Synthesizing Full Adder Design to Gates

Short Explanation: What Synthesis Is

Synthesis is the process of converting a Verilog RTL description into a gate-level implementation using predefined logic elements from a device library. In simple terms, it takes the high-level design (like equations written with +, ^, &) and maps them to actual hardware building blocks. For ASICs, these are standard cells such as AND, OR, XOR gates from a cell library. In FPGA tools like Vivado, the logic is mapped to LUTs (Look-Up Tables), flip-flops, and input/output buffers available in the target device. This step transforms the abstract code into a realizable circuit that can later be implemented on hardware.

Short Explanation: What a Netlist Is

A netlist is the gate-level representation of a design produced after synthesis. It describes the circuit using the actual hardware primitives available in the target technology. In ASIC flows, a netlist is built from standard cells such as AND, OR, XOR, and flip-flops from a cell library. In FPGA flows like Vivado, the RTL is instead mapped into device primitives such as LUTs (Look-Up Tables) for logic, IBUFs for inputs, and OBUFs for outputs. The netlist connects these elements together to implement the same functionality as the original Verilog code, but now in a form that can be realized on the physical hardware.

Synth_script.tcl

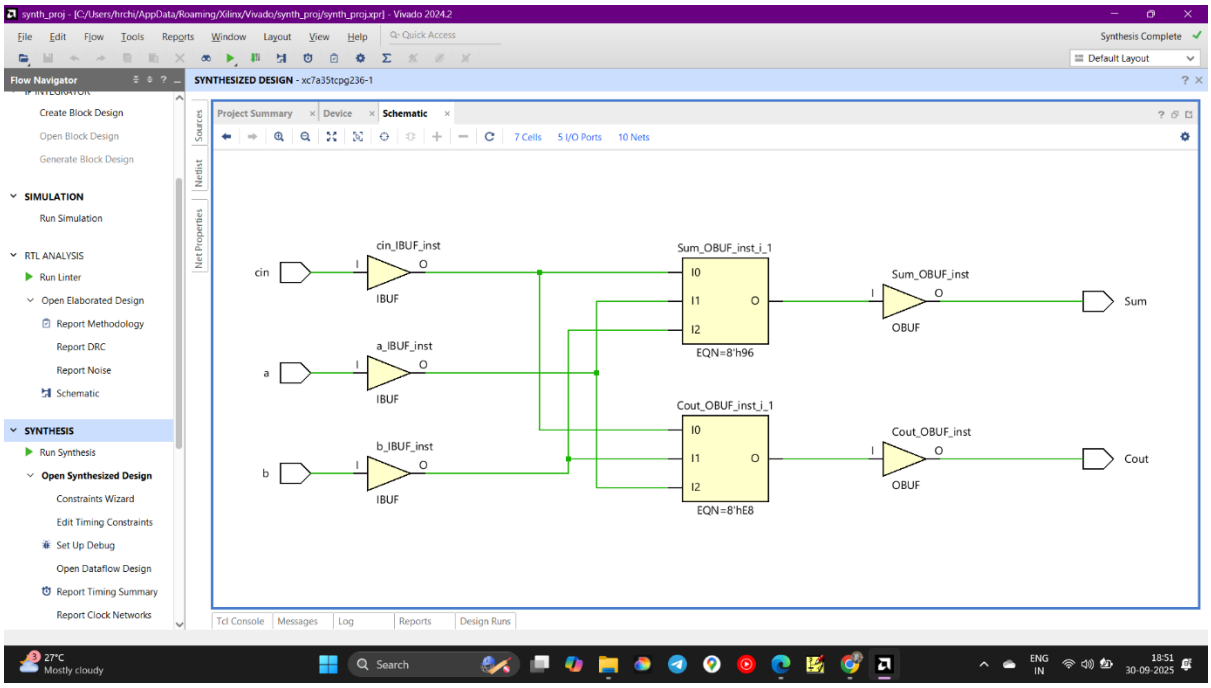
```
# 1. Creating a new project
create_project synth_proj ./synth_proj -part xc7a35tcpg236-1 -force

# 2. Adding Verilog source file
add_files C:/Users/hrchi/VIVADO_PROJECTS/Soft_Nexis_Projects/Full_adder.v

# 3. Settting the top module name
set_property top Full_adder [current_fileset]

# 4. Running Synthesis
launch_runs synth_1
wait_on_run synth_1

# 5. for Opening synthesized design
open_run synth_1

# 6. Writing the synthesized netlist (Gate-level Verilog)
write_verilog -force ./synthesized_netlist.v

# 7. for Writing reports
report_utilization -file ./utilization_report.txt
report_timing_summary -file ./timing_report.txt
```

