Automatic clock gating on Yosys RTLIL level

The American University In Cairo

Dr. Mohamed Shalan

Youssef Ashraf Kandil

AUC
THE AMERICAN UNIVERSITY IN CAIRO

Abstract

The aim of this project is to build an automatic clock gating utility that is generic for all SKY_130 RTL designs. For the sake of generality and simplicity, this project deals with Yosys RTLIL (register-transfer-level intermediate language), and utilizes the already existing commands implemented in Yosys frontend synthesizer.

AUC
THE AMERICAN UNIVERSITY IN CAIRO

Identifying problem and solution

As mentioned in the Yosys documentation, the ASIC digital circuit design passes through seven abstraction layers:

- System Level
- High Level
- Behavioral Level
- Register-Transfer Level (RTL)
- Logical Gate Level
- Physical Gate Level
- Switch Level

Identifying problem and solution

The clock gating utility can be implemented on any of these abstraction levels. However, we specifically chose the Yosys RTLIL , which is a Logical Gate Level, since it presents logical format of the circuit instead of a behavioural one such as Verilog, as well as a high level of abstraction.

- Dealing with a logical format eliminates the burden of dealing with syntax variety of a behavioural hardware description language.
- Yosys RTLIL has a minimal amount of basic cells, thus reduces the complexity of the manipulation we need to perform.

AUC
THE AMERICAN UNIVERSITY IN CAIRO

Procedure

- Scripting
  - DFF Optimization
  - Memory Optimization
  - Technology Mapping
  - Synthesis
- Map File
  - DFFEs to DFFs
- Clock gate sizing

Scripting

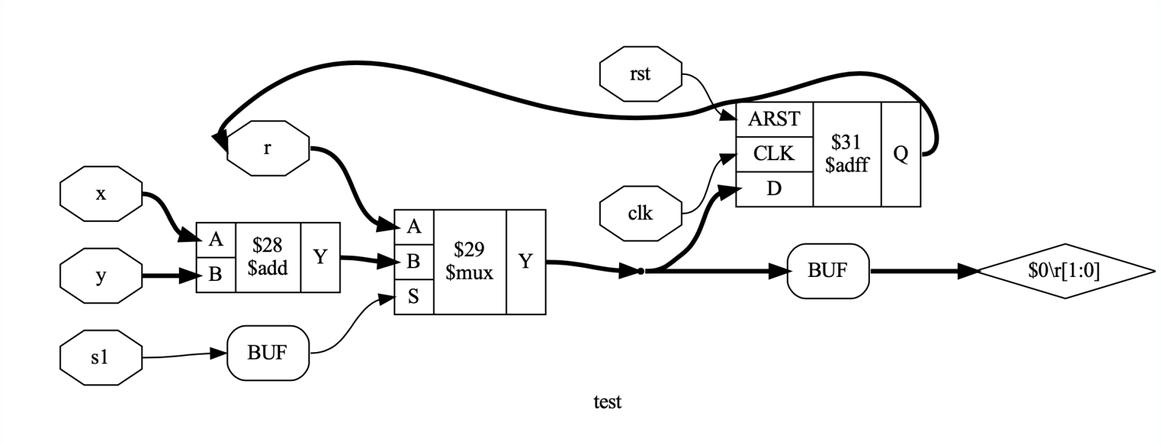
example script

```
read_verilog your_design.v
read_verilog blackbox_clk_gates.v
hierarchy -check -top your_design
show -prefix before_techmap -format pdf your_design
proc;
opt;;
memory_map
opt;;
techmap -map map_file.v;;
show -prefix  after_techmap_pass -format pdf your_design
synth -top your_design
dfflibmap -liberty sky130_hd.lib
abc -D 1250 -liberty sky130_hd.lib
splitnets
opt_clean -purge
write_verilog -noattr -noexpr -nohex -nodec -defparam
 outout_gatelevel.gl.v
write_blif  output.blif
write_edif  synth.edif
```

