A Coarse-Grain FPGA Overlay for Executing Data Flow Graphs

Davor Capalija and Tarek Abdelrahman

University of Toronto

CARL 2012

FPGAs as Accelerators

- Emergence of FPGAs as accelerators
  - Gaining popularity in heterogeneous HPC
  - Raw computational power rivals CPUs and GPUs
- FPGA strengths
  - Massive parallelism (1 million LUTs)
  - Customization of computation and interconnect
  - Reconfiguration
- Exponential increase in FPGA resource count will continue to 2030 (FPGA 2012 workshop)

FPGA "Programmability Wall"

- FPGA programming is much harder than software systems programming
- FPGA programming
  - HDL such as Verilog
  - High-level synthesis from C-like languages
- Knowledge of low-level FPGA fabric and tools
  - Timing (Fmax), resources (LUTs), place and route
- FPGA synthesis takes hours, even days
- The iterative nature of this process makes it even harder

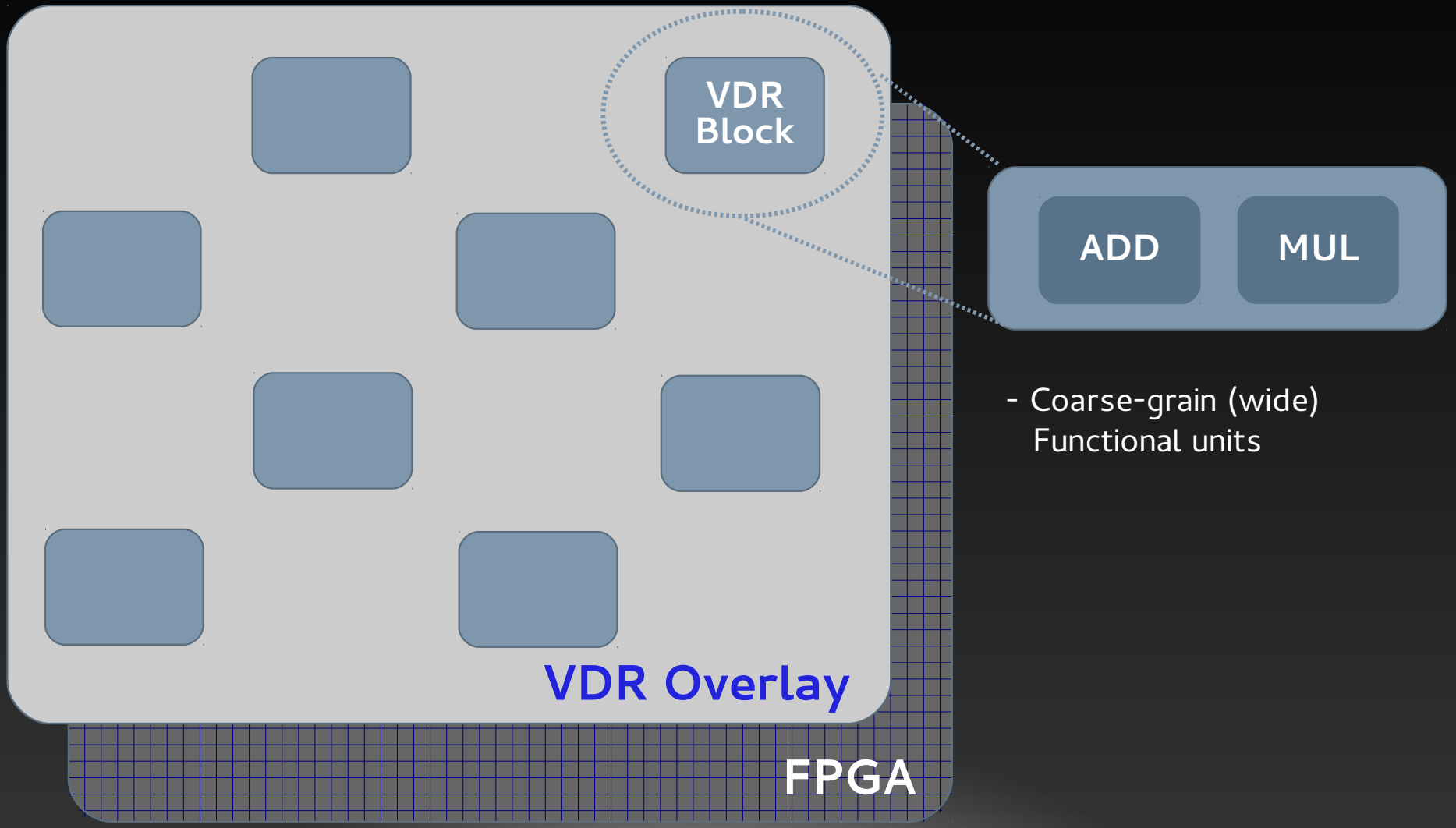
Our Goal

- Hide FPGA details (timing, LUTs)
  - Raise the level of abstraction to data flow graphs
- Radically reduce compile time
  - Remove synthesis from the tool chain
- Keep FPGA strengths
  - Massive parallelism
  - Customization of computation and interconnect
  - Reconfiguration

