HardCloud: Automatic Offloading to Cluster Accelerators

Ramon Nepomuceno, Ciro Ceissler
and Prof. Dr. Guido Araújo

UNICAMP – University of Campinas
“... we have entered
the era of accelerators ...”

[EETimes 2017 - Google Searches for Better Silicon by Prasad Sabada]
Figure. Number of Specialized Blocks - Apple SoCs

[Die photo from Chipworks]
[Accelerators annotated by Sophia Shao @ Harvard]
The FPGAs are 40 times faster than a CPU at processing Bing’s custom al-

Data Center
Accelerators are good, but..

- How to integrate a software application to an accelerator?
- How to decompose an application to accelerators?
- How to rapidly design lots of accelerators?
- How to design and manage the shared resources?
Accelerators are good, but..

- How to integrate a software application to an accelerator?
- How to decompose an application to accelerators?
- How to rapidly design lots of accelerators?
- How to design and manage the shared resources?
Extending OpenMP
To Program an FPGA
What is OpenMP?
It is an Application Program Interface.

What is it for?
To make explicit the parallelism of your code.
Let's start our hands-on session?

- We will use a docker container
- Please execute:
  
  ```
  $ mkdir /home/$USER/lab6
  $ docker run --name hardcloud -v/home/$USER/lab6:/root/lab6 -it hardcloud/harp-simulation:2.0
  ```
Our First OpenMP program

```
#include<stdio.h>
#include "omp.h"
#define N 4

int main(){
    #pragma omp parallel for
    for(int i = 0; i < N; i++){
        int x = omp_get_thread_num();
        printf("Hi, I am thread %d\n", x);
    }
}
```

$ cd ~/lab6
$ source ~/setup.hardcloud
$ clang++ -fopenmp hello.cpp -o hello.exe
$ ./hello.exe
> Hi, I am thread 0
> ...
> Hi, I am thread 3
But OpenMP goes beyond parallel for

```c
/* some code */

#pragma omp target device(GPU) map(to: X) map(from: Y)
#pragma omp parallel for
for (int i = 0; i < elements_size; i++) {
    Y[i] = X[i];
}

/* some code */
```
#pragma omp target device(GPU) map(to: X) map(from: Y)
```c
#pragma omp target device(GPU)

map(to: X) map(from: Y)
```
```c
#pragma omp target device(GPU) map(to: X) map(from: Y)
```
What have we learned so far?

- Accelerators are good.
- But we need better ways to integrate software and accelerators.
- OpenMP target directive is a good option.
What is our goal?

- Is to use FPGA as an accelerator with OpenMP target directive.
FPGA
FPGAs

What is it?
● It is an array of logic gates that can be programmable.

What is it for?
● It is for prototyping and testing;
● And use as an accelerator;
Our First FPGA Circuit

```
module basic_and (
    input [3:0] A,
    input [3:0] B,
    output [3:0] C
);

assign C = A & B;
endmodule
```

`basic_and.sv`
module basic_and_tb();
  reg [3:0] a, b;
  wire [3:0] c;

  basic_and DUT (
    .A(a), .B(b), .C(c)
  );

  initial begin
    a = 4'b0101;
    b = 4'b1110;
    $display("input A=%b\n", a);
    $display("input B=%b\n", b);
    #20
    $display("output C=%b\n", c);
    $finish;
  end
endmodule
Execute:

```
$ cd ~/lab6
$ source ~/setup.hardcloud
$ export LM_LICENSE_FILE= 'port@server'
$ vlib work
$ vlog basic_and.sv basic_and_tb.sv
$ vsim -c basic_and_tb
$VSIM 1> run
# input     A=0101
# input     B=1110
# output  C=0100
```
What we have learned so far?

- Accelerators are good.
- But we need better ways to integrate software and accelerators.
- OpenMP target directive is a good option.
- What is and how to program an FPGA.
- We can use an FPGA as an accelerator.
HARDCLOUD
What is Hardcloud?
It is an extension of OpenMP plus a hardware API for Intel's HARP architecture.

What is it for?
To allow OpenMP to offload data and computation to an FPGA.
The Intel HARP architecture
/* some code */

#pragma omp target device(HARP) map(to: X) map(from: Y)
#pragma omp parallel for use(hrw) module(loopback)
for (int i = 0; i < elements_size; i++) {
    Y[i] = X[i];
}

/* some code */
#pragma omp target device(HARP) map(to: X) map(from: Y)
#pragma omp target device(HARP) map(to: X) map(from: Y)
```c
#include <stdio.h>
#include "omp.h"

int main()
{
    for(int i = 0; i < N; i++)
    {
        A[i] = i;
    }
    #pragma omp target device(HARPSIM) map(to: A) map(from: B)
    #pragma omp parallel for use(hrw) module(loopback)
    for(int i = 0; i < N; i++)
    {
        B[i] = A[i];
    }
    [ ... ]
}
```
HardCloud

$ source ~/setup.hardcloud
$ export LM_LICENSE_FILE="port@server"
$ cd ~/hardcloud/ase
$ make clean
$ make AFU=loopback
$ make sim

$ tmux
Ctrl+b+%  

wait for: Ready for simulation...

