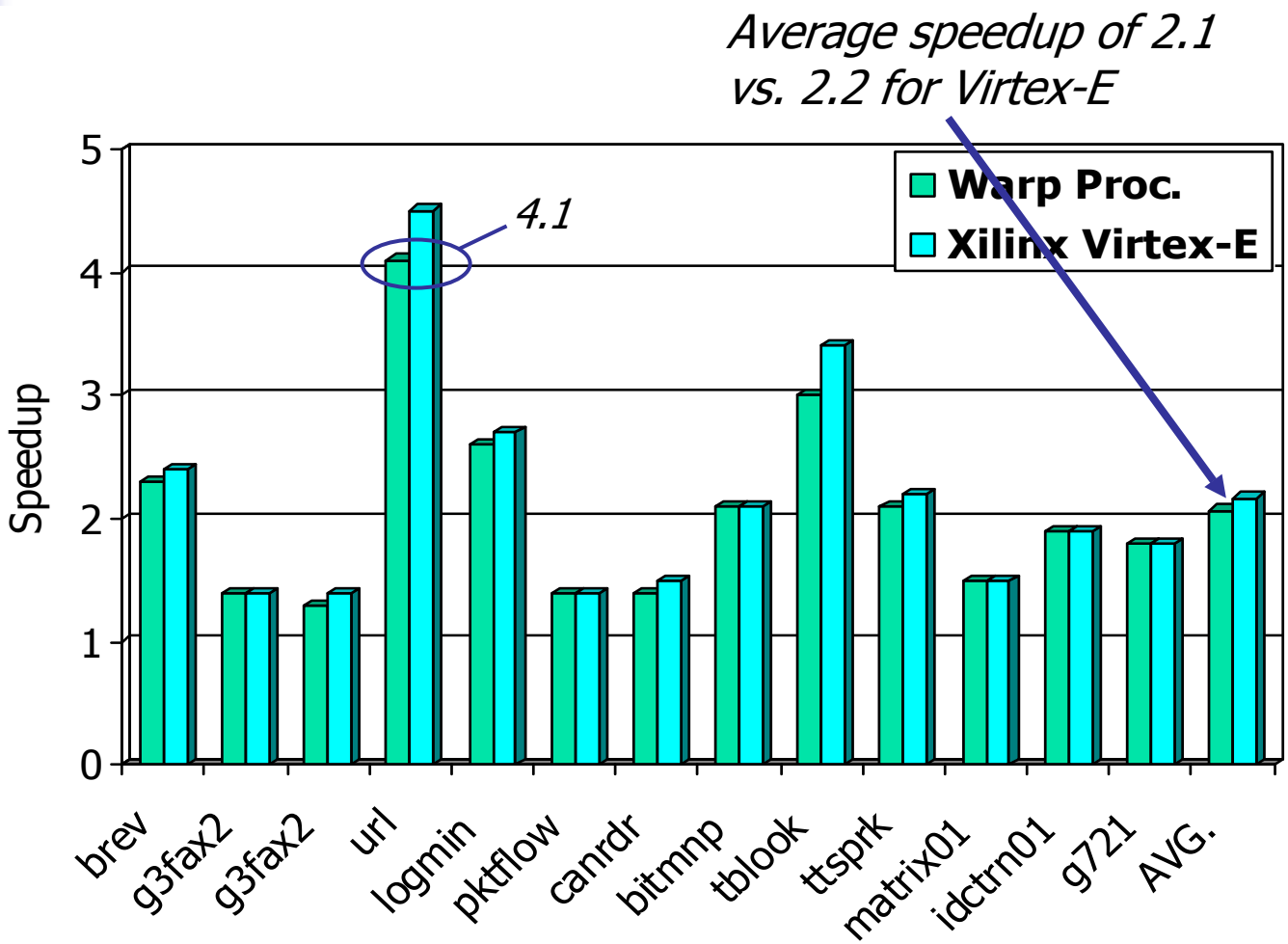
- Warp Processor
  - 75 MHz ARM7 processor
  - Configurable logic fabric with fixed frequency of 60 MHz
  - Used dynamic partitioning CAD tools to map critical region to hardware
    - Executed on an ARM7 processor
    - Active for roughly 10 seconds to perform partitioning
- Versus traditional HW/SW Partitioning
  - 75 MHz ARM7 processor
  - Xilinx Virtex-E FPGA (executing at maximum possible speed)
  - Manually partitioned software using VHDL
  - VHDL synthesized using Xilinx ISE 4.1 on desktop