Our Approach: VDR Overlay

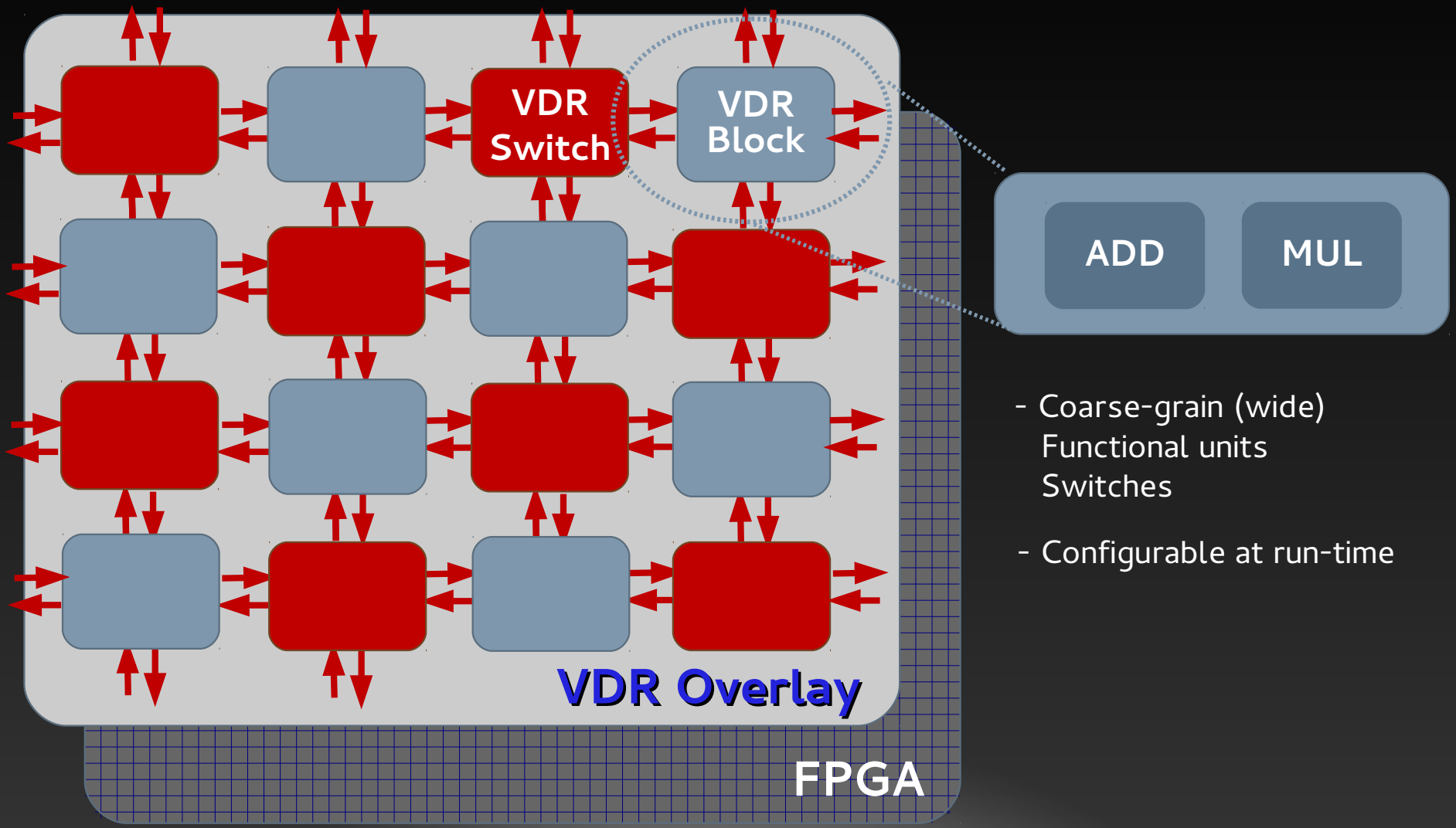
- **V**irtual **D**ynamically **R**econfigurable Overlay
- Pre-synthesized FPGA circuit designed to execute data flow graphs
- It can be **configured** at **run-time** to implement any data flow graph