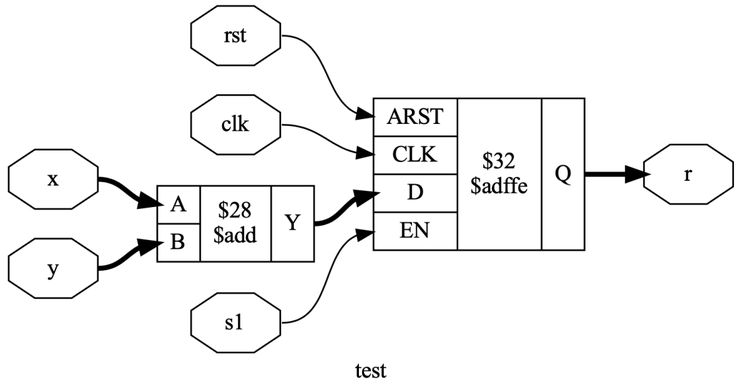
AUC
THE AMERICAN UNIVERSITY IN CAIRO

Scripting

- After reading the Verilog modules and the blackbox modules for the SKY_130 clock gating cells, the (proc;) command in then issued to model the Verilog RTL into RTLIL basic cells.
- Then followed by a necessary (opt;;) command for the sake of optimizing and cleaning the design, and more importantly, changing all ($MUX then $DFF) instances to combined ($DFFE) cells.
    - The opt command can be replaced by this command (opt_dff)



