ASMD Design 1

16 June 2021 03:17 PM

Design Requirements

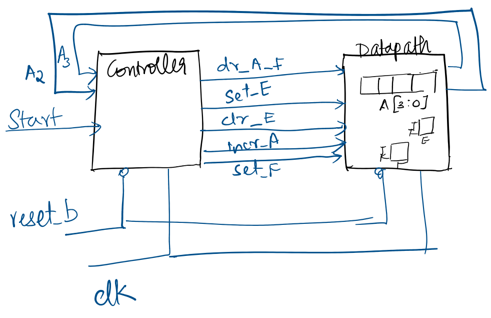
The datapath unit is to consist of two *JK* flip-flops *E* and *F*, and one four-bit binary counter *A[3:0]*. The individual flip-flops in *A* are denoted by A_3, A_2, A_1, and A_0, with A_3 holding the most significant bit of the count. A signal, *Start*, initiates the system's operation by clearing the counter *A* and flip-flop *F*. At each subsequent clock pulse, the counter is incremented by 1 until the operations stop. Counter bits A_2 and A_3 determine the sequence of operations:

If $A_2 = 0$, *E* is cleared to 0 and the count continues.

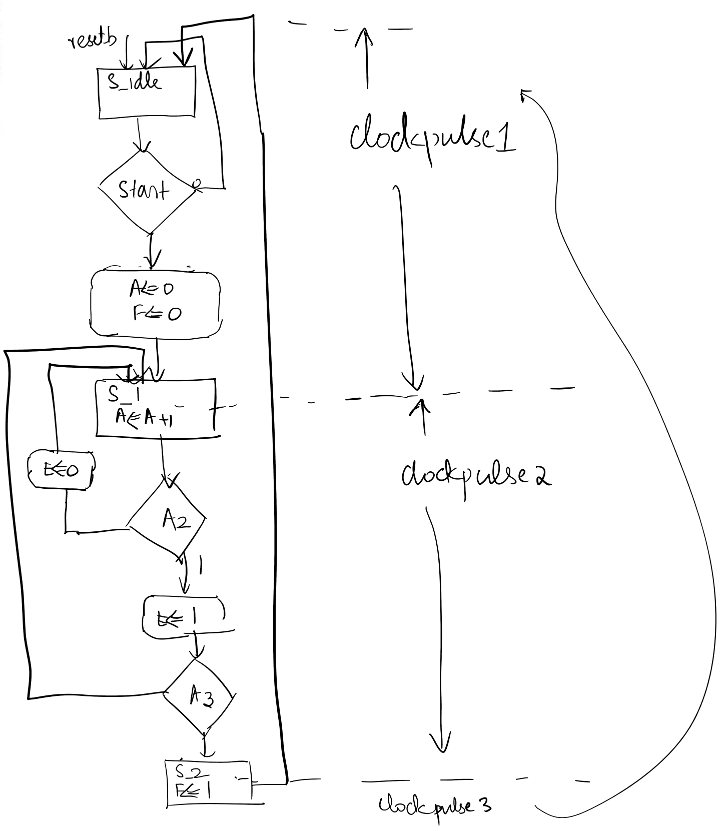
If $A_2 = 1$, *E* is set to 1; then, if $A_3 = 0$, the count continues, but if $A_3 = 1$, *F* is set to 1 on the next clock pulse and the system stops counting.

Then, if *Start* = 0, the system remains in the initial state, but if *Start* = 1, the operation cycle repeats.