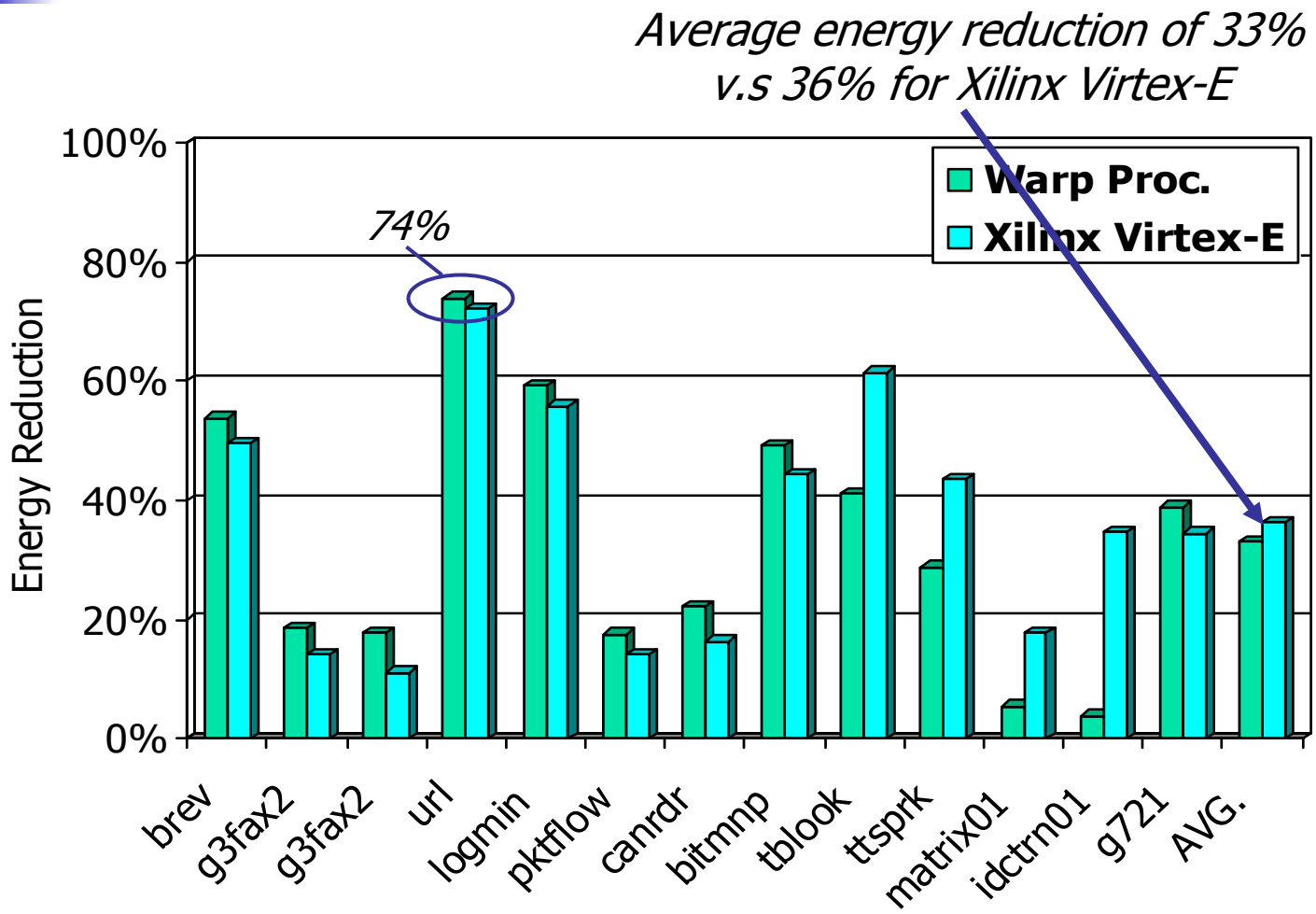
Warp Processors: Initial Results

Performance Speedup



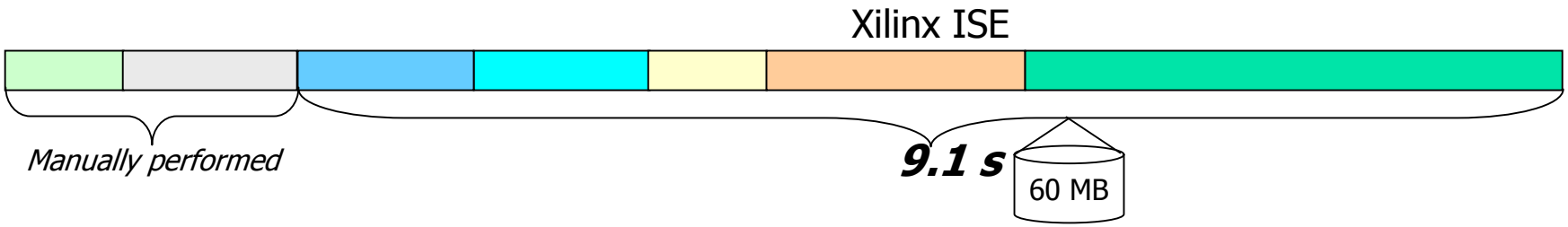
Warp Processors: Initial Results

Energy Reduction

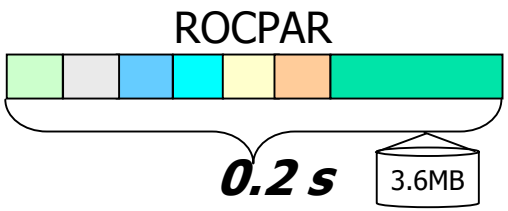


Warp Processors

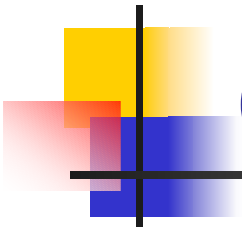
Execution Time and Memory Requirements (on PC)



