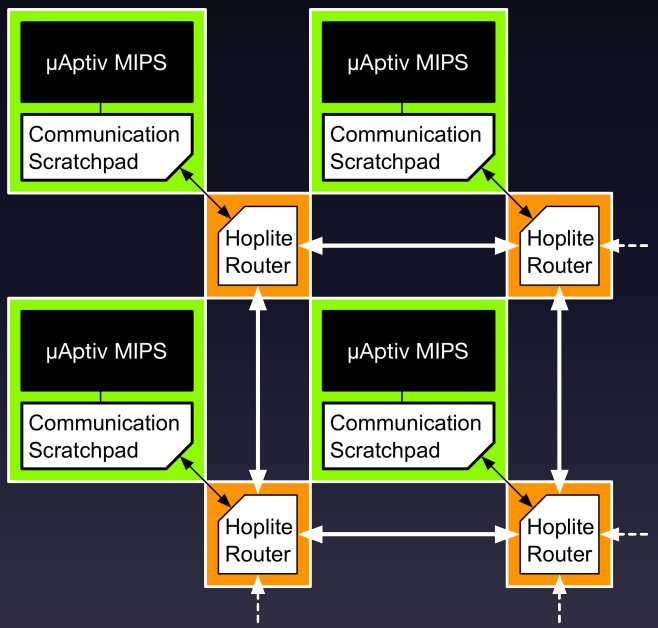
120-core microAptiv MIPS Overlay for the Terasic DE5-NET FPGA board

Chetham Kumar H B, Prashant Ravi, Gourav Modi,
Nachiket Kapre
nachiket@uwaterloo.ca

Claim

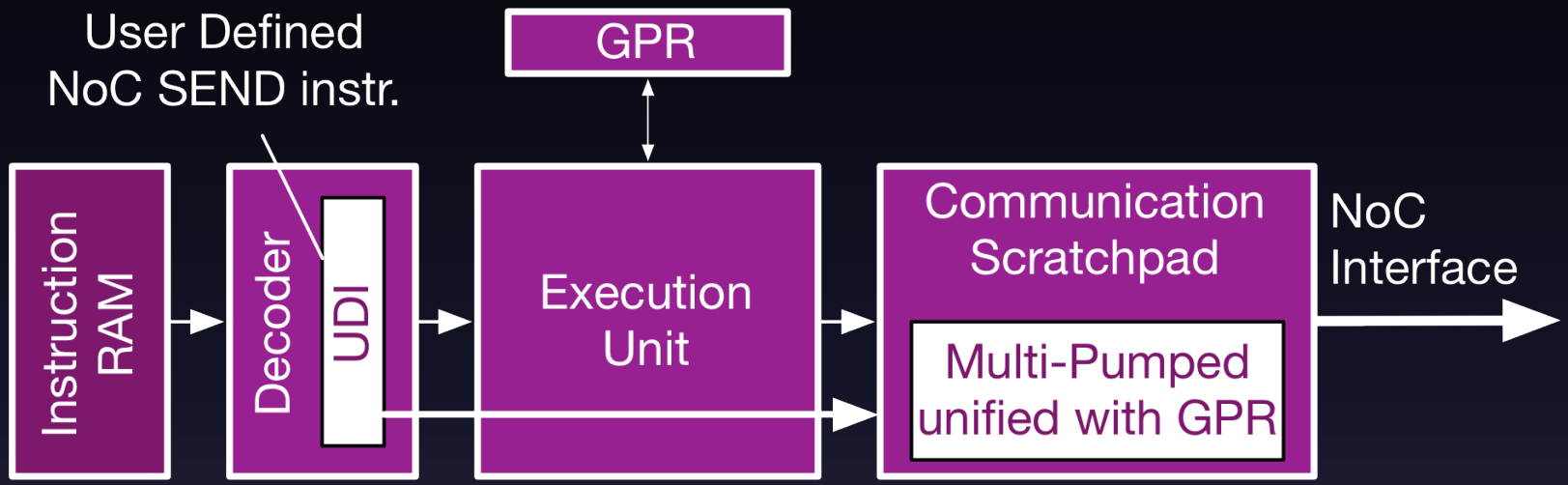
- Fit 120-core MIPS μaptiv processor on Stratix V (DE5-Net board)
  - 2.4K ALMs, 2K FFs, 2 DSPs, 3 M20Ks, 94 MHz
  - 3–4$\times$ smaller than baseline implementation
- Key Idea:
  - FPGA-specific modifications to RTL
  - Repeal and replace cache hierarchy with scratchpads
  - Add message-passing instructions + software API