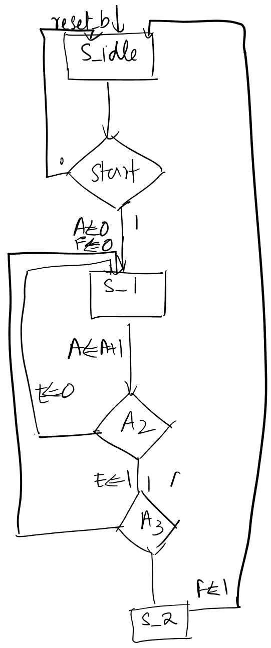
Design block diagram



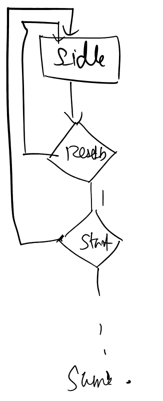
ASM chart



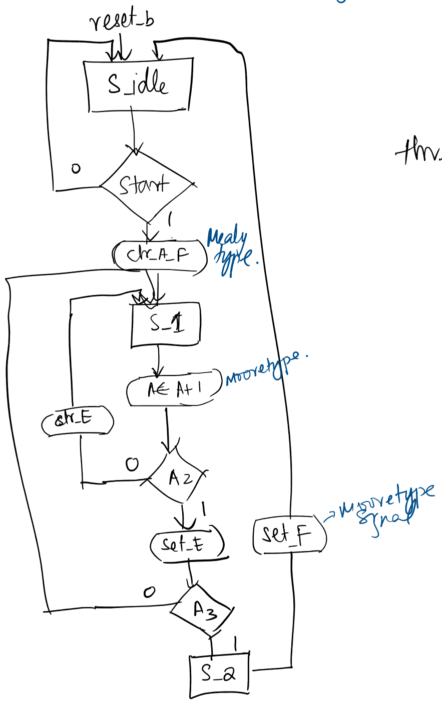
ASMD chart with asynchronous reset



ASMD chart with synchronous reset



Complete ASMD chart with control signals



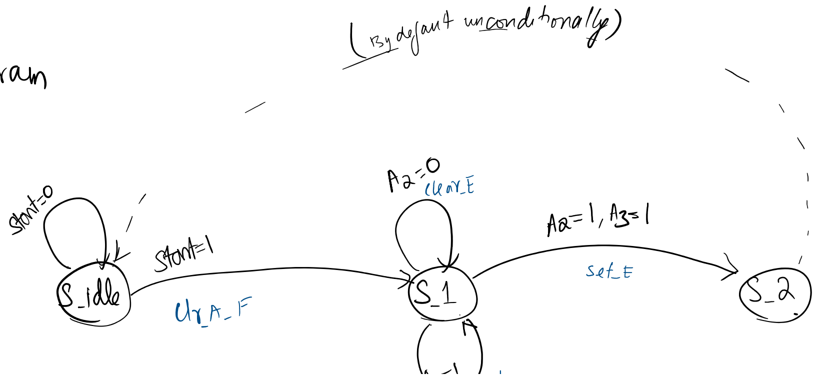
Voice note

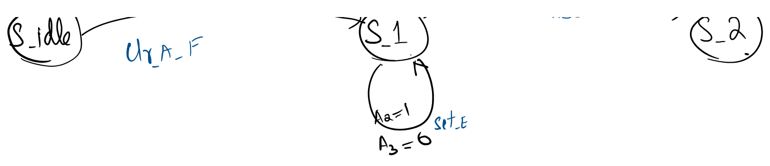
asmd1

three states

	G_1 G_0	
S_idle	00	Gray code assignment
S_1	01	
S_2.	11	

State diagram





$S\_idle \rightarrow S\_1$, clr_A_F $\qquad A \Leftarrow 0$, $F \Leftarrow 0$

$S\_1 \rightarrow S\_1$, incr_A ; $\qquad A \Leftarrow A+1$

if ($A_2=1$) then set_E ; $\qquad E \Leftarrow 0$

if ($A_2=0$) then clear_E $\qquad E \Leftarrow 1$

$S\_1 \rightarrow S\_2$, incr_A

Set_E

S_2 → ~~S~~ idle ; set_F $\qquad F \Leftarrow 1$

Assuming the counter starts from the reset state (All 0 in counter & FF)

	Counter				Flip Flops		Condition	State
	A_3	A_2	A_1	A_0	E	F		
initially	0	0	0	0	0	0	$A_2=0$ $A_3=0$	S_1
cp1	0	0	0	1	0	0		
cp2	0	0	1	0	0	0		
cp3	0	0	1	1	0	0	$A_2=0$ $A_3=0$, so clear E and go to ↳ non-blocking f	
~~cp4~~ A_2 becomes 1 still here E is not set	0	1	0	0	0	0	$A_2=1$ $A_3=0$	S_1
as before that the clock pulse saw A_2 to be 0 so E is 0, only in next cp the change is seen.	0	1	0	1	1	0		
	0	1	1	0	1	0		
	0	1	1	1	1	0		
A_2 becomes 0 still here E is not reset	1	0	0	0	1	0	$A_2=0$ $A_3=1$	
	1	0	0	1	0	0		
	1	0	1	0	0	0		
	1	0	1	1	0	0		
$A_2=1$ $A_3=1$ still E is not set	1	1	0	0	0	0	$A_2=1$ $A_3=1$	
E is set F is not set	1	1	0	1	1	0	$A_2=1$ $A_3=1$	S_2
E and F is set	1	1	0	1	1	1		S_idle.

