Winter 2023

COEN 6541 : FUNCTIONAL HARDWARE VERIFICATION

Dr. Otmane Ait Mohamed

CALC1 TEST PLAN PROJECT REPORT

Date : 14-02-2023

Submitted by:

Prithvik Adithya Ravindran - 40195464
Avaneesh Reddy Gaddam - 40217009
Irfanul Islam - 40207134
Sonam Zam Guddati - 40237335

TABLE OF CONTENTS

1. ABSTRACT

This technical report presents the verification of an RTL design implementation of a four-operation calculator. The RTL design was implemented using Verilog HDL and simulated using QuestaSim. The design was verified by performing various tests on the calculator's arithmetic operations. The verification tests were successful for a couple of test cases and failed for a few. This report presents the scenarios for which the calculator design may require further development and testing.

2. INTRODUCTION

Calc1 is an RTL design implementation that has four basic operations. The four operations include:

- Add
- Subtract
- Shift Left
- Shift Right

The calculator operates when a "requestor" sends an operator through the command input bus followed by the data of the operand. It has four input ports & each "requestor" may send the command & data through one of the four ports. They all have equal priority & each port must wait for the previous response before sending the next command. Moreover, all the ports can handle a single command parallelly. Each command will return a response with the result unless it is an error condition.

3. PROJECT DESCRIPTION

A. CALC1 DESIGN DESCRIPTION

The figure below shows the design for Calc1 with a clock signal.

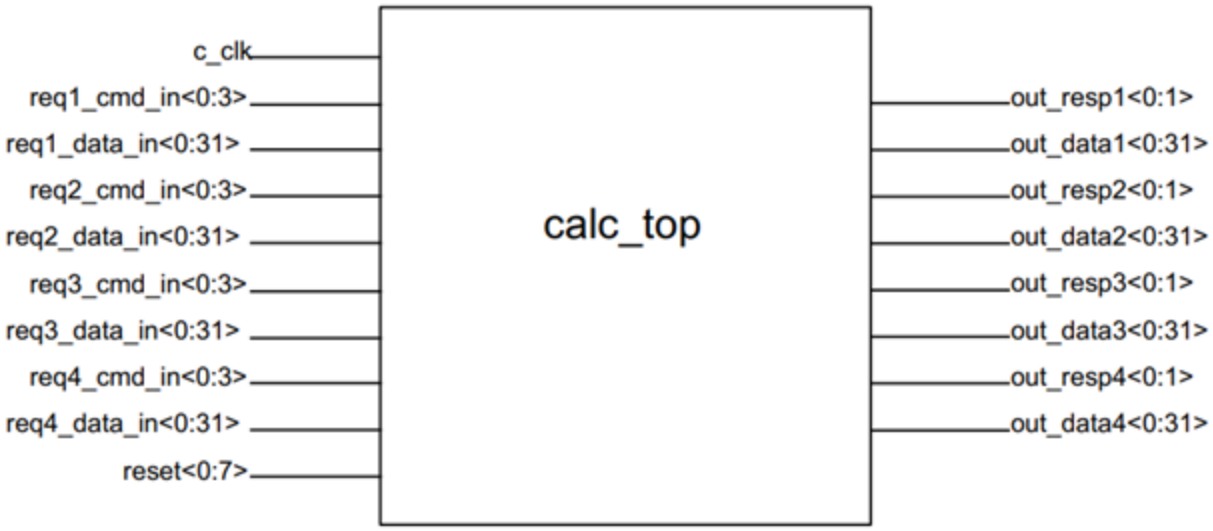

Fig 1.1 Calculator design description.

The calculator has 4 different input ports consisting of a command bus & a data bus, with a clock & a reset bus. On the output side, the calculator has 4 output ports with their own response bus.

B. EXPECTED FUNCTIONALITY

Each of the four Calc1 ports has two different input & output buses. The first input bus, reqX_cmd_in<0:3> (X is the port numbers 1,2,3 or 4) consists of a 4-bit bus through which the command is sent. The possible commands & their respective decode values are given below.

Command	Decode value
No operation	‘0000’b
Add	‘0001’b
Subtract	‘0010’b
Shift Left	‘0101’b
Shift Right	‘0110’b
Invalid	All others

Table 1.1. Calculator command description

For the second input bus, reqX_data_in<0:31> it passes the operand data to the calculator. It takes 2 cycles to send a complete command & operand data. The first cycle, the requestor ports pass the operand 1 data & then operand 2 data in the next cycle.