VDR Overlay Architecture



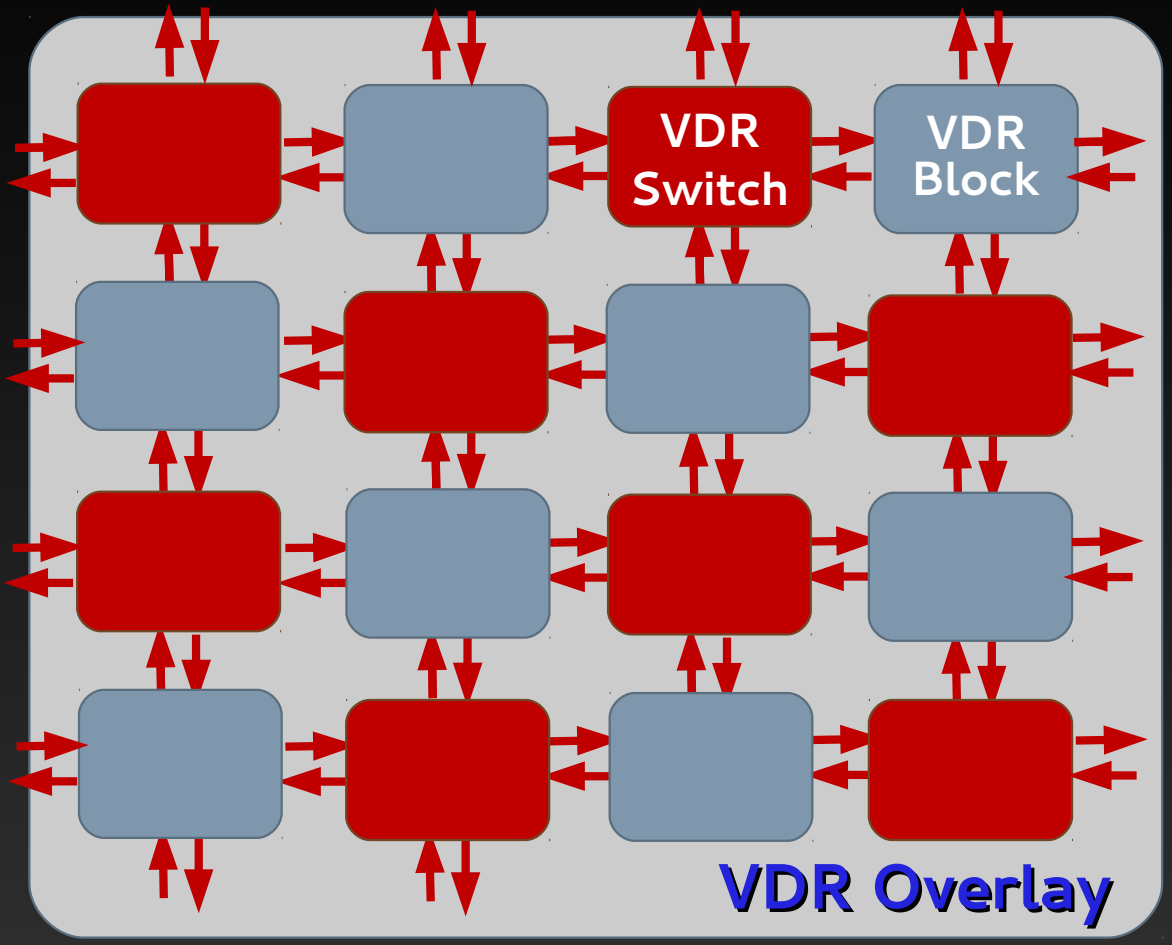
- Coarse-grain (wide) Functional units

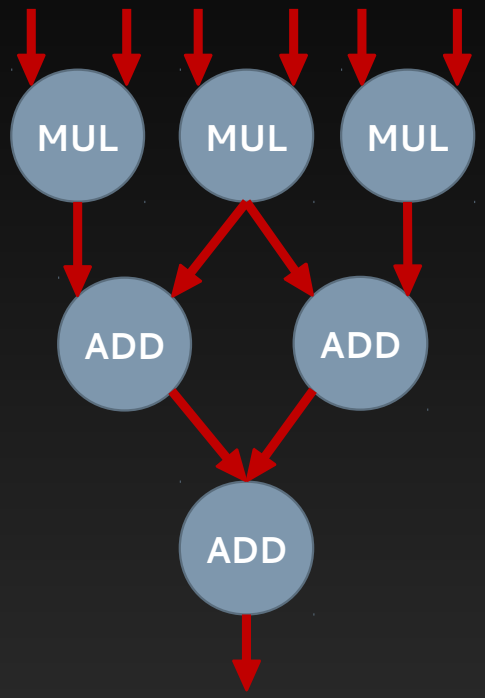
VDR Overlay Architecture



- Coarse-grain (wide)
  Functional units
  Switches
- Configurable at run-time

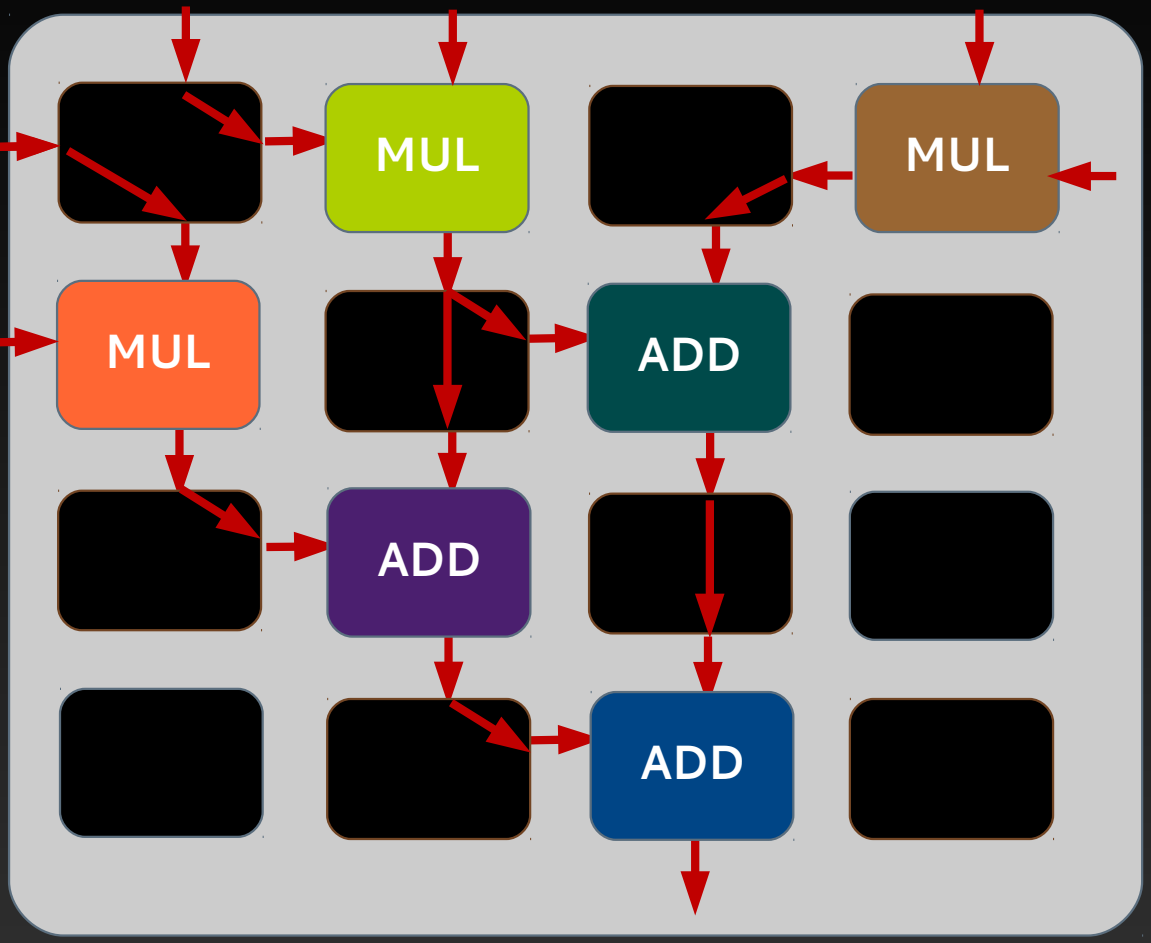
DFG-to-VDR mapping

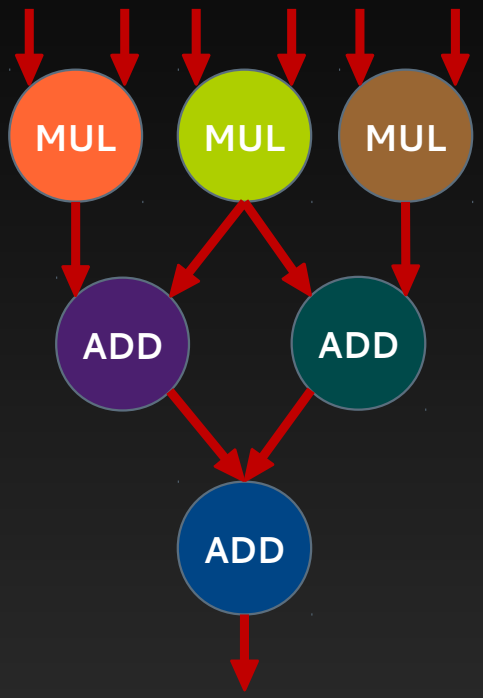




Data Flow Graph (DFG)