~~[illegible]~~o 1. When $A = (A_3\ A_2\ A_1\ A_0)$ 0011, the next (4th) clock pulse increments the counter to 0100, but that same clock edge sees the value of A_2 as 0, so E remains cleared. The next (5th) pulse changes the counter from 0100 to 0101, and because A_2 is equal to 1 *before* the clock pulse arrives, E is set to 1. Similarly, E is cleared to 0 not when the count goes from 0111 to 1000, but when it goes from 1000 to 1001, which is when A_2 is 0 in the *present* value of the counter.

When the count reaches 1100, both A_2 and A_3 are equal to 1. The next clock edge increments A by 1, sets E to 1, and transfers control to state *S_2*. Control stays in *S_2* for only one clock period. The clock edge associated with the path leaving *S_2* sets flip-flop F to 1 and transfers control to state *S_idle*. The system stays in the initial state *S_idle* as long as *Start* is equal to 0.

From an observation of Table 8.3, it may seem that the operations performed on E are delayed by one clock pulse. This is the difference between an ASMD chart and a conventional flowchart. If Fig. 8.9(d) were a conventional flowchart, we would assume that A is first incremented and the incremented value would have been used to check the status of A_2. The operations that are performed in the digital hardware, as specified by a block in the ASMD chart, occur during the same clock cycle and not in a sequence of operations following each other in time, as is the usual interpretation in a conventional flowchart. Thus, the value of A_2 to be considered in the decision box is taken

} → operations occur Concurrently in ASMD (sequentially in conventional flow chart)

from the value of the counter in the present state and before it is incremented. This is because the decision box for E belongs with the same block as state *S_1*. The digital circuits in the control unit generate the signals for all the operations specified in the present block *prior to the arrival of the next clock pulse*. The next clock edge executes all the operations in the registers and flip-flops, including the flip-flops in the controller that determine the next state, using the present values of the output signals of the controller. Thus, the signals that control the operations in the datapath unit are formed in the controller in the clock cycle (control state) *preceding* the clock edge at which the operations execute.

present block *prior to the arrival of the next clock pulse*. The next clock edge executes all the operations in the registers and flip-flops, including the flip-flops in the controller that determine the next state, using the present values of the output signals of the controller. Thus, the signals that control the operations in the datapath unit are formed in the controller in the clock cycle (control state) *preceding* the clock edge at which the operations execute.

No. of states = $\boxed{3}$ → we need atleast 2 bits $G_1 G_0$

S_idle	00
S_1	01
S_2	11

(1 bit change so the combinational logic will be simplified)

STATE TABLE

State symbol	Present state Q_t		Input			Next state Q_{t+1}		output				
	G_1	G_0	Start	A_2	A_3	$G_{1\,t+1}$	$G_{0\,t+1}$	set_E	clr_E	set_F	clr_A_F	incr_A
S_idle.	0	0	0	X	X	0	0	0	0	0	0	0
S_idle.	0	0	1	X	X	0	1	0	0	0	1	0
S_1	0	1	X	0	X	0	1	0	1	0	0	1
S_1	0	1	X	1	0	0	1	1	0	0	0	1
S_1	0	1	X	1	1	1	1	1	0	0	0	1
S_2	1	1	X	X	X	0	0	0	0	1	0	0

(We realise using D-ff so no separate excitation table is required) $Q_{t+1} = D$

We need the expression for the next state & outputs as f(present state, I/P)

There are 5 variables. So we go by inspection instead of k-map.

$G_{0\,t+1} = D_{G_0} = \text{Start}.\text{S\_idle} + G_0$

$D_{G_0} = \text{Start}.\cancel{G_1'}G_0' + G_0$

[S_idle can be represented only as G_0' as $G_0 = 0$ in S_idle Other all states have $G_0 = 1$]

