Logic Synthesis

Electronic Design Automation (Fall 2022)

Karthik B K <karthik.bk@incoresemi.com>

September 26 & 27

Outline

Introduction

What is Logic Synthesis

What is Logic Synthesis

- Behaviour -> Gate Level Implementation

What is Logic Synthesis

- Behaviour -> Gate Level Implementation
- Multiple Steps:

What is Logic Synthesis

- Behaviour -> Gate Level Implementation
- Multiple Steps:
    - Translation
    - Logic Optimization
    - ~~Technology Mapping~~

What is Logic Synthesis

- Behaviour -> Gate Level Implementation
- Multiple Steps:
    - Translation
    - Logic Optimization
    - ~~Technology Mapping~~
- High Level Synthesis
- Gate Level Synthesis

Some things to know

- What is ’synthesizable’ ?

Some things to know

- What is ’synthesizable’ ?
  - Anything that can be represented in boolean forms

Some things to know

- What is ’synthesizable’ ?
    - Anything that can be represented in boolean forms
- Objects to be synthesized

Some things to know

- What is ’synthesizable’ ?
    - Anything that can be represented in boolean forms
- Objects to be synthesized
    - Boolean functions
    - State Machines

Some things to know

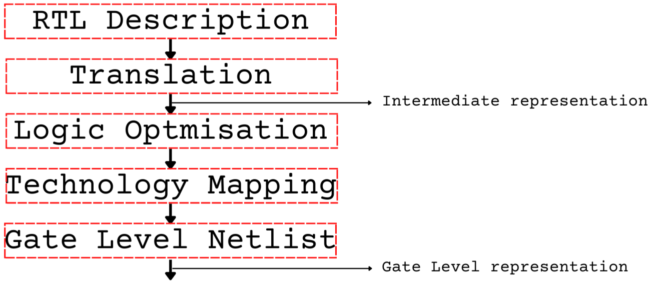
- What is ’synthesizable’ ?
    - Anything that can be represented in boolean forms
- Objects to be synthesized
    - Boolean functions
    - State Machines
- How do you know if it’s done correctly?

Some things to know

- What is ’synthesizable’ ?
    - Anything that can be represented in boolean forms
- Objects to be synthesized
    - Boolean functions
    - State Machines
- How do you know if it’s done correctly?
    - Gate level equivalence checking

Logic Synthesis Flow

Steps in Logic Synthesis



Logic Synthesis Flow

Exploring ABC

Installing ABC

```
$ git clone https://github.com/berkeley-abc/abc.git
$ cd abc; make -j$(nproc)
```

And-Invert Graphs

- Represented in terms of AND and INV only

And-Invert Graphs

- Represented in terms of AND and INV only
- Try it out! $a'b + ab + a(a' + b)$

And-Invert Graphs

- Represented in terms of AND and INV only
- Try it out! $a'b + ab + a(a' + b)$
- Oh wait this seems complex. let’s slow down!

Truth Table Optimisations - 2 level

- What is a truth table?
    - tells the truth of a function (in terms of inputs)

Truth Table Optimisations - 2 level

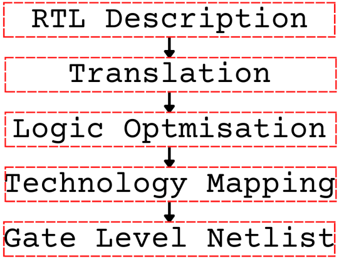
- What is a truth table?
  - tells the truth of a function (in terms of inputs)
  - write a simple one (and it’s kmap)

Truth Table Optimisations - 2 level

- What is a truth table?
    - tells the truth of a function (in terms of inputs)
    - write a simple one (and it's kmap)
    - let's understand this, and optimize it

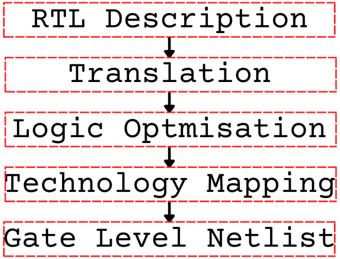
Description - Optimisation (Activity)

- Write an FSM



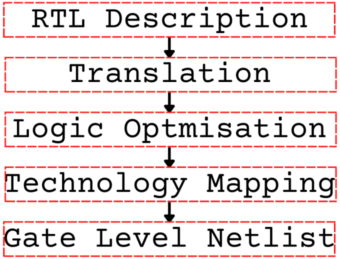
Description - Optimisation (Activity)