46x improvement



On a 75Mhz ARM7: only 1.4 s

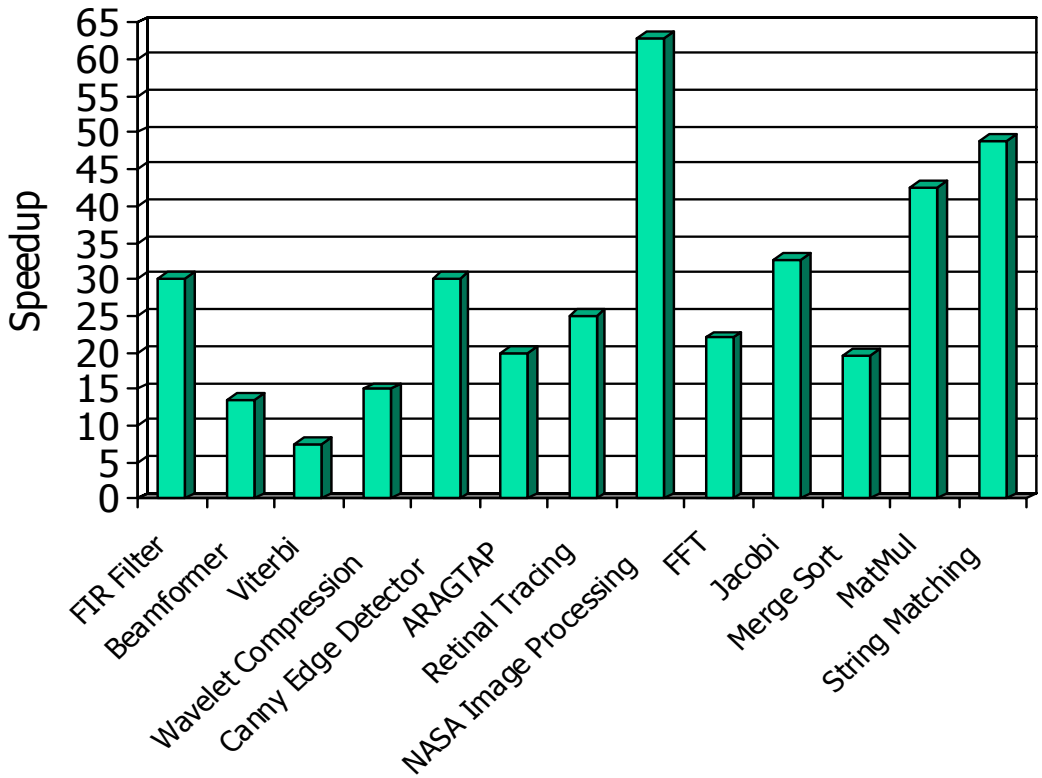


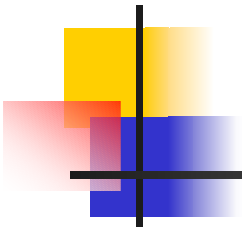
Current/Future Work

- Extending Warp Processors
  - Multiple software loops to hardware
  - Handling custom sequential logic
  - Better synthesis, placement, routing
- **JIT FPGA Compilation**
  - Idea: standard binary for FPGA
    - Similar benefits as standard binary for microprocessor
      - e.g., portability, transparency, standard tools

Future Directions

- Warp Processors may achieve speedups of 10x to 1000x
  - Hardware/software partitioning shows tremendous speedup
- Working to improve tools/fabric towards these results





Publications

- A Configurable Logic Architecture for Dynamic Hardware/Software Partitioning, R. Lysecky and F. Vahid, Design Automation and Test in Europe Conference (DATE), February 2004.
- Frequent Loop Detection Using Efficient Non-Intrusive On-Chip Hardware, A. Gordon-Ross and F. Vahid, ACM/IEEE Conf. on Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2003; to appear in special issue "Best of CASES/MICRO" of IEEE Trans. on Comp.
- A Codesigned On-Chip Logic Minimizer, R. Lysecky and F. Vahid, ACM/IEEE ISSS/CODES conference, 2003.
- Dynamic Hardware/Software Partitioning: A First Approach. G. Stitt, R. Lysecky and F. Vahid, Design Automation Conference, 2003.
- On-Chip Logic Minimization, R. Lysecky and F. Vahid, Design Automation Conference, 2003.
- The Energy Advantages of Microprocessor Platforms with On-Chip Configurable Logic, G. Stitt and F. Vahid, IEEE Design and Test of Computers, November/December 2002.
- Hardware/Software Partitioning of Software Binaries, G. Stitt and F. Vahid, IEEE/ACM International Conference on Computer Aided Design, November 2002.