Synthesizing Full Adder Design to Gates

Note: Full_adder.v file extracted from previous executed project.

```
`timescale 1ns / 1ps
module Full_adder(
    input a,b,cin,
    output Sum,Cout
    );
   assign Sum=a^b^cin;
   assign Cout= (a&b)|((a^b)&cin);
endmodule
```

Schematic Diagram



Netlist diagram from Vivado.

Snippet from netlist file

```
// Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
// Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights Reserved.
// ----------------------------------------------------------------------------
// Tool Version: Vivado v.2024.2 (win64) Build 5239630 Fri Nov 08 22:35:27 MST 2024
// Date        : Tue Sep 30 10:47:38 2025
// Host        : HARISH running 64-bit major release  (build 9200)
// Command     : write_verilog -force ./synthesized_netlist.v
// Design      : Full_adder
// Purpose     : This is a Verilog netlist of the current design or from a specific cell of the
design. The output is an
//               IEEE 1364-2001 compliant Verilog HDL file that contains netlist information
obtained from the input
//               design files.
// Device      : xc7a35tcpg236-1
```

Synthesizing Full Adder Design to Gates

```
// ------------------------------------------------------------------------------
`timescale 1 ps / 1 ps

(* STRUCTURAL_NETLIST = "yes" *)
module Full_adder
   (a,b,cin,Sum,Cout);
  input a;
  input b;
  input cin;
  output Sum;
  output Cout;

  wire Cout;
  wire Cout_OBUF;
  wire Sum;
  wire Sum_OBUF;
  wire a;
  wire a_IBUF;
  wire b;
  wire b_IBUF;
  wire cin;
  wire cin_IBUF;

    OBUF Cout_OBUF_inst(.I(Cout_OBUF),.O(Cout));
  (* SOFT_HLUTNM = "soft_lutpair0" *)
  LUT3 #(.INIT(8'hE8))
    Cout_OBUF_inst_i_1 (.I0(cin_IBUF),.I1(b_IBUF),.I2(a_IBUF),.O(Cout_OBUF));
    OBUF Sum_OBUF_inst (.I(Sum_OBUF),.O(Sum));
  (* SOFT_HLUTNM = "soft_lutpair0" *)
  LUT3 #(.INIT(8'h96))
    Sum_OBUF_inst_i_1 (.I0(cin_IBUF),.I1(a_IBUF),.I2(b_IBUF),.O(Sum_OBUF));
    IBUF a_IBUF_inst (.I(a),.O(a_IBUF));
    IBUF b_IBUF_inst (.I(b),.O(b_IBUF));
    IBUF cin_IBUF_inst(.I(cin),.O(cin_IBUF));
endmodule
```

Please note this point

“In FPGA synthesis, my design was mapped into LUT3 primitives with INIT values implementing XOR and majority logic, and connected using IBUF/OBUF primitives for I/Os. This is equivalent to the ASIC netlist shown in the PDF, but uses FPGA standard cells instead of ASIC cells.

Resource utilization report and Timing Summary

You can refer it from utilization_report.txt & timing_report.txt files attached with this document