$G_{1\,t+1} = D_{G_1} = A_2 A_3 \text{ S\_1}$

$D_{G_1} = A_2 A_3 G_1' G_0$

$\text{Set\_E} = A_2 \text{ S\_1} = A_2 G_1' G_0$

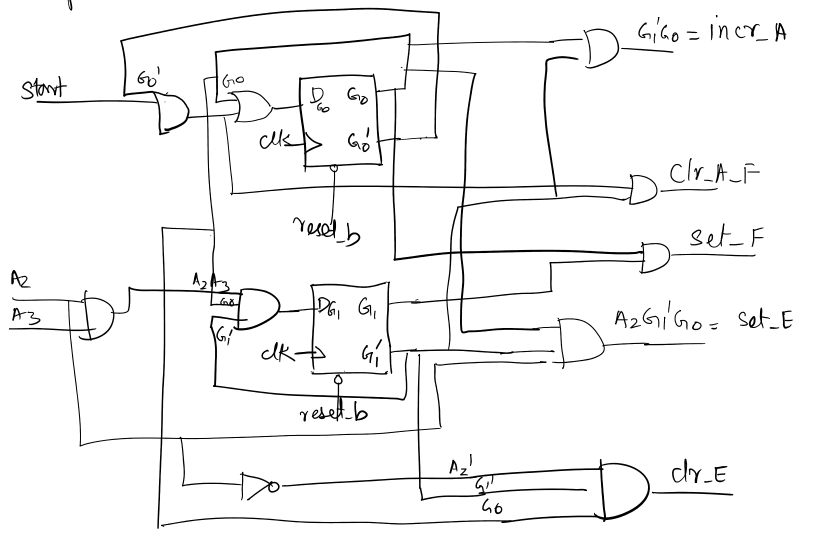
$\text{clr\_E} = A_2' \text{ S\_1} = A_2' G_1' G_0$

$\text{set\_F} = \text{S\_2} = G_1 G_0$

$\text{clr\_A\_F} = \text{S\_idle}.\text{start} = G_0' \cancel{G_1'}.\text{Start}$

$\text{incr\_A} = \text{S\_1} = G_1' G_0$

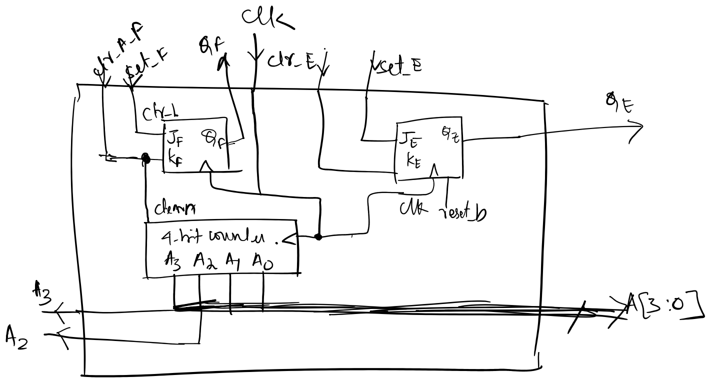
Controlpath design – manual



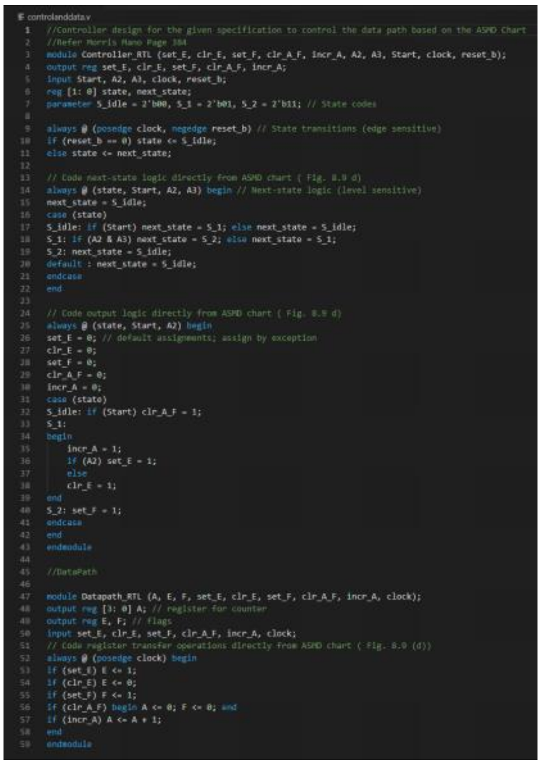
(The gates can be optimised further and 2-input AND gates can be used)

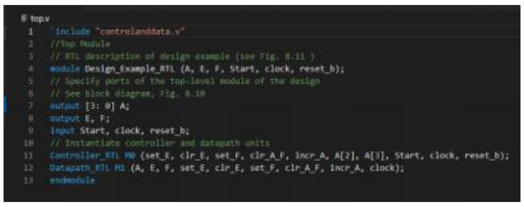
(The gates can be optimised further and 2-input AND gates can be used)
(The optimisation is usually automatically done by the synthesizer