On the output side there are two buses, the response bus out_respX<0:1> & the result data bus out_dataX<0:31> for each port. The response bus is active for one cycle when the operation is computed for the port. At least 3 cycles are needed to complete an operation, but it can be more & will depend on the other ports. The table below shows the responses for possible cases.

Response Decode	Response meaning
‘00’b	No response on this cycle.
‘01’b	Successful response. Response data is on the output data bus.
‘10’b	Overflow, underflow or invalid command. Overflow/underflow only valid for the add or subtract commands.
‘11’b	Unused response value.

Table 1.2. Calculator output response code

The result data bus should only be sampled when the response bus shows a successful response decode, which will contain the result of the operation for that port.The figure below shows the timing diagram containing the command & response sequence for one successful command on a port.

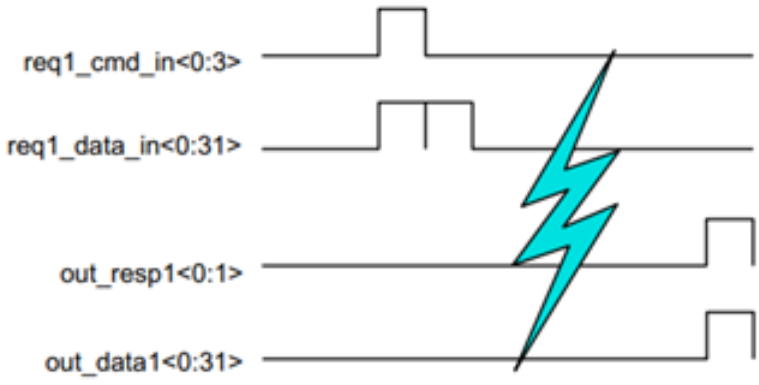

Fig 1.2 Timing diagram

From the figure, it is seen that in the first cycle sequence the command & operand 1 data are sent. The following cycle, the operand 2 data is sent. After a few cycles passes, the response & result appears on the output side of the calculator.

Only one operation should be ongoing in each port. When a port sends a command, it has to wait for the response of the current command before sending another command. The port can also be idle for any number of cycles in between commands. Each port is independent of each other & receive commands parallelly. So, the calculator may work on a maximum of four commands. The responses for all the ports won't be concurrent even though each port has equal priority as there are limited resources. The design contains two ALUs, one for adds & subtracts, while the other for shift operations. Therefore, the design logic would serialize the responses when concurrent commands are sent. The commands are serviced on a first come, first served basis.

From the calculator design, it can be seen that there is a bus called reset<0:7> on the input side. It is used for clearing the internal state of the calculator. When reset<0:7> is set to '11111111'b it activates the reset. This value needs to be held for seven consecutive cycles for a complete reset of the calculator. All other input buses except for c_clk should be set to zero during reset.

The arithmetic operands are treated as unsigned data. The most significant bit is a data bit not a sign bit. During overflow in an add operation, the high order bit has a carry-out while underflow occurs in a subtract operation. The table below shows some overflow/underflow examples with their responses.

Command	Operand1	Operand2	Response	Result data
Add	"80002345"X	"00010000"X	Successful	"80012345"X
Add	"FFFFFFFF"X	"00000001"X	Overflow	None
Subtract	"FFFFFFFF"X	"11111111"X	Successful	"EEEEEEEE"X
Subtract	"11111111"X	"20000000"X	Underflow	None

Table 1.3 Calculator command examples

C. VERIFICATION TESTING REQUIREMENTS

Calc1 has a simple design & requires verification at the top level. Also, the specifications mentioned only describes the top level. Hence, direct testing method is used to verify the different operations & ports functioning properly.

4. TOOLS USED

The software QuestaSim is used for the simulation & verification of Calc1.

