Improvement of Cache System Automatic Design Tool for Heterogeneous Multi-core

Taiga YUKAWA and Takahiro SASAKI, Aichi Prefectural University, JAPAN

愛知県立大学 Aichi Prefectural University

Background

➢ **Requirements for multi-core processors**

- ✓ High-performance
- ✓ Low-energy

For use with ...

Machine Learning

IoT

Wearable Device

Positive Solution

Heterogeneous Multi-core Processors (HMP)

➢ **Auto-design tool for HMP**

FabHetero project

- It has problems at cache generation part (FabCache)
  - Design size explosion by changing the specifications
  - Low maintainability

It does not meet the FabHetero's concept

Motivation

We propose a **less design effort and higher maintainable** cache-generation module by improving the cache-generation module used in FabHetero.

Conclusion

We propose a general-purpose cache-generation module to improve the design efficiency of FabCache.
Our method seems to be effective in reducing design effort, and it can reduce about 12.6% of codes.

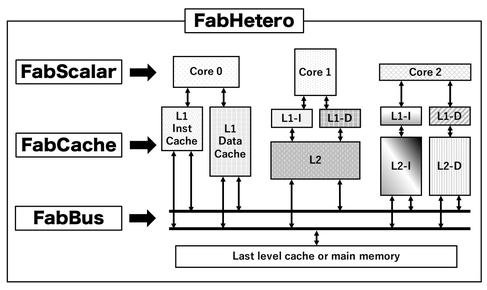
Previous Work | FabHetero

➢ **FabHetero's Goal**

Reducing design effort of HMP

➢ **Functions of FabHetero**

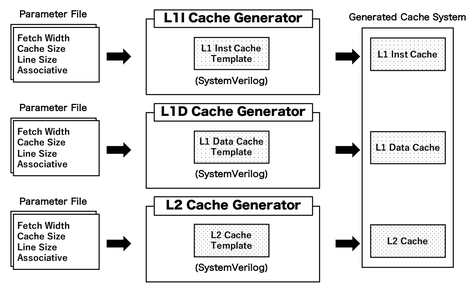
- Generating HMP design automatically with parameters.
- FabHetero consists of three generators.
  - FabScalar ... for generating Super Scalar
  - FabCache ... for generating Cache System
  - FabBus ... for generating Interconnection Networks



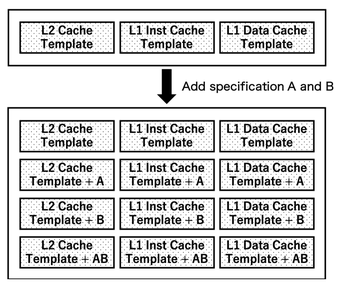
Overview of FabHetero

➢ **Problems of FabCache**

- It uses **three generators** to generate only one cache system.
- **Increasing design data** occurs when changing specifications.
- Some design data has the same structures and it is **redundant**.



FabCache



Increading design data

Proposed Solution

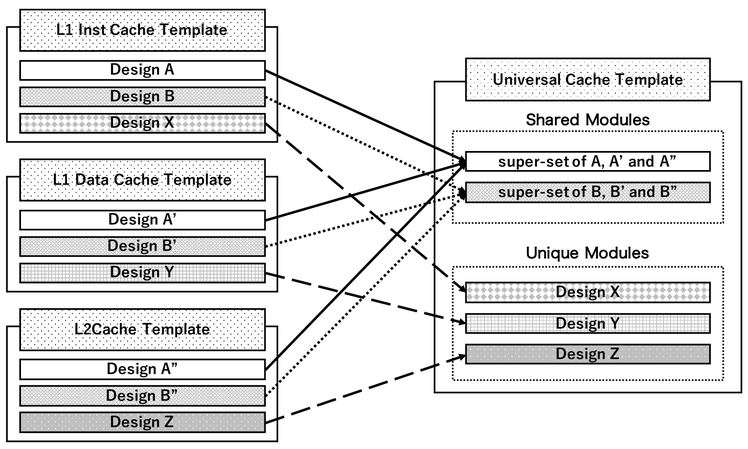
➢ **Strategy of the Proposed Method**

Create a **single cache design** that has the same functions as the FabCache

- ✓ Reduction of the amount of code
- ✓ Improvement of maintainability

➢ **Implementation of Proposed Module**

- The proposed design consists of **two types** of modules.
  - Shared-modules ... common part between some designs
  - Unique-modules ... selectable designs