Datapath design

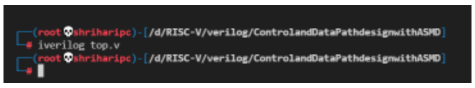


Behaviour level Verilog design

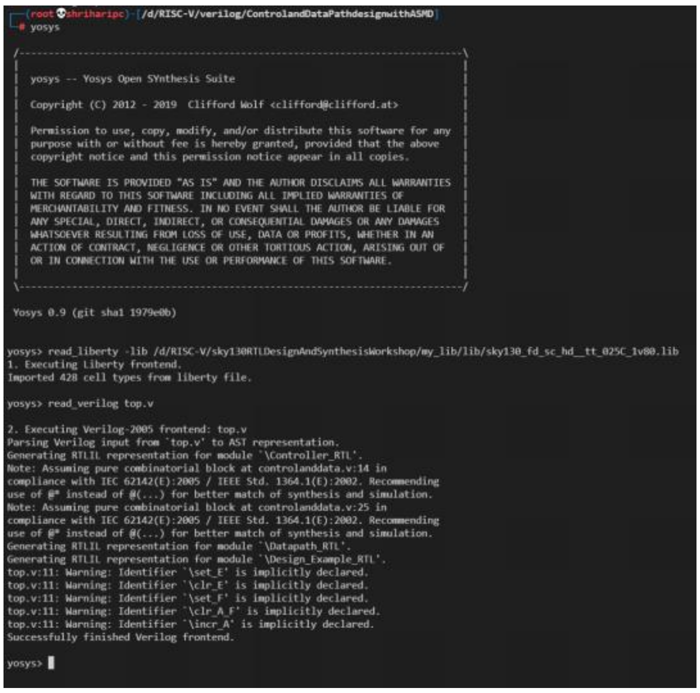

controlanddata.v
```
//Controller design for the given specification to control the data path based on the ASMD Chart
//Refer Morris Mano Page 384
module Controller_RTL (set_E, clr_E, set_F, clr_A_F, incr_A, A2, A3, Start, clock, reset_b);
output reg set_E, clr_E, set_F, clr_A_F, incr_A;
input Start, A2, A3, clock, reset_b;
reg [1: 0] state, next_state;
parameter S_idle = 2'b00, S_1 = 2'b01, S_2 = 2'b11; // State codes

always @ (posedge clock, negedge reset_b) // State transitions (edge sensitive)
if (reset_b == 0) state <= S_idle;
else state <= next_state;

// Code next-state logic directly from ASMD chart ( Fig. 8.9 d)
always @ (state, Start, A2, A3) begin // Next-state logic (level sensitive)
next_state = S_idle;
case (state)
S_idle: if (Start) next_state = S_1; else next_state = S_idle;
S_1: if (A2 & A3) next_state = S_2; else next_state = S_1;
S_2: next_state = S_idle;
default : next_state = S_idle;
endcase
end

// Code output logic directly from ASMD chart ( Fig. 8.9 d)
always @ (state, Start, A2) begin
set_E = 0; // default assignments; assign by exception
clr_E = 0;
set_F = 0;
clr_A_F = 0;
incr_A = 0;
case (state)
S_idle: if (Start) clr_A_F = 1;
S_1:
begin
    incr_A = 1;
    if (A2) set_E = 1;
    else
    clr_E = 1;
end
S_2: set_F = 1;
endcase
end
endmodule

//DataPath

module Datapath_RTL (A, E, F, set_E, clr_E, set_F, clr_A_F, incr_A, clock);
output reg [3: 0] A; // register for counter
output reg E, F; // flags
input set_E, clr_E, set_F, clr_A_F, incr_A, clock;
// Code register transfer operations directly from ASMD chart ( Fig. 8.9 (d))
always @ (posedge clock) begin
if (set_E) E <= 1;
if (clr_E) E <= 0;
if (set_F) F <= 1;
if (clr_A_F) begin A <= 0; F <= 0; end
if (incr_A) A <= A + 1;
end
endmodule
```


top.v
```
`include "controlanddata.v"
//Top Module
// RTL description of design example (see Fig. 8.11 )
module Design_Example_RTL (A, E, F, Start, clock, reset_b);
// Specify ports of the top-level module of the design
// See block diagram, Fig. 8.10
output [3: 0] A;
output E, F;
input Start, clock, reset_b;
// Instantiate controller and datapath units
Controller_RTL M0 (set_E, clr_E, set_F, clr_A_F, incr_A, A[2], A[3], Start, clock, reset_b);
Datapath_RTL M1 (A, E, F, set_E, clr_E, set_F, clr_A_F, incr_A, clock);
endmodule
```