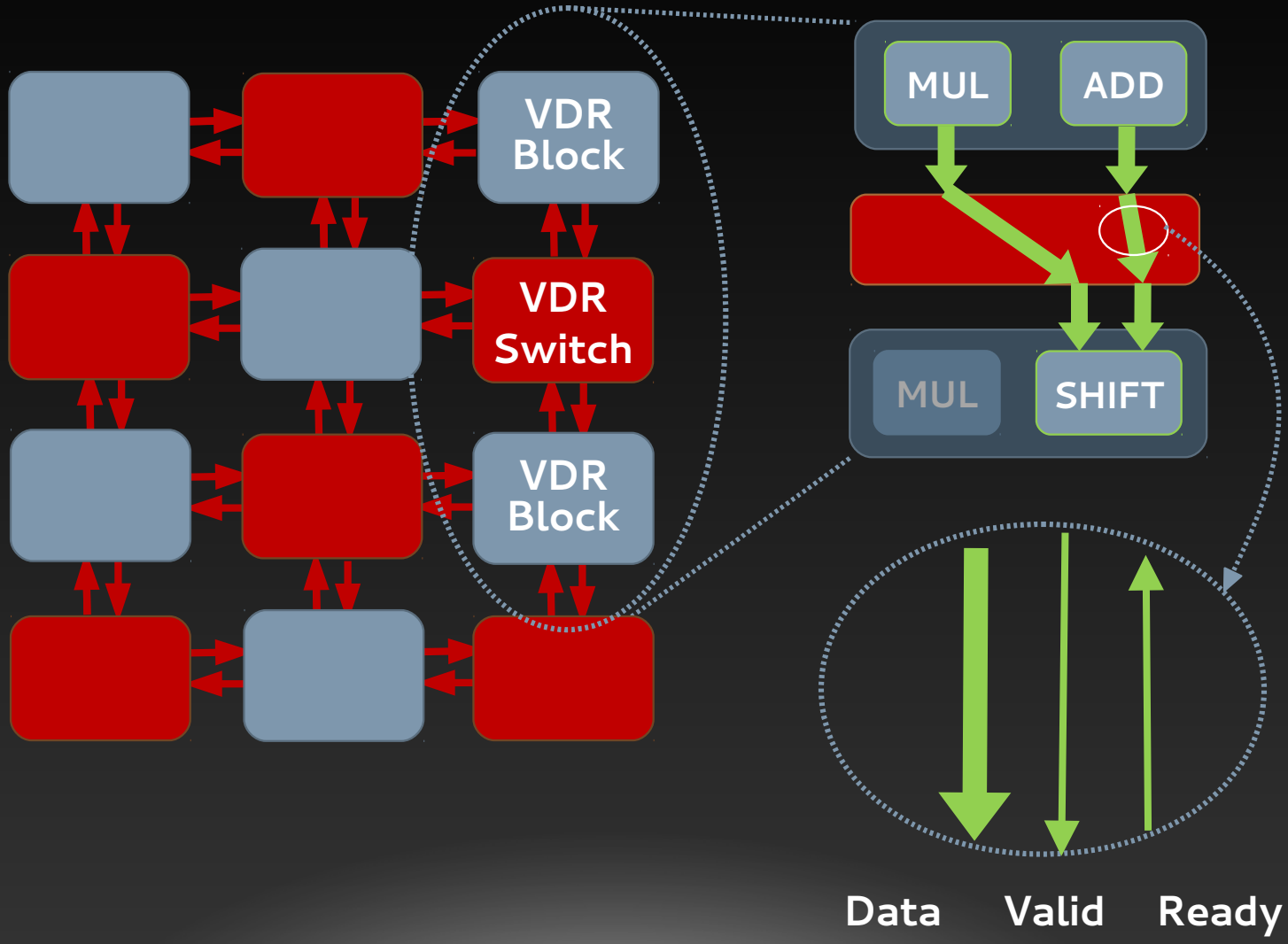
DFG-to-VDR mapping





Data Flow Graph (DFG)

VDR Overlay Synchronization



VDR Overlay – Benefits

- Direct DFG execution
  - Bridges the gap between SW and HW
- Rapid compile time (DFG-to-VDR)
  - Seconds
- Rapid reconfiguration time (DFG-to-VDR)
  - Several microseconds
- Can be targeted by a static compiler or a dynamic run-time system

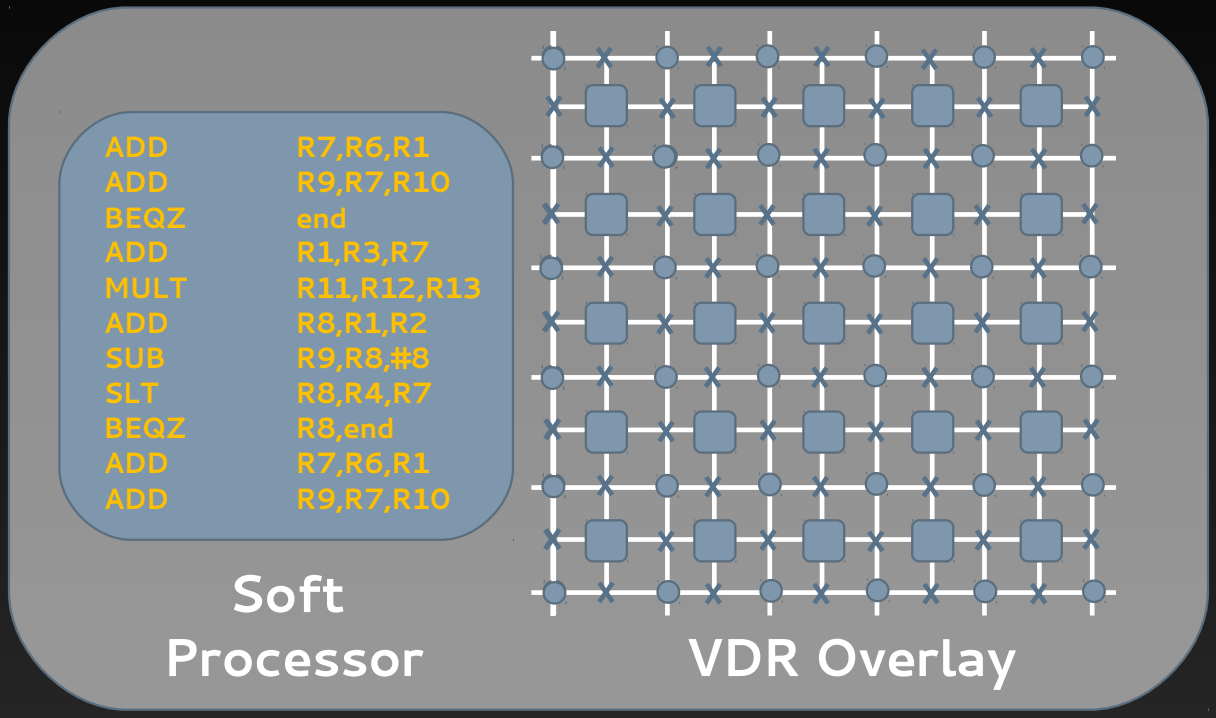
VDR Overlay – Benefits

- Virtualization
  - Can be implemented on any commodity FPGA
- DFG portability
  - Across vendors, device families and sizes
- DFG relocation within VDR
  - Easier dynamic and partial reconfiguration