System-Level View



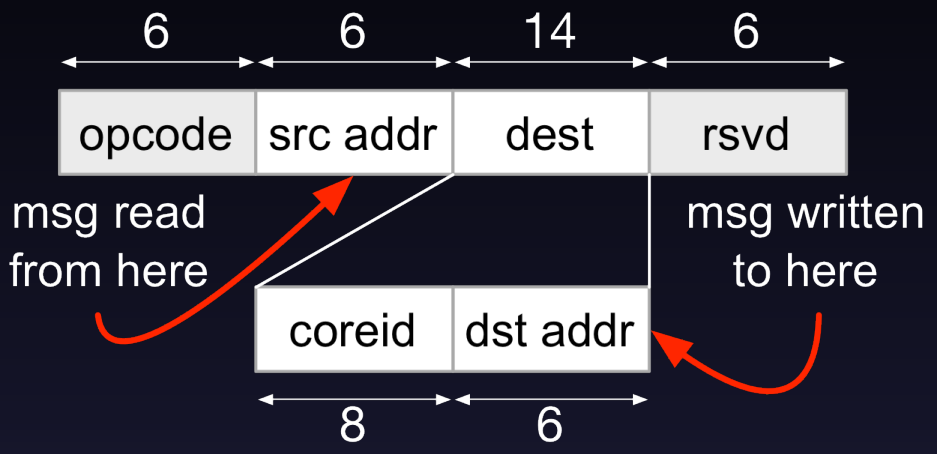
- Add packet-switched NoC router to each processor
- Split off the communication scratchpad
- Introduce a UDI (User Defined Instruction)
- Prune FPGA logic utilization

Processor Modification



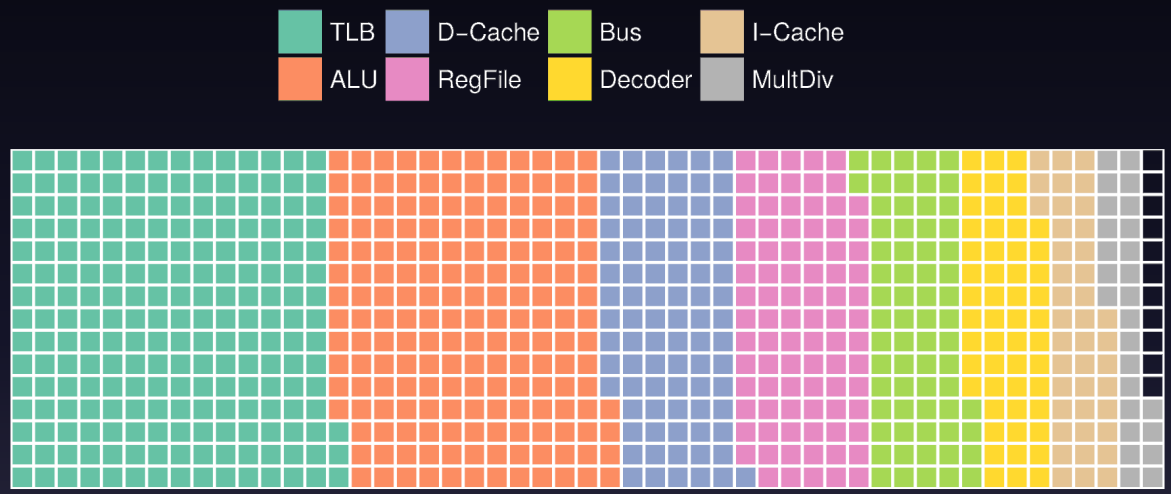
- Modify register file to include a *communication scratchpad*
- Only message-passing instruction access region
- Modify decoder to interpret the UDI format