Compiling the verilog code


```
┌──(root💀shriharipc)-[/d/RISC-V/verilog/ControlandDataPathdesignwithASMD]
└─# iverilog top.v
┌──(root💀shriharipc)-[/d/RISC-V/verilog/ControlandDataPathdesignwithASMD]
└─# █
```

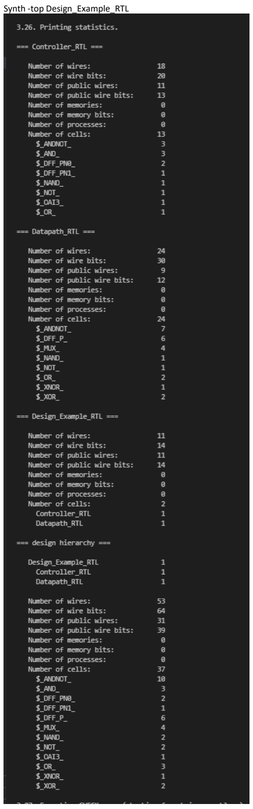
Invoking yosys



```
┌──(root💀shriharipc)-[/d/RISC-V/verilog/ControlandDataPathdesignwithASMD]
└─# yosys

 /----------------------------------------------------------------------------\
 |                                                                            |
 |  yosys -- Yosys Open SYnthesis Suite                                       |
 |                                                                            |
 |  Copyright (C) 2012 - 2019  Clifford Wolf <clifford@clifford.at>           |
 |                                                                            |
 |  Permission to use, copy, modify, and/or distribute this software for any  |
 |  purpose with or without fee is hereby granted, provided that the above    |
 |  copyright notice and this permission notice appear in all copies.         |
 |                                                                            |
 |  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES  |
 |  WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF          |
 |  MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR   |
 |  ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES    |
 |  WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN     |
 |  ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF   |
 |  OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.            |
 |                                                                            |
 \----------------------------------------------------------------------------/

 Yosys 0.9 (git sha1 1979e0b)


yosys> read_liberty -lib /d/RISC-V/sky130RTLDesignAndSynthesisWorkshop/my_lib/lib/sky130_fd_sc_hd__tt_025C_1v80.lib
1. Executing Liberty frontend.
Imported 428 cell types from liberty file.

yosys> read_verilog top.v

2. Executing Verilog-2005 frontend: top.v
Parsing Verilog input from `top.v' to AST representation.
Generating RTLIL representation for module `\Controller_RTL'.
Note: Assuming pure combinatorial block at controlanddata.v:14 in
compliance with IEC 62142(E):2005 / IEEE Std. 1364.1(E):2002. Recommending
use of @* instead of @(...) for better match of synthesis and simulation.
Note: Assuming pure combinatorial block at controlanddata.v:25 in
compliance with IEC 62142(E):2005 / IEEE Std. 1364.1(E):2002. Recommending
use of @* instead of @(...) for better match of synthesis and simulation.
Generating RTLIL representation for module `\Datapath_RTL'.
Generating RTLIL representation for module `\Design_Example_RTL'.
top.v:11: Warning: Identifier `\set_E' is implicitly declared.
top.v:11: Warning: Identifier `\clr_E' is implicitly declared.
top.v:11: Warning: Identifier `\set_F' is implicitly declared.
top.v:11: Warning: Identifier `\clr_A_F' is implicitly declared.
top.v:11: Warning: Identifier `\incr_A' is implicitly declared.
Successfully finished Verilog frontend.

yosys> █
```

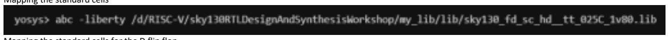
Synthesis

Synth -top Design_Example_RTL



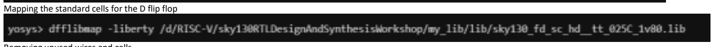
```
3.26. Printing statistics.

=== Controller_RTL ===

   Number of wires:                 18
   Number of wire bits:             20
   Number of public wires:          11
   Number of public wire bits:      13
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 13
     $_ANDNOT_                       3
     $_AND_                          3
     $_DFF_PN0_                      2
     $_DFF_PN1_                      1
     $_NAND_                         1
     $_NOT_                          1
     $_OAI3_                         1
     $_OR_                           1

=== Datapath_RTL ===

   Number of wires:                 24
   Number of wire bits:             30
   Number of public wires:           9
   Number of public wire bits:      12
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 24
     $_ANDNOT_                       7
     $_DFF_P_                        6
     $_MUX_                          4
     $_NAND_                         1
     $_NOT_                          1
     $_OR_                           2
     $_XNOR_                         1
     $_XOR_                          2

=== Design_Example_RTL ===

   Number of wires:                 11
   Number of wire bits:             14
   Number of public wires:          11
   Number of public wire bits:      14
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  2
     Controller_RTL                  1
     Datapath_RTL                    1

=== design hierarchy ===

   Design_Example_RTL                1
     Controller_RTL                  1
     Datapath_RTL                    1

   Number of wires:                 53
   Number of wire bits:             64
   Number of public wires:          31
   Number of public wire bits:      39
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 37
     $_ANDNOT_                      10
     $_AND_                          3
     $_DFF_PN0_                      2
     $_DFF_PN1_                      1
     $_DFF_P_                        6
     $_MUX_                          4
     $_NAND_                         2
     $_NOT_                          2
     $_OAI3_                         1
     $_OR_                           3
     $_XNOR_                         1
     $_XOR_                          2
```