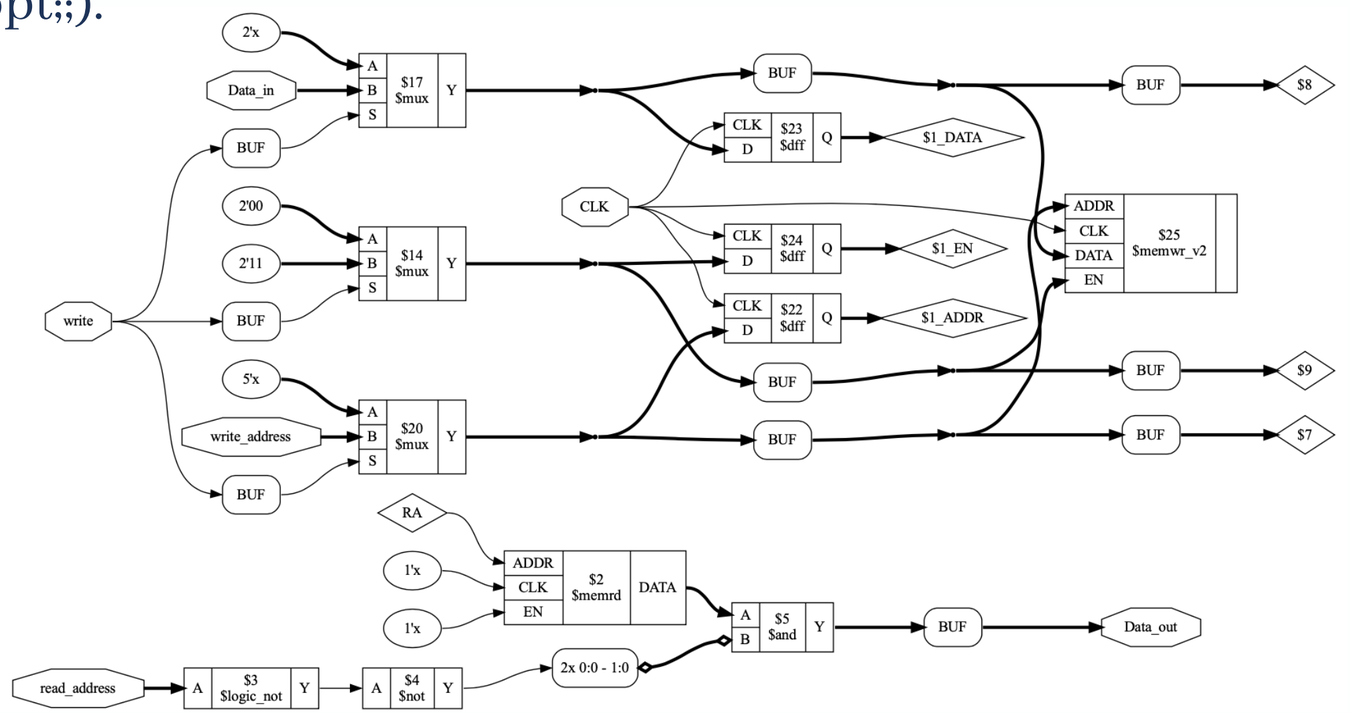
before opt



after opt

Scripting

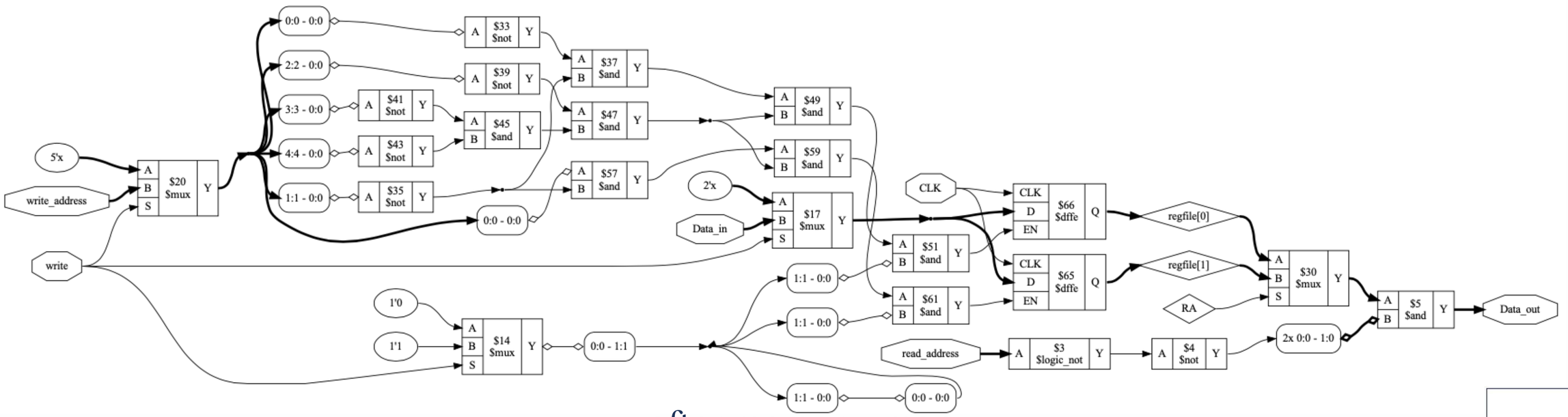
- Then the memory_map command is used to break down any memory structure into basic DFF cells, incase Yosys detects an array of registers, for example, a register file.
- This command is used so that we can insert a unique clock gate cell for each register, then followed by (opt;;).



before memory_map

Scripting

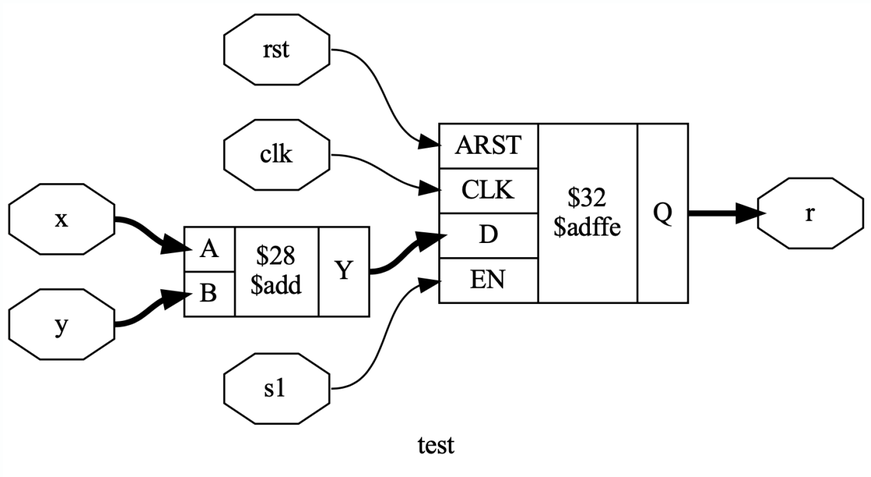
- Then the memory_map command is used to break down any memory structure into basic DFF cells, incase Yosys detects an array of registers, for example, a register file.
- This command is used so that we can insert a unique clock gate cell for each register, then followed by (opt;;).