VDR Overlay – Research Issues

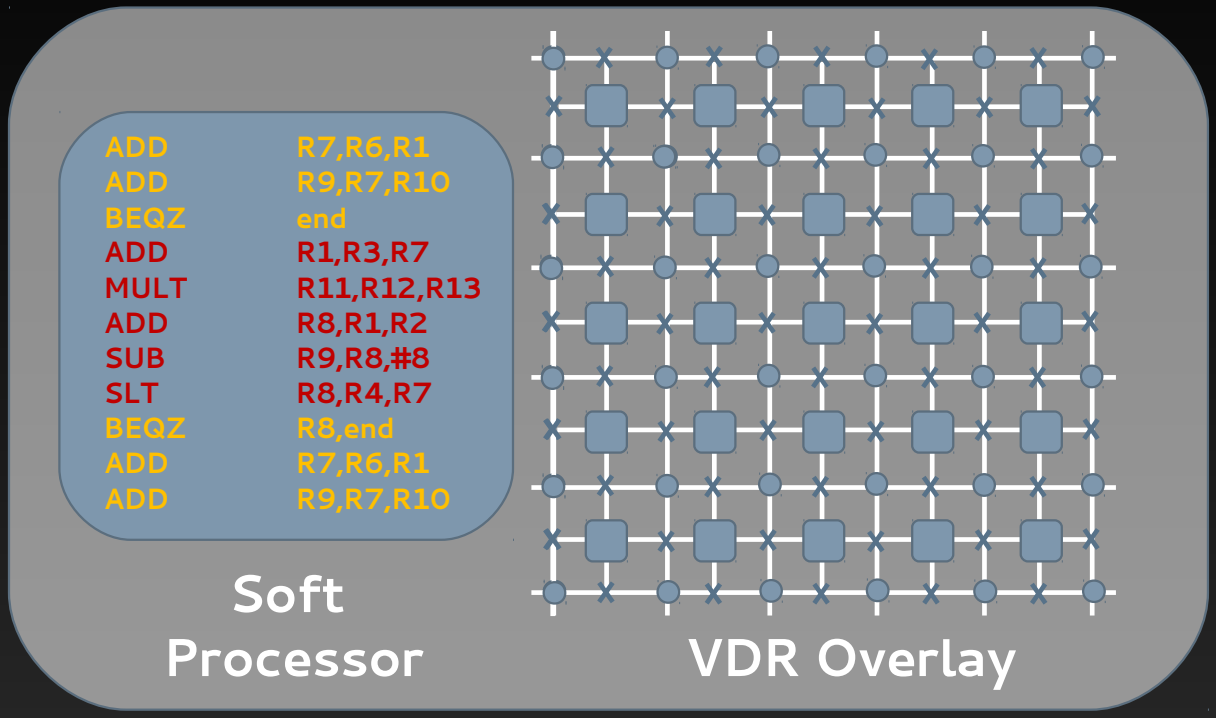
- VDR Block architecture
  - Generic FUs, custom FUs
  - Homogenous, heterogeneous
- Interconnect topology
  - e.g. 2-NN, 4-NN, 8-NN
- DFG-to-VDR mapping algorithm
- High performance
- Low FPGA resource overhead
- Scalability – large overlays

Feasibility Study
– Dynamic Soft Processor Acceleration



Commodity FPGA

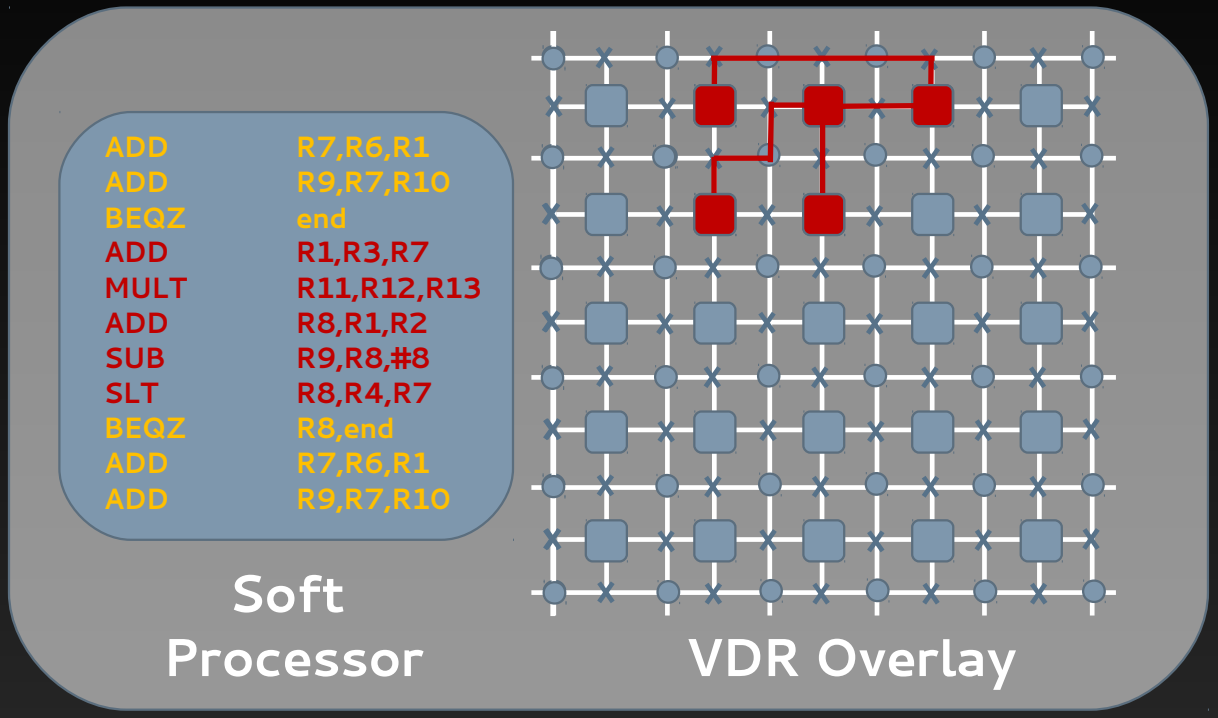
Feasibility Study
– Dynamic Soft Processor Acceleration