Mapping the standard cells



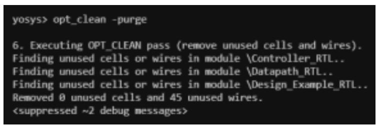
```
yosys> abc -liberty /d/RISC-V/sky130RTLDesignAndSynthesisWorkshop/my_lib/lib/sky130_fd_sc_hd__tt_025C_1v80.lib
```

Mapping the standard cells for the D flip flop



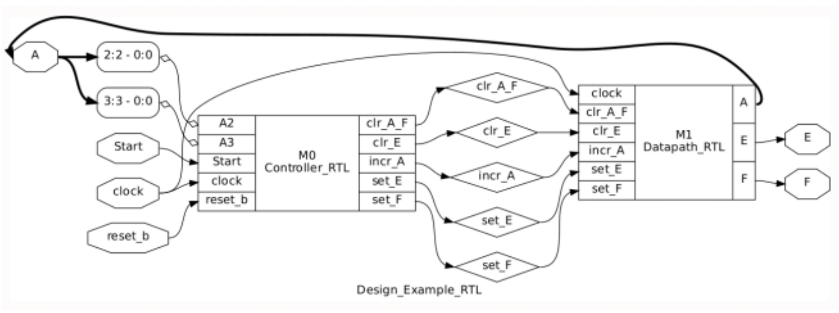
```
yosys> dfflibmap -liberty /d/RISC-V/sky130RTLDesignAndSynthesisWorkshop/my_lib/lib/sky130_fd_sc_hd__tt_025C_1v80.lib
```

Removing unused wires and cells

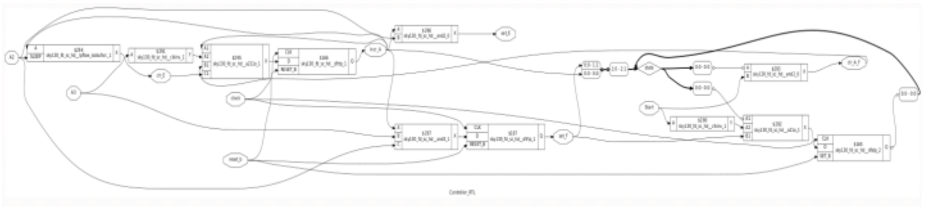


```
yosys> opt_clean -purge

6. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \Controller_RTL..
Finding unused cells or wires in module \Datapath_RTL..
Finding unused cells or wires in module \Design_Example_RTL..
Removed 0 unused cells and 45 unused wires.
<suppressed ~2 debug messages>
```