UDI Instruction Format



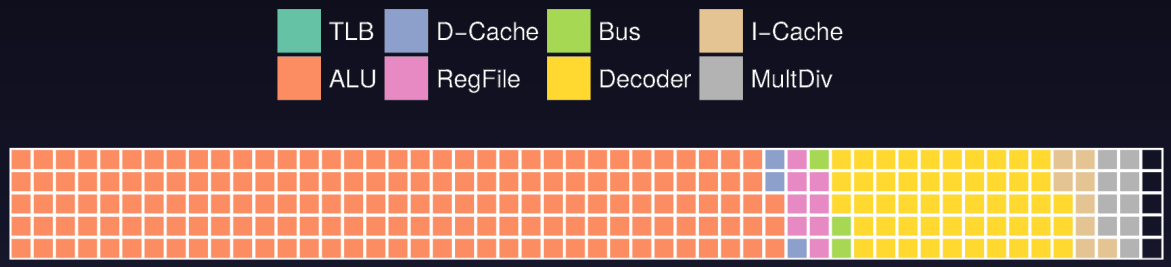
- Only support **SEND**-style messages
- Supply **src addr** – in local comm. scratchpad
- Also provide **coreid** + **dst addr** for identifying target
  - **coreid** used by NoC to route message to destination
- Addresses embedded in instruction – construction cost per packet

FPGA Pruning



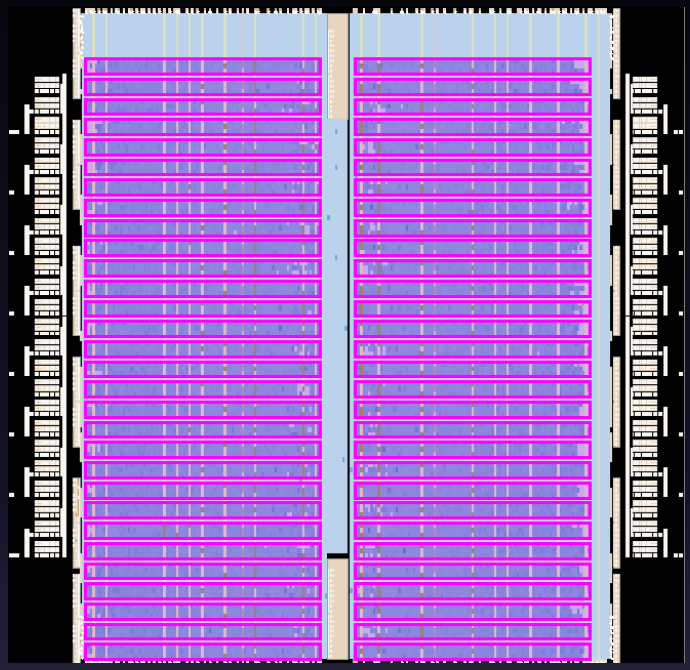
- Dominated by cache hierarchy (**TLB** + **D-Cache**)
- Divider+Multiplier **MultDiv** does not use DSP blocks
- Register File **RegFile** does not use M20Ks
- Extraneous logic for SoC environment

