Dyser

(a) Switches
 FU Functional unit
 Parallelism specialization enhancements

(b) Dyser Region
 Configuration
Coarse-grained Block Diagram

- Scratchpad Slices
- L1 Cache
- L2 Cache Slice
Performance/Area

- AES
- DMM
- Flow Classifier
- FFT
- Graph500
- K-means
- KMP Search
- Merge Sort
- SHA-256
- Mean

Performance Ratio
if (incoming > cur)
    send(cur); cur := incoming;
else
    send(incoming);
Merge Sort

for x = 1..NPASSES
    for y = 1..k
        // control loop

if (listA > listB ||
    (listA.finished && !listB.finished))
    send(listB);
else if (!listA.finished)
    send(listA);
Merge sort worker – PC+RegQueue

check_a: beqz %in0.notEmpty, check_a // listA
check_b: beqz %in1.notEmpty, check_b // listB
check_o: beqz %out0.notFull, check_o // outList
  beq %in0.tag, EOL, a_done
  beq %in1.tag, EOL send_a
  cmp.lt %r0, %in0.first, %in1.first
  bnez %r0, send_a
send_b: enq %out0, %in1.first
  deq %in1
  jump check_a
send_a: enq %out0, %in0.first
  deq %in0
  jump check_a
a_done: beq %in1.first, EOL, done
  jump send_b
done: deq %in0
  deq %in1
  return;

<table>
<thead>
<tr>
<th>Static Insts</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Insts/Iteration</td>
<td>10</td>
</tr>
<tr>
<td>Avg Branches/Iteration</td>
<td>7</td>
</tr>
</tbody>
</table>
### PC-based design alternatives

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC (Baseline)</td>
<td>PEs use program counters, communicate using shared-memory queues.</td>
</tr>
<tr>
<td>+RegQueue</td>
<td>Expose register-mapped queues to ISA, test via active polling.</td>
</tr>
<tr>
<td>+FusedDeq</td>
<td>Destructive read of queue registers without separate instructions.</td>
</tr>
<tr>
<td>+RegQSelect</td>
<td>Allow indirect jump based on register queue status bits.</td>
</tr>
<tr>
<td>+RegQStall</td>
<td>Issue stalls on queue input/output registers without special instructions.</td>
</tr>
<tr>
<td>+QMultiThread</td>
<td>Stalling on empty/full queue yields thread.</td>
</tr>
<tr>
<td>+Predication</td>
<td>Predicate registers that can be set using queue status bits.</td>
</tr>
<tr>
<td>+Augmented</td>
<td>ISA augmented with all of the above features except +QMultiThread.</td>
</tr>
</tbody>
</table>
Merge sort worker – PC+Augmented

```
start:    beq  %in0.tag, EOL, a_done
          beq  %in1.tag, EOL, send_a
          cmp.ge p2, in0.first, in1.first
send_b:  (p2) enq  %out0, %in1.first (deq %in1)
send_a:  (!p2) enq  %out0, in0.first (deq %in0)
          jump  start
a_done:  cmp.ne p2, %in1.first, EOL
          (p2) jump  send_b
          nop  (deq %in0, deq %in1)
          return;
```

<table>
<thead>
<tr>
<th>Static Instrs</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Instrs/Iter (Issued)</td>
<td>6</td>
</tr>
<tr>
<td>Avg Instrs/Iter (Committed)</td>
<td>5</td>
</tr>
<tr>
<td>Avg Branches/Iter</td>
<td>3</td>
</tr>
<tr>
<td>Speedup vs PC+RegQueue (Fig 3)</td>
<td>1.4x</td>
</tr>
</tbody>
</table>
Guarded Actions

- Program consists of rules that may perform computations and read/write state.
- Each rule specifies conditions (guard) under which it is allowed to fire.
- Separates description and execution of data (rule body) from control (guards).
- A scheduler is generated (or provided by hardware) that evaluates the guards and schedules rule execution.
- Sources of Parallelism:
  - Intra-Rule parallelism
  - Inter-Rule parallelism
  - Scheduler overlap with Rule execution
  - Parallel access to state

```plaintext
rule X (A > 0 && B != C) {
    A <= B + 1;
    B <= B - 1;
    C <= B * A;
}
rule Y (...) {...}
rule Z (...) {...}
```
doCheck:
  when (!p0 && %in0.tag != EOL
       && %in1.tag != EOL) do
    cmp.ge p1, %in0.data, %in1.data (p0 := 1)
sendA:
  when (p0 && p1) do
    enq %out0, %in0.data (deq %in0, p0 := 0)
sendB:
  when (p0 && !p1) do
    enq %out0, %in1.data (deq %in1, p0 := 0)
drainA:
  when (%in0.tag != EOL && %in1.tag == EOL) do
    enq %out0, %in0.data (deq %in0)
drainB:
  when (%in0.tag == EOL && %in1.tag != EOL) do
    enq %out0, %in1.data (deq %in1)
bothDone:
  when (%in0.tag == EOL && %in1.tag == EOL) do
    nop (deq %in0, deq %in1)

<table>
<thead>
<tr>
<th>Static Insts</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Insts/Iteration</td>
<td>2</td>
</tr>
<tr>
<td>Speedup vs PC+RegQueue (Fig 3)</td>
<td>5×</td>
</tr>
<tr>
<td>Speedup vs PC+Augmented (Fig 4)</td>
<td>3×</td>
</tr>
</tbody>
</table>
Scheduler

13

Priority Encoder

Trigger Resolution

Channel Status  Tags

Predicate Registers

P  P  P

P  P  P

P  P  P

P  P  P

Triggered Instruction

Execute

Datapath
Static Instructions

![Bar chart showing static instructions for different benchmarks and techniques.](chart.png)

- **AES**, **DMM**, **FFT**, **Flow Classifier**, **Graph-500**, **k-means**, **KMP Search**, **Merge Sort**, **SHA-256**, **Mean**

**Legend:**
- PC+RegQ
- PC+Augmented
- TIA
Dynamic Instructions
Control Instructions Evaluation

![Bar chart showing average instructions per iteration for different benchmarks and conditions.](chart.png)