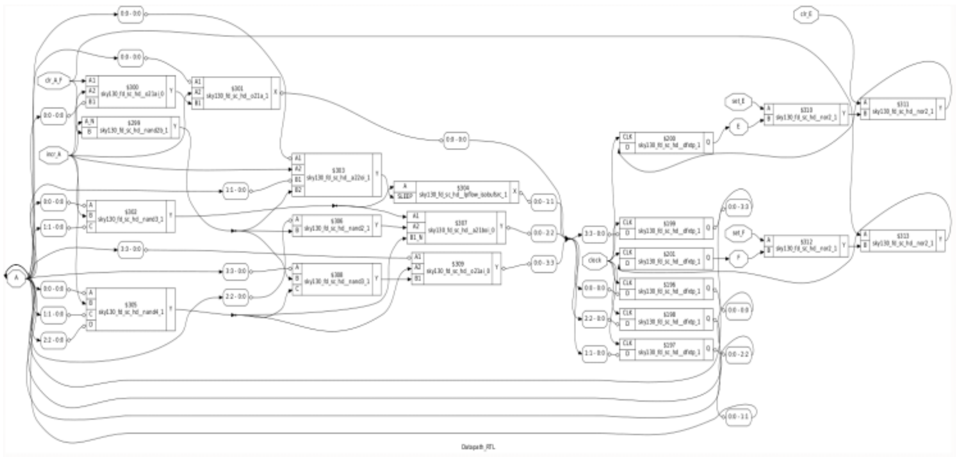
Yosys> show Design_Example_RTL



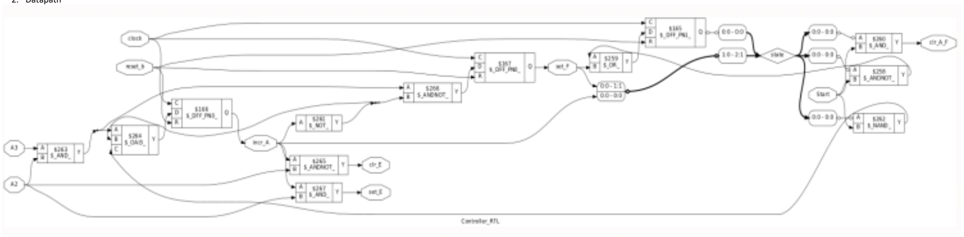
Yosys>show Controller_RTL

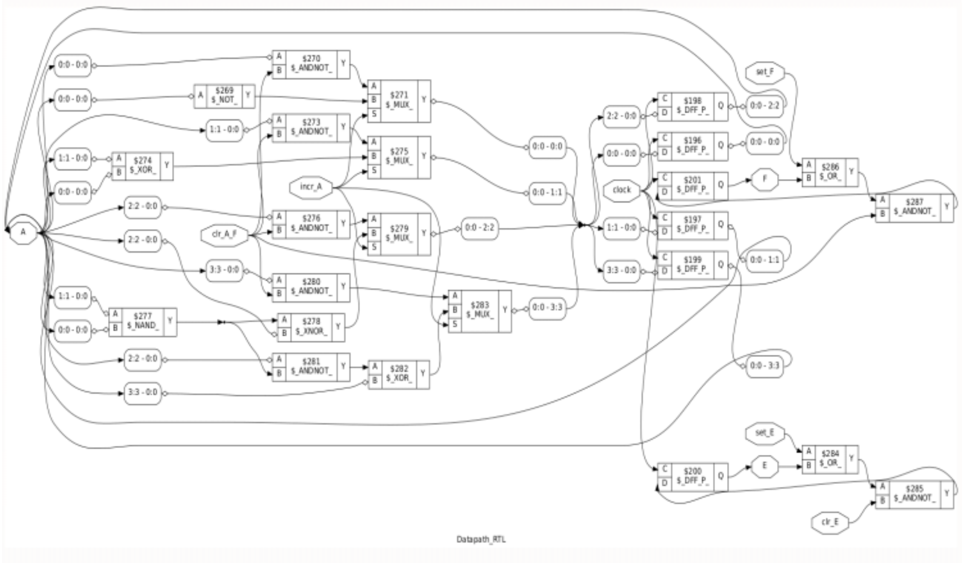


Yosys>show Datapath_RTL



---Note ---
---Controller and datapath before mapping the standard cells----
1. Controller
2. Datapath


clock
reset_b
A3
A2
Start
$165 $_DFF_PN1_
$167 $_DFF_PN0_
$166 $_DFF_PN0_
$263 $_AND_
$264 $_OAI3_
$266 $_ANDNOT_
$261 $_NOT_
$265 $_ANDNOT_
$267 $_AND_
$259 $_OR_
$260 $_AND_
$258 $_ANDNOT_
$262 $_NAND_
state
incr_A
clr_E
set_E
set_F
clr_A_F
Controller_RTL


A
incr_A
clr_A_F
clock
set_F
set_E
clr_E
E
F
$270 $_ANDNOT_
$269 $_NOT_
$271 $_MUX_
$273 $_ANDNOT_
$274 $_XOR_
$275 $_MUX_
$276 $_ANDNOT_
$279 $_MUX_
$280 $_ANDNOT_
$277 $_NAND_
$278 $_XNOR_
$283 $_MUX_
$281 $_ANDNOT_
$282 $_XOR_
$198 $_DFF_P_
$196 $_DFF_P_
$201 $_DFF_P_
$197 $_DFF_P_
$199 $_DFF_P_
$200 $_DFF_P_
$286 $_OR_
$287 $_ANDNOT_
$284 $_OR_
$285 $_ANDNOT_
Datapath_RTL