Commodity FPGA

- Monitor execution to detect a hot section of code

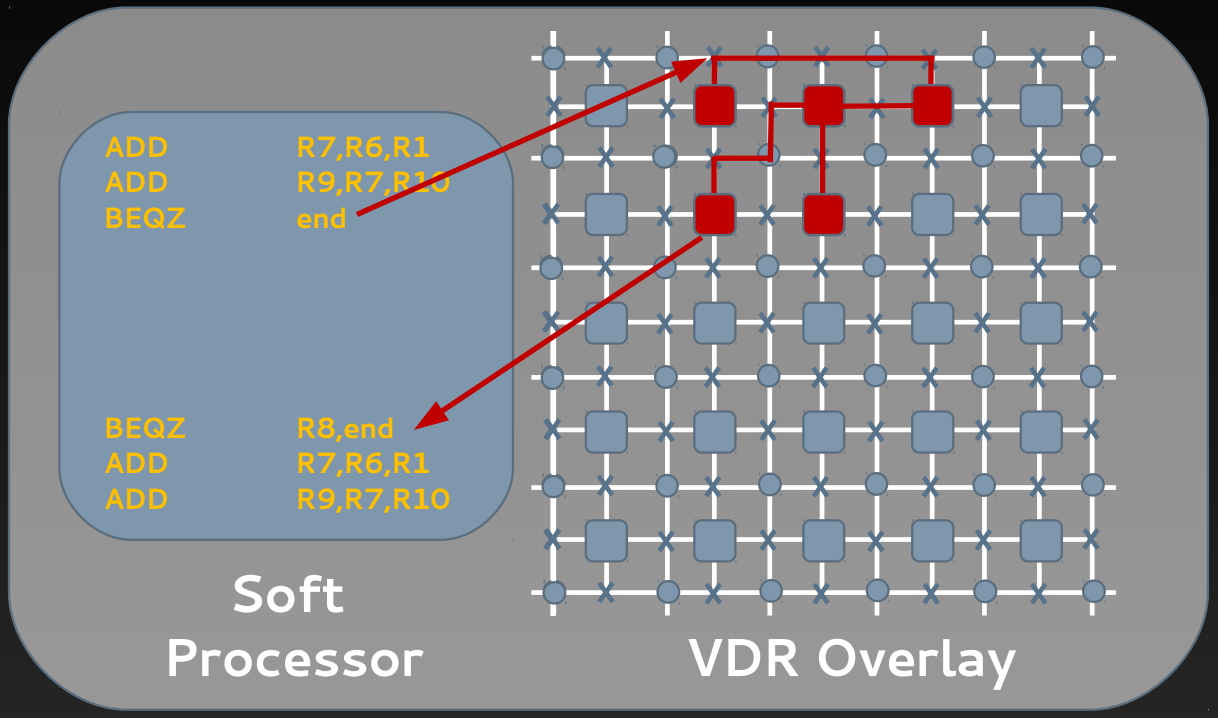
Feasibility Study
– Dynamic Soft Processor Acceleration



Commodity FPGA

- Monitor execution to detect a hot section of code
- Create a DFG and map it to the VDR

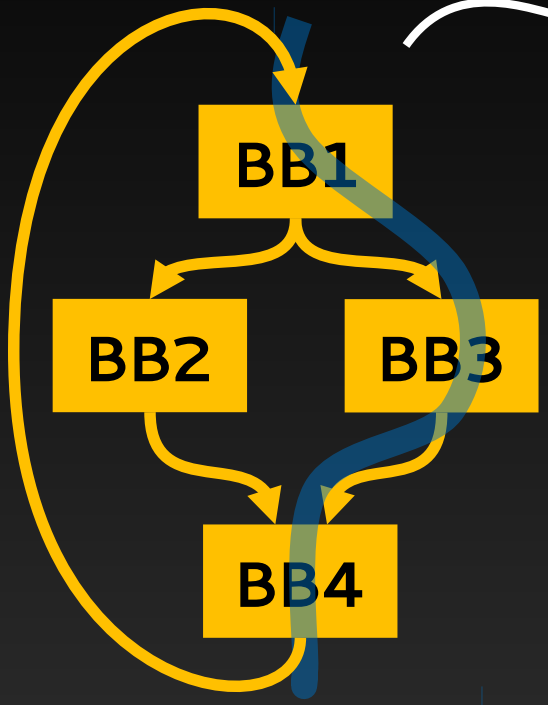
Feasibility Study
– Dynamic Soft Processor Acceleration



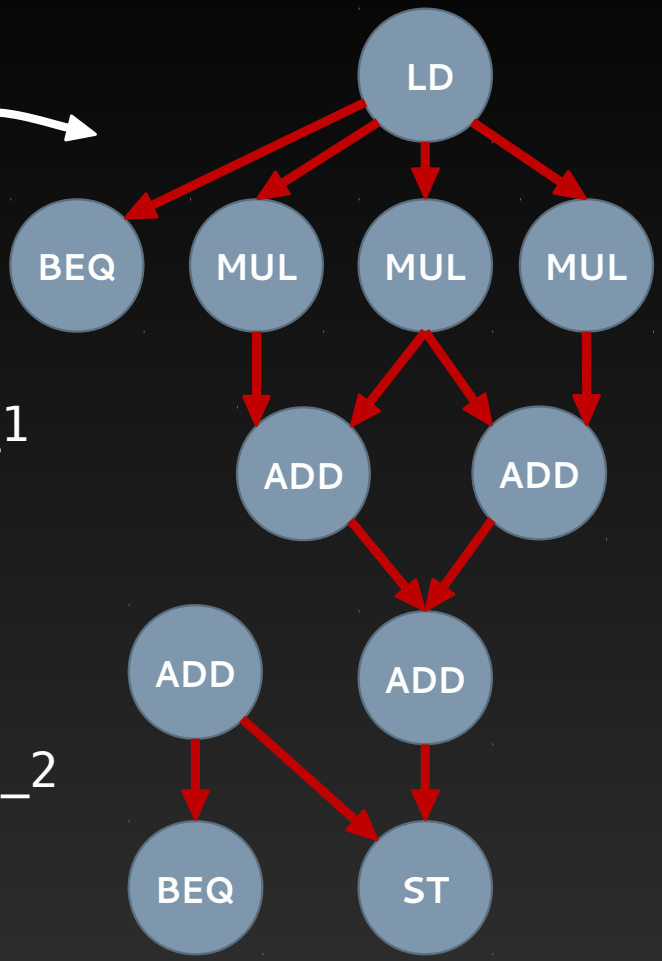
Commodity FPGA

- Monitor execution to detect a hot section of code
- Create a DFG and map it to the VDR
- Configure the VDR
- Execute the DFG on the VDR

Traces and Data Flow Graphs




```
ld      r2,0(r1)
mul     r3,r3,r2
mul     r4,r4,r2
mul     r5,r5,r2
beq     r2,r0,exit_1
add     r6,r3,r4
add     r7,r4,r5
add     r8,r6,r7
addi    r9,r9,4
st      r8,0(r9)
beq     r9,r15,exit_2
```

