A New Coarse-Grained Reconfigurable Architecture with Fast Data Relay and Its Compilation Flow

Enabling Hardware Support for Fast Data Relay

- **Architectural overview:**
  - Host CPU initializes the kernel computation on our FDR-CGRA.
  - DMA is in charge of system-level dataflow.
  - FDR-CGRA can focus on accelerating time-consuming kernels.

- **Processing elements:**
  - Support concurrent computation and communication with dedicated computing path and bypassing path for Fast Data Relay.
  - Use bypassing register file to temporary deposition of intermediate data from other PEs.

- **Wire-based companion channels:**
  - Inter-tile communication: direct connections between PEs on the tile boundaries.
  - Intra-tile communication: using companion channels.
    - Vertical: send scalar data from the sender PE to any PE in the same column.
    - Horizontal: send scalar data from the sender PE to any PE in the same row.

- **Routing region:**
  - Constructed as shown in Fig. 2 by the following steps:
    1. Line up all PEs in the two-dimensional computation grid along the horizontal and vertical axes.

- **Placement and routing for operations**
  - Rip-up & reroute
  - Resource constraints:
    - For IDCT, 1-tile (16-PE) FDR-CGRA can achieve IPC of 11.1 (gain 21%)

- **Fast Data Relay (FDR):**
  - FDR is a combination of hardware and compiler techniques to enable efficient multi-cycle data communication among Processing Elements (PEs) in a Coarse-Grained Reconfigurable Architecture (CGRA).
  - We name the proposed architecture FDR-CGRA.

- **Advantages of FDR-CGRA:**
  - A corner-to-corner transmission in a tile can be finished in two cycles;
  - Data communication can be done as a background operation without disturbing computation;
  - Source operands can have multiple copies in different PEs so that a dependent PE can find a local copy in its vicinity.

- **Higher Performance**
  - Benchmark programs are extracted from xvid 1.1.3 and JM7.5b
  - Preprocess with Low Level Virtual Machine (LLVM)
  - Performance is evaluated as Instructions-Per-Cycle (IPC)
  - Table 1: Experimental results for video application kernels

<table>
<thead>
<tr>
<th>Kernel functions of H.264</th>
<th>IPC for IDCT_ROW: 35.7 (gain 29%) vs. 27.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDR-CGRA vs. ADRES [1]:</td>
<td>IPC for IDCT_COLL: 35.9 (gain 9%) vs. 33.0</td>
</tr>
<tr>
<td>FDR-CGRA vs. RCP [2]:</td>
<td>For IDCT, 16-issue RCP achieved IPC of 9.2</td>
</tr>
<tr>
<td>Kernel functions of H.264</td>
<td>For IDCT, 1-tile (16-PE) FDR-CGRA can achieve IPC of 11.1 (gain 21%)</td>
</tr>
<tr>
<td>(a) Companion channels glue PEs together</td>
<td>(b) Structure of Processing Element</td>
</tr>
<tr>
<td>(c) FDR-CGRA system overview</td>
<td></td>
</tr>
</tbody>
</table>

- **Placement and routing for operations**

  - **Placement frontend:**
    - It only suggests a trial operation mapping and leaves the detailed resource conflict problems to be resolved later in the backend routing stage, virtually any scheduling algorithm can be used in the frontend stage. We find from experiment that even the simple and fast list scheduling can give a fairly good initial placement.

  - **Routing backend:**
    - Rip-up & reroute: Iteratively perform operations as follows: (1) Pick a routed path, usually a path that can afford extra latency without degrading overall performance; (2) rip-up (or break) the path by de-allocating all the resources (i.e., channels and ports) taken by this path; (3) based on the existing congestion information of the routing region, find an uncongested detour and reroute the path.

  - Non-volatile copy: distribute reusable operands across a tile as non-volatile copies so that a dependent PE can find a local copy in its vicinity.

  - Schedule step relaxation: If rip-up and reroute and non-volatile copy cannot resolve all the congestions, schedule step relaxation will be invoked to insert extra scheduling steps into the most congested regions.

- **Compilation for Fast Data Relay**

  - Benchmark programs are extracted from xvid 1.1.3 and JM7.5b
  - Preprocess with LLVM
  - Performance is evaluated as Instructions-Per-Cycle (IPC)
  - Table 1: Experimental results for video application kernels

- **Advantages of FDR-CGRA:**
  - A corner-to-corner transmission in a tile can be finished in two cycles;
  - Data communication can be done as a background operation without disturbing computation;
  - Source operands can have multiple copies in different PEs so that a dependent PE can find a local copy in its vicinity.

- **Higher Performance**
  - Benchmark programs are extracted from xvid 1.1.3 and JM7.5b
  - Preprocess with LLVM
  - Performance is evaluated as Instructions-Per-Cycle (IPC)
  - Table 1: Experimental results for video application kernels

<table>
<thead>
<tr>
<th>Benchmark programs</th>
<th>IPC for IDCT_ROW: 35.7 (gain 29%) vs. 27.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDR-CGRA vs. ADRES [1]:</td>
<td>IPC for IDCT_COLL: 35.9 (gain 9%) vs. 33.0</td>
</tr>
<tr>
<td>FDR-CGRA vs. RCP [2]:</td>
<td>For IDCT, 16-issue RCP achieved IPC of 9.2</td>
</tr>
<tr>
<td>Kernel functions of H.264</td>
<td>For IDCT, 1-tile (16-PE) FDR-CGRA can achieve IPC of 11.1 (gain 21%)</td>
</tr>
</tbody>
</table>

- **Placement and routing for operations**

  - **Placement frontend:**
    - It only suggests a trial operation mapping and leaves the detailed resource conflict problems to be resolved later in the backend routing stage, virtually any scheduling algorithm can be used in the frontend stage. We find from experiment that even the simple and fast list scheduling can give a fairly good initial placement.

  - **Routing backend:**
    - Rip-up & reroute: Iteratively perform operations as follows: (1) Pick a routed path, usually a path that can afford extra latency without degrading overall performance; (2) rip-up (or break) the path by de-allocating all the resources (i.e., channels and ports) taken by this path; (3) based on the existing congestion information of the routing region, find an uncongested detour and reroute the path.

    - Non-volatile copy: distribute reusable operands across a tile as non-volatile copies so that a dependent PE can find a local copy in its vicinity.

    - Schedule step relaxation: If rip-up and reroute and non-volatile copy cannot resolve all the congestions, schedule step relaxation will be invoked to insert extra scheduling steps into the most congested regions.