$ cd ~/hardcloud/samples/loopback/sw
$ source ~/setup.hardcloud
$ make
$ cd build
$ ./harpsim_loopback
$ [HardCloud] result: PASS
What just happened?
Intel HARP System
- **VTP (Virtual to Physical)**: Read and write from virtual addresses
- **ROB (Reorder Buffer)**: Return read responses in request order
● Manages communication with the host
• Controls the flow of data to/from the shared memory
/* some code */

#pragma omp target device(HARP) map(to: X) map(from: Y)
#pragma omp parallel for use(hrw) module(loopback)
for (int i = 0; i < elements_size; i++) {
   Y[i] = X[i];
}

/* some code */
/* some code */

#pragma omp target device(HARP) map(to: X) map(from: Y)
#pragma omp parallel for use(hrw) module(loopback)
for (int i = 0; i < elements_size; i++) {
    Y[i] = X[i];
}

/* some code */
Shared Memory

program the FPGA
Shared Memory

create DSM buffer

...
Shared Memory

reset blocks
/* some code */

#pragma omp target device(HARP) map(to: X) map(from: Y)
#pragma omp parallel for use(hrw) module(loopback)
for (int i = 0; i < elements_size; i++) {
    Y[i] = X[i];
}

/* some code */
Shared Memory

- DSM
- X
- Y
- ...

create map to/from buffers
/* some code */

#pragma omp target device(HARP) map(to: X) map(from: Y)
#pragma omp parallel for use(hrw) module(loopback)
for (int i = 0; i < elements_size; i++) {
    Y[i] = X[i];
}

/* some code */
execute
wait by polling
Shared Memory

- DSM
- X
- Y
- ...

finish
/* some code */

#pragma omp target device(HARP) map(to: X) map(from: Y)
#pragma omp parallel for use(hrw) module(loopback)
for (int i = 0; i < elements_size; i++) {
    Y[i] = X[i];
}

/* some code */
How to connect your core?
module IP
(
    input logic clk,
    input logic reset,
    input logic start,
    output logic finish,
    hcBuffers_if buffer
);

/* IP implementation */

endmodule : IP
buffer

[read request]
- read_stream(id, size)
- read_indexed(id, offset)

[read response]
- valid()
- data()

[write request]
- write_stream(id, data)
- write_indexed(id, offset, data)
logic [31:0] cnt_request;
t_rd_state rd_state;
t_rd_state rd_next_state;
t_ccip_c0_ReqMemHdr rd_hdr;
always_ff@(posedge clk or posedge reset) begin
  if (reset) begin
    cnt_request <= '0;
  end
  else begin
    logic [31:0] request;
    logic [31:0] response;
    if ((rd_state == S_RD_FETCH) && (cnt_request < FFT_FIFO_DEPTH) && !ccip_rx.c0TxAlmFull) begin
      request = 32'h1;
    end
    else begin
      request = 32'h0;
    end
    if ((ccip_rx.c0.rspValid) && (ccip_rx.c0.hdr.resp_type == eRSP_RDLINE)) begin
      response = 32'h1;
    end
    else begin
      response = 32'h0;
    end
    cnt_request <= cnt_request + request - response;
  end
end
always_ff@(posedge clk or posedge reset) begin
  if (reset) begin
    ccip_c0_tx.valid    <= 1'b0;
    rd_offset           <= '0;
    rd_hdr = t_ccip_c0_ReqMemHdr'(0);
  end
  else begin
    case (rd_state)
      S_RD_IDLE:
        begin
          ccip_c0_tx.valid <= 1'b0;
        end
      S_RD_FETCH:
        begin
          if (cnt_request >= FFT_FIFO_DEPTH) begin
            ccip_c0_tx.valid <= 1'b0;
          end
          else if (!ccip_rx.c0TxAlmFull) begin
            rd_hdr.cl_len  = eCL_LEN_1;
            rd_hdr.address = hc_buffer[1].address + rd_offset;
            ccip_c0_tx.valid    <= 1'b1;
            ccip_c0_tx.hdr      <= rd_hdr;
            rd_offset           <= t_ccip_clAddr'(rd_offset + 1);
          end
          else begin
            ccip_c0_tx.valid <= 1'b0;
          end
        end
      S_RD_WAIT:
        begin
          ccip_c0_tx.valid <= 1'b0;
        end
      S_RD_FINISH:
        begin
          ccip_c0_tx.valid <= 1'b0;
        end
    endcase
  end
end
always_ff@(posedge clk or posedge reset) begin
  if (reset) begin
    ~100 lines of code
  end
end

Before HardCloud

Using HardCloud

always_ff@(posedge clk or posedge rst) begin
  if (rst) begin
    buffer.read_idle();
  end
  else begin
    if (!buffer.read_fifo_is_full()) begin
      buffer.read_stream(0, 512);
    end
    else begin
      buffer.read_idle();
    end
  end
end

~100 lines of code
Diving into the loopback