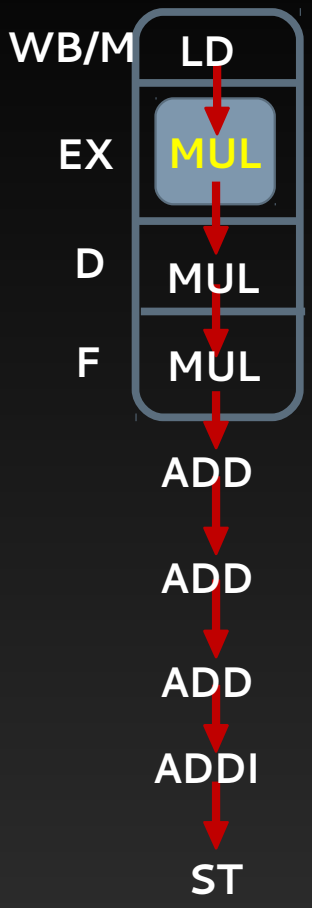


Trace = BB1,BB3,BB4

Straightline section of code

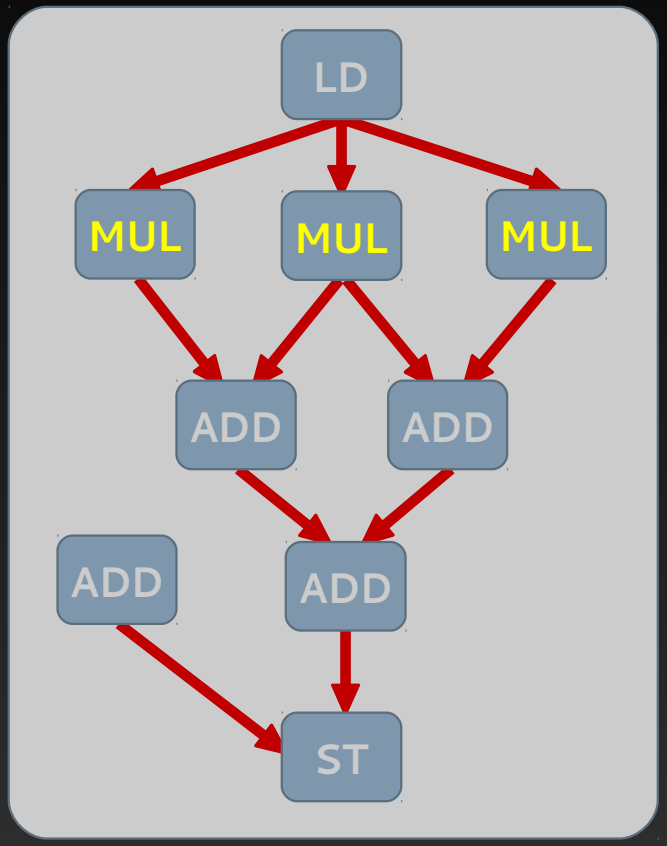
DFG

Acceleration Model



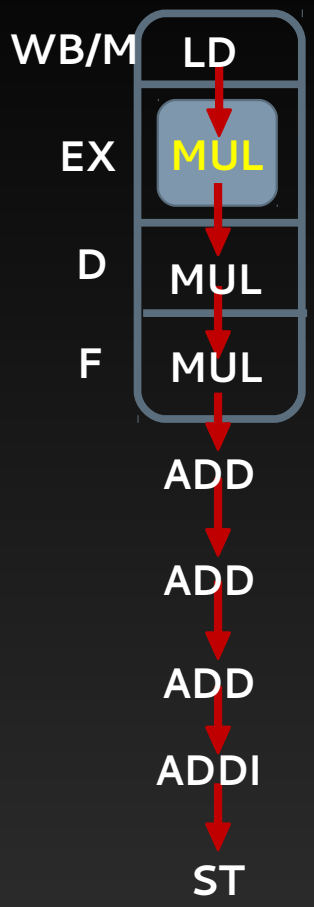
Soft processor

Instruction-level parallelism



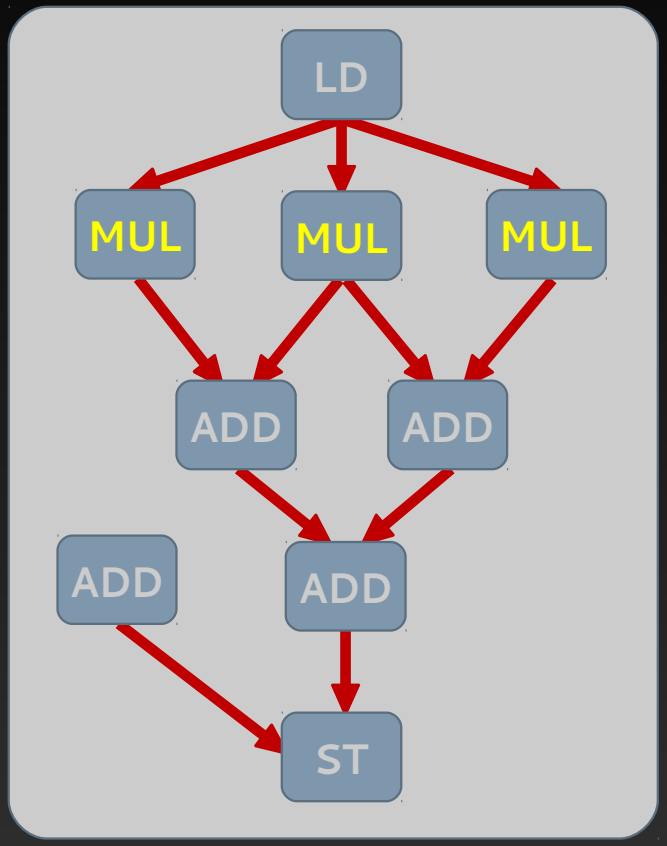
VDR Overlay

Acceleration Model



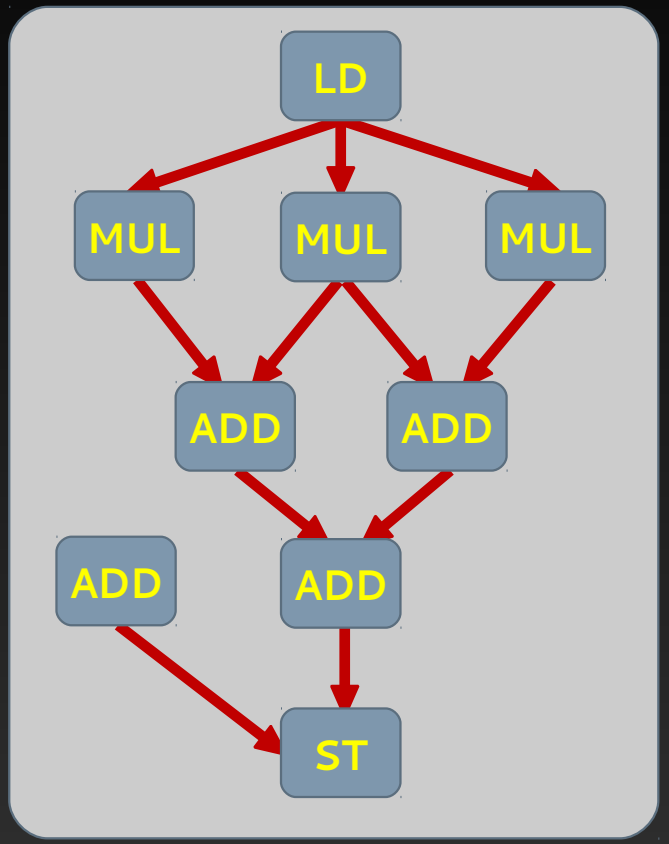
Soft processor

Instruction-level parallelism



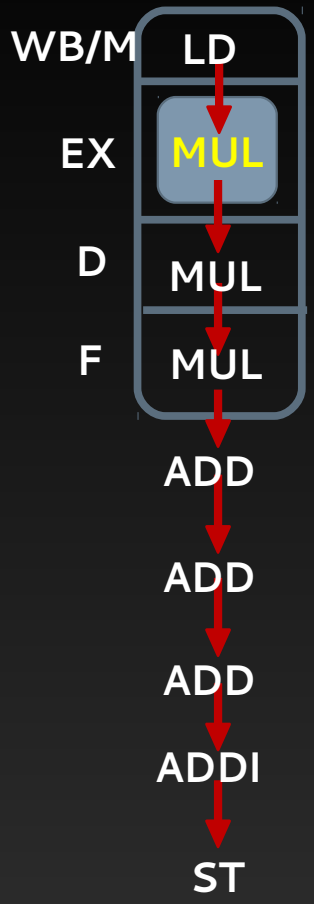
VDR Overlay

Pipeline parallelism



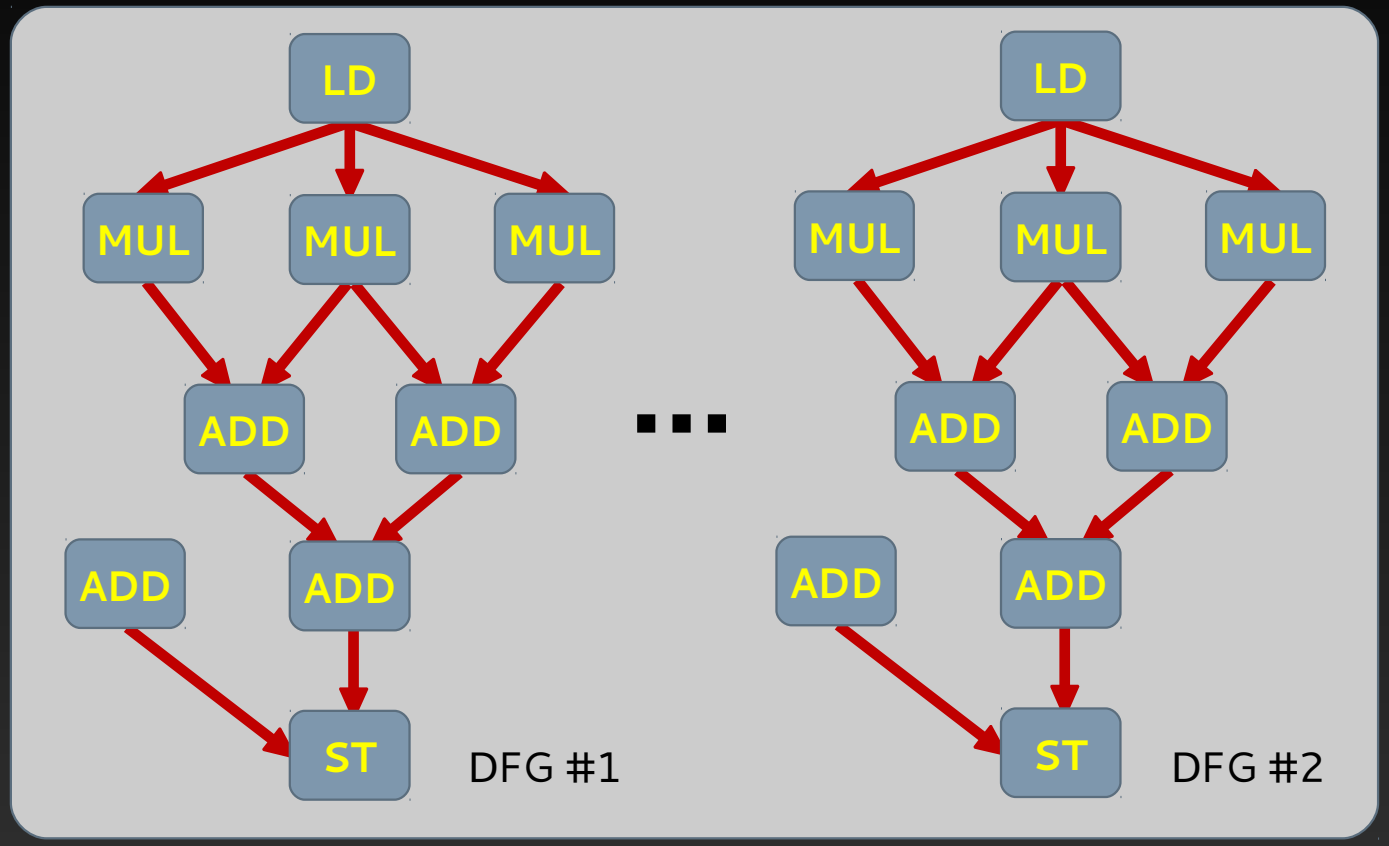
VDR Overlay

Acceleration Model



Soft processor

DFG-level parallelism (replication)



VDR Overlay

Feasibility Study

- A system around the Nios II soft processor
  - DE3 board with Altera Stratix III
- RGB2YIQ benchmark from the EEMBC suite
  - A single trace (single DFG)
- A VDR overlay
  - Application-tailored FUs and interconnect topology

Feasibility Study

- Measured
  - Overlay overhead
  - Benchmark speedup
- Compared
  - Nios II
  - VDR overlay
  - Hardwired overlay
- Hardwired overlay
  - Same units as the overlay
  - Not configurable - switches removed

VDR Overlay Overhead