- Write an FSM
- Write it’s boolean form



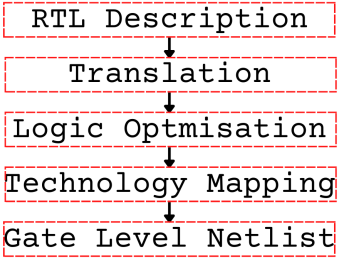
Description - Optimisation (Activity)

- Write an FSM
- Write it's boolean form
- Break it down



Description - Optimisation (Activity)

- Write an FSM
- Write it's boolean form
- Break it down
- ~~Convert it to NAND2~~



Same thing using ESPRESSO - A Heuristic Approach

Levels of minimisation

- Two level
  - AND plane
  - OR plane
  - Goal is to reduce num. of AND gates and their inputs.
- Multi level
  - let’s not get into this for now :)

Levels of minimisation

- Two level
    - AND plane

Levels of minimisation

- Two level
  - AND plane
  - OR plane

Levels of minimisation

- Two level
    - AND plane
    - OR plane
    - Goal is to reduce num. of AND gates and their inputs.

Levels of minimisation

- Two level
  - AND plane
  - OR plane
  - Goal is to reduce num. of AND gates and their inputs.
- Multi level
  - let’s not get into this for now :)

Two-Level Minimisation

- Steps:

Two-Level Minimisation

- Steps:
    - EXPAND - expand this cover

Two-Level Minimisation

- Steps:
    - EXPAND - expand this cover
    - IRREDUNDANT - keep only essential ones

Two-Level Minimisation

- Steps:
    - EXPAND - expand this cover
    - IRREDUNDANT - keep only essential ones
    - REDUCE - shrink this cover

Two-Level Minimisation

- Steps:
    - EXPAND - expand this cover
    - IRREDUNDANT - keep only essential ones
    - REDUCE - shrink this cover
- Until timeout or stable covers are obtained[1]

[1]Multiple times - Heuristic Approach

A detailed explaination using k-maps

- k-maps as cubes
- live example

An example from RISC-V instruction decoder

- show op_addr_mask from decode_box
- show abc script in misc/ from decode_box

Logic Synthesis using Yosys

Install and Setup Yosys

```
$ git clone https://github.com/YosysHQ/yosys.git
$ cd yosys; make -j$(nproc); sudo make install
```

Steps for Logic Synthesis (using Yosys)

```
yosys> read -sv design_name.v /* Read using verilog frontend */
```

Steps for Logic Synthesis (using Yosys)

```
yosys> read -sv design_name.v /* Read using verilog frontend */
yosys> hierarchy -top design_name /* set the top module */
```

Steps for Logic Synthesis (using Yosys)

```
yosys> read -sv design_name.v /* Read using verilog frontend */
yosys> hierarchy -top design_name /* set the top module */
yosys> proc /* convert always blocks to netlist elements */
```

Steps for Logic Synthesis (using Yosys)

```
yosys> read -sv design_name.v /* Read using verilog frontend */
yosys> hierarchy -top design_name /* set the top module */
yosys> proc /* convert always blocks to netlist elements */
yosys> techmap /* map to generic (internal) library */
```

Steps for Logic Synthesis (using Yosys)

```
yosys> read -sv design_name.v /* Read using verilog frontend */
yosys> hierarchy -top design_name /* set the top module */
yosys> proc /* convert always blocks to netlist elements */
yosys> techmap /* map to generic (internal) library */
yosys> write_verilog synth.v /* write the netlist */
```

Steps for Logic Synthesis (using Yosys)

```
yosys> read -sv design_name.v /* Read using verilog frontend */
yosys> hierarchy -top design_name /* set the top module */
yosys> proc /* convert always blocks to netlist elements */
yosys> techmap /* map to generic (internal) library */
yosys> write_verilog synth.v /* write the netlist */
yosys> dfflibmap -liberty my_library.lib /* map seq elements */
```

Steps for Logic Synthesis (using Yosys)

```
yosys> read -sv design_name.v /* Read using verilog frontend */
yosys> hierarchy -top design_name /* set the top module */
yosys> proc /* convert always blocks to netlist elements */
yosys> techmap /* map to generic (internal) library */
yosys> write_verilog synth.v /* write the netlist */
yosys> dfflibmap -liberty my_library.lib /* map seq elements */
yosys> abc -liberty my_library.lib /* map comb cells */
```

Writing sample design using bsv

synthesizing your own design (live demo) - and sending it through OpenLane

Conclusion

Conclusion

Use opensource tools (!!)

Open for Discussions

Thank You