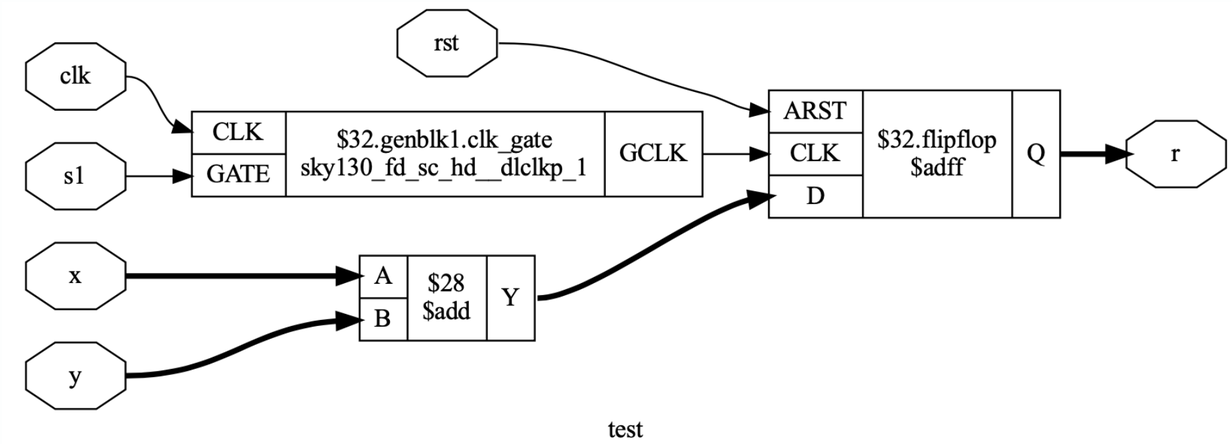
after memory_map

Scripting

- Now, we use the technology mapping command (techmap -map map_file.v;;) to replace any enabled D-flipflop with a clock-gated-flipflop.
- Here is an example of replacing only $ADFFE with $ADFF, however, in the map file section I will show how to make the technology mapping generic.



before techmap



after techmap

Scripting

- Finally, the design is synthesized and mapped to SKY_130 standard cell library using ABC backend synthesizer.

```
module test(x, y, clk, s1, rst, r);
 wire _0_;
 wire _1_;
 wire _2_;
 wire _3_;
 input clk;
 output r;
 input rst;
 input s1;
 input x;
 input y;
 sky130_fd_sc_hd__clkinv_1 _4_ (
 .A(rst),
 .Y(_0_)
 );
 sky130_fd_sc_hd__xnor2_1 _5_ (
 .A(x),
 .B(y),
 .Y(_3_)
 );
 sky130_fd_sc_hd__nor2_1 _6_ (
 .A(r),
 .B(s1),
 .Y(_2_)
 );
 sky130_fd_sc_hd__a21oi_1 _7_ (
 .A1(s1),
 .A2(_3_),
 .B1(_2_),
 .Y(_1_)
 );
 sky130_fd_sc_hd__dfrtp_1 _8_ (
 .CLK(clk),
 .D(_1_),
 .Q(r),
 .RESET_B(_0_)
 );
endmodule
```