Version name	Fmax	ALUTs	ALUTs relative
Nios II	290	1483	1
VDR	172	4753	3.2
Hardwired	286	2978	2

- VDR overlay resource breakdown
  - 60% units and 40% switches

Benchmark Acceleration

Version name	DFG replicas	Ports/Mem (total ports)	Speedup
Nios II	--	--	1
VDR-1	1	1 (1)	3.31
VDR-2	1	2 (2)	4.85
VDR-2 x2	2	2 (4)	8.82
HDW-1	1	1 (1)	5.87
HDW-2	1	2 (2)	8.87

Overlay Execution Breakdown

- For a 640x480 frame: ~11 M cycles

Execution Component	%
Trace detection	0.06
Trace-to-VDR mapping	9.19
Overlay reconfiguration	0.01
Register initialization	0.01
Overlay execution	90.73

Current Work

- Regular generic overlay
  - Generic integer FUs
  - Topology - 4-NN with pass-through links
- High-throughput overlay
  - Scalability - from 2x2 up to 32x32
  - High Fmax – 400+ MHz
- Flexible mapping algorithm
  - Ability to map any trace/DFG
  - Place and route algorithm

Next Steps

- Memory interfaces
  - Streaming, FIFOs, caches, scratch-pad memories
- Application customized FUs
  - Integer, fixed-point, floating-point
  - Specialized FUs
- Application customized topology
  - e.g. 4-NN with hops, 6-NN

Thank You