5. TEST PLAN

A. TEST PLAN FOR BASIC FUNCTION TESTS:

S.No.	Test Case Description
1	Perform ADD operation on two operands and obtain successful response at respective response bus port.
2	Perform SUBTRACT operation on two operands and obtain successful response at respective response bus port.
3	Perform SHIFT RIGHT operation on two operands and obtain successful response at respective response bus port.
4	Perform SHIFT RIGHT operation on two operands and obtain successful response at respective response bus port.
5	Perform ADD operation on two operands and obtain the result operand1 + operand2 at respective result data bus port.
6	Perform SUBTRACT operation on two operands and obtain the result operand1 - operand2 at respective result data bus port.
7	Perform SHIFT RIGHT operation on two operands and obtain the result operand1 shifted right operand2 places* at the respective result data bus port.
8	Perform SHIFT RIGHT operation on two operands and obtain the result operand1 shifted left operand2 places* at the respective result data bus port.
9	Obtain OVERFLOW response at the respective output response port when the high order bit (bit 0) has a carry-out in ADD operation.
10	Obtain UNDERFLOW response at the respective output response port when a larger number is subtracted from a smaller number.
11	Obtain no result at the respective output result data port when the high order bit (bit 0) has a carry-out in ADD operation.
12	Obtain no result at the respective output result data port when a larger number is

	subtracted from a smaller number.

Table 1.4 Test case description

*For both shift commands, only the low order 5 bits (reqX_data_in(27:31)) of operand2 (the shift amount) are used. The Calc1 logic ignores bits 0 to 26 of the shift operand2. This allows the operand1 data to be shifted any amount from 0 to 31 places (inclusive).

B. TEST PLAN FOR ADVANCED FUNCTION TESTS:

S.No.	Test Case Description
1	For each port, check that each command can have any command follow it without leaving the state of the design dirty such that the folowing command is corrupted.
2	Across all ports (eg. four concurrent ADDs doesnt interfere with each other) check that each command can have any command follow it without leaving the design dirty, such that the folowing command is corrupted.
3	Check that there is fairness across all four ports such that no port has higher priority than others
4	Check that high order 27 bits are ignored in the second operand of shift commands

Table 1.5. Advanced test plan

C. TEST PLAN FOR ADVANCED FUNCTION TESTS - DATA DEPENDENT CASES:

S.No.	Test Case Description
1	Add two numbers that overflow by 1. Expected output response: 10’b (OVERFLOW)
2	Add two numbers whose sum is “FFFFFFFF”X.Expected output response:01’b Expected output result: FFFFFFFF
3	Subtract two equal numbers. Expected output response: 01’b Expected output result: 0
4	Subtract a number that underflows by 1. Expected output response: 10’b (UNDERFLOW)
5	Shift left 0 places. Expected output response: 01’b Expected output result: Operand 1
6	Shift right 0 places. Expected output response: 01’b

	Expected output result: Operand 1
7	Shift left 31 places (max allowable shift places). Expected output response: 01’b
8	Shift right 31 places (max allowable shift places). Expected output response: 01’b

Table 1.6. Data dependent case test plan

D. TEST PLAN FOR GENERIC FUNCTION TESTS AND CHECKS:

S.No.	Test Case Description
1.	Check that the design correctly handles illegal commands
2.	Check all outputs all the time. Calc1 should not generate superfluous output values.
3.	Check that the reset function correctly resets the design

TABLE 1.7. Generic function test plan

6. VERIFICATION RESULTS

The verification test results are given in the attached Excel sheet with the report.

7. BUG REPORT

No	CASE	BUG
1	Invalid Command	For an invalid command, the response shown is successful instead of invalid.
2	Operations in Port 4	The port does not support ADD & SUB operations.
3	ADD operators in non-sequential order	When ADD operators are run non-sequentially there seems to be a priority in the output side which executes sequentially from ports 1 to 4.
4	Adding 2 numbers that overflow by 1	Gives the successful response ‘01’b instead of the expected response ‘10’b.
5	Subtracting a number that underflows by 1	Gives the successful response ‘01’b instead of the expected response ‘10’b.
6	Shift 31 places in data dependent corner case	Shift right working but shift left is not.

Table 1.8. Bug Report Table

8. CONCLUSION

From the verification test results, it can be concluded that the design has quite a few bugs that are hampering the design from working perfectly under certain scenarios. It was a good thing direct testing method was used for the verification of the design. But, practically this method is not feasible as this project had a simple RTL design whereas larger designs with direct testing would be very critical & time consuming. It is because each test case would then be needed to be thought out & translated to the code for each case to verify the system. As a result, random testing is preferred compared to direct testing in larger designs.