before clock gating

```
module test(x, y, clk, s1, rst, r);
wire _0_;
wire _1_;
wire _2_;
input clk;
output r;
input rst;
input s1;
input x;
input y;
sky130_fd_sc_hd__clkinv_1 _3_ (
.A(rst),
.Y(_0_)
);
sky130_fd_sc_hd__xor2_1 _4_ (
.A(x),
.B(y),
.X(_1_)
);
sky130_fd_sc_hd__dfrtp_1 _5_ (
.CLK(_2_),
.D(_1_),
.Q(r),
.RESET_B(_0_)
);
sky130_fd_sc_hd__dlclkp_1 _6_ (
.CLK(clk),
.GATE(s1),
.GCLK(_2_)
);
endmodule
```

after clock gating

AUC
THE AMERICAN UNIVERSITY IN CAIRO

Map File

($DFFE) -->($DFF + clk_gate)
map_file example

```
////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
module \$dffe ( CLK, D, EN, Q);
    parameter CLK_POLARITY =1'b1;
    parameter EN_POLARITY =1'b1;
    parameter WIDTH =1;

    input  CLK, EN;
    input [WIDTH -1:0] D;
    output [WIDTH -1:0] Q;

    wire GCLK;

    generate
        if (WIDTH < 4) begin
            sky130_fd_sc_hd__dlclkp_1  clk_gate ( .GCLK(GCLK), .CLK(CLK), .GATE(EN) );
            end
        else if (WIDTH < 17) begin
            sky130_fd_sc_hd__dlclkp_2  clk_gate ( .GCLK(GCLK), .CLK(CLK), .GATE(EN) );
            end
        else begin
            sky130_fd_sc_hd__dlclkp_4  clk_gate ( .GCLK(GCLK), .CLK(CLK), .GATE(EN) );
        end
    endgenerate

    $dff  #(
            .WIDTH(WIDTH),
            .CLK_POLARITY(CLK_POLARITY),
            )
            flipflop(
            .CLK(GCLK),
            .D(D),
            .Q(Q)
            );
endmodule
////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
```

AUC
THE AMERICAN UNIVERSITY IN CAIRO

Map File

- As mentioned before, the aim of using the technology map command is to map all registers with enables to clock gated registers.
- The map file provides replacement cells to Yosys to be used during technology mapping.
- Since there are a few different types of D-flipflops in the RTLIL library, the map file needs to include a mapping module for each type for the utility to be generic.
- The (yosys/kernel/rtlil.h) file, in the Yosys GitHub repo, represents all the different DFF cells

Map File ($DFF cells)

- Here are the different types of D-flipflops in the RTLIL library:
  - $Dff / $Dffe
  - $Dffsr / $Dffsre
  - $Adff / $Adffe
  - $Aldff / $Aldffe
  - $Sdff / $Sdffe
  - $Sdffce

- The file is includes a mapping from each of the specified type with enable, to the same type without an enable, except for $Sdffce because it does not have a version without an enable.

Map File (Clock gate sizing)

- The sky_130 library has the following clock gate cells:
  - sky130_fd_sc_hd__dlclkp_1
  - sky130_fd_sc_hd__dlclkp_2
  - sky130_fd_sc_hd__dlclkp_4

- Since Yosys can not perform sizing on the clock gating cells as they are provided as blackbox cells, the following convention is followed for sizing
  - sky130_fd_sc_hd__dlclkp_1 for widths 0-3 bits
  - sky130_fd_sc_hd__dlclkp_2 for widths 4-16 bits
  - sky130_fd_sc_hd__dlclkp_4 otherwise

References

- https://yosyshq.net
- https://github.com/YosysHQ
- https://github.com/YosysHQ/yosys-manual-build/releases/download/manual/presentation.pdf
- https://github.com/YosysHQ/yosys-manual-build/releases/download/manual/manual.pdf

The American University In Cairo

Dr. Mohamed Shalan
Youssef Ashraf Kandil

Email Addresses

mshalan@aucegypt.edu
youssefkandil@aucegypt.edu

Presentation link

go to link