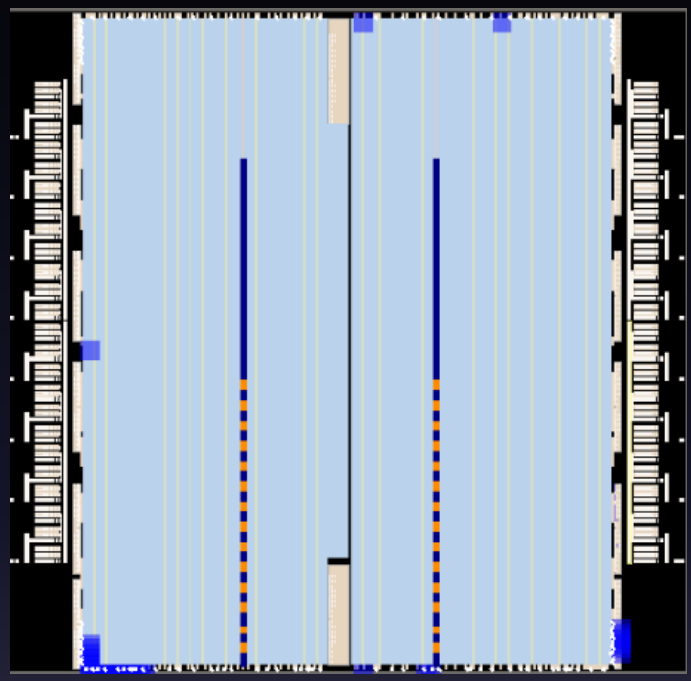
FPGA Pruning



- Dominated by cache hierarchy (**TLB** + **D-Cache**)
- Divider+Multiplier **MultDiv** does not use DSP blocks
- Register File **RegFile** does not use M20Ks
- Extraneous logic for SoC environment

FPGA Mapping





- **Original**: 8.1K LUTs + 8.4K FFs + 0 DSPs + 17 M20Ks
- **Modified**: 2.4K LUTs + 2.1K FFs + 2 DSPs + 3 M20Ks
- 3–4× lower resource use (better DSP use)
  - NoC costs 100 ALMs + FFs per router
- 94 MHz critical path in Decode+ALU

Programming Model

```
// Loop over non-zeros, to compute Ax=b
for(m=0; m<M; m++) {
 b[m]=0;
 int edges = a_index[m+1]-a_index[m];
 for(n=0; n<edges; n++) {
  b[m] += a_nz[a_index[m]+n]
    *x[col_index[a_index[m]+n]];
 }
}
```

- Exploit user-programmable (UDI) `mips_udi_i_nv` C function
- Message passing format packs src + dest information
- Code spends 2–3 lines generating the instruction code
- The formatted bits passed as argument of `mips_udi_i_nv` function

Programming Model

```
// Send vector values to cores that need it
mips_udi_i_nv(0x09,0x00000); // UDI Start
for(int m=0; m<M; m++) {
 src = &b[m];
 int edges = s_index[m+1]-s_index[m];
 pkt = src<<14;
 for(int n=0; n<edges; n++) {
  // construct packet
  dest = &x[invcol_index[s_index[m]+n]];
  pkt |= dest<<5;
  // send packet
  mips_udi_i_nv(0x08,pkt);
 }
}
mips_udi_i_nv(0x0A,0x00000); // UDI End
```

- Exploit user-programmable (UDI) `mips_udi_i_nv` C function
- Message passing format packs src + dest information
- Code spends 2–3 lines generating the instruction code
- The formatted bits passed as argument of `mips_udi_i_nv` function

Conclusions

- Fit a 120-core μaptiv MIPS multi-core on a Stratix V DE5-Net FPGA Board
  - FPGA-specific logic pruning + optimization = 3–4$\times$ fewer ALMs + FFs + M20Ks, better use of DSPs for multiplier implementation
  - Lightweight NoC integration + MIPS UDI instruction formatting for communication support
- **Future Work**:
  - Smaller + faster MIPS core starting with an MCU baseline
  - Software APIs using the **mips_udi_i_nv** UDI function