Design strategy

Select the structure with parameters

```
ID_Shared_Module #(
    .USE_AS_D_CACHE(1)
) instance_used_for_d_cache(
    .clk(clk),
    .reset(reset),
    ...
);
```

```
module D_Module(
    input clk,
    input reset,
    ...
);

...

always_ff @(posedge clk)
begin
int i,j;
    if(reset)
    begin
        dataCtr <= 0;

        for(i = 0; i < L1DC_SIZE_LINE; i++)
        begin
            l2dataBuf[i] <= 0;
        end
    end
    else
    begin
        if (l2ready)
        begin
            dataCtr <= dataCtr + SIZE_BUS;

            for(j=0;j<SIZE_BUS;j++)
            begin
                l2dataBuf[dataCtr+j] <= l2dataBus.dataQ[j];
            end
        end

        else
        begin
            dataCtr <= 0;
        end
    end
end

...
```

```
module I_Module(
    input clk,
    input reset,
    ...
);

...

always_ff @(posedge clk)
begin
    if(reset)
    begin
        instCtr <= 0;
    end

    else
    begin
        if(l2InstBus.ready)
        begin
            instCtr <= instCtr+`SIZE_BUS;
        end

        else
        begin
            instCtr <= 0;
        end
    end
end

always_ff @(posedge clk)
begin
int i, j;
    if(!l2InstBus.ready)
    begin
        for(j = 0; j < `SIZE_BUS; j++)
        begin
            dataBuf[instCtr+j] <= l2InstBus.dataQ[j];
        end
    end
end

...
```

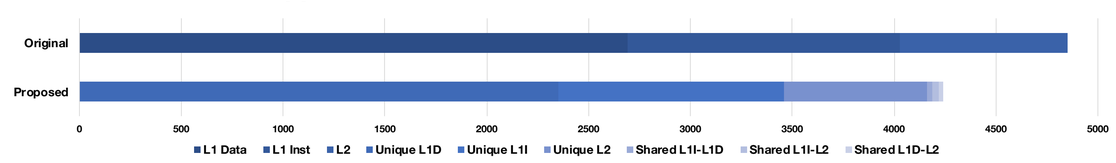
```
module ID_Shared_Module #(
    parameter USE_AS_D_CACHE = 0,
    parameter USE_AS_I_CACHE = 0,
    ...
)(
    input clk,
    input reset,
    ...
);

...

generate
    always_ff @(posedge clk)
    begin
        int i,j;
        if(reset)
        begin
            Ctr <= 0;

            if(USE_AS_D_CACHE)
            begin
                for(i = 0; i < L1DC_SIZE_LINE; i++)
                begin
                    dataBuf[i] <= 0;
                end
            end
        end
        else
        begin
            if(USE_AS_D_CACHE && l2ready || USE_AS_I_CACHE && l2Bus.ready)
            begin
                Ctr <= Ctr + `SIZE_BUS;
                for(j=0;j<SIZE_BUS;j++)
                begin
                    dataBuf[dataCtr+j] <= dataBus.dataQ[j];
                end
            end

            else
            begin
                Ctr <= 0;
            end
        end
    end
endgenerate

...
```

Example of Shared-module implementation

Effectiveness of improvement

The total HDL code amount of the FabCache is approximately 4850 lines without comments.
Out of those, about 610 lines (12.6%) can be shared between the designs such as L1-Inst, L1-Data or L2.

The approximate number of lines of code in each module



Future Works

We are going to implement the proposed method in SystemVerilog and synthesize the implemented module with the Design Compiler to evaluate the effectiveness and overhead of our proposed method.
We also plan to evaluate the proposed module from the viewpoint of the total HDL code amount and the hardware area to compare the proposed module with the original FabCache.