Calc Project Overview

Design Specs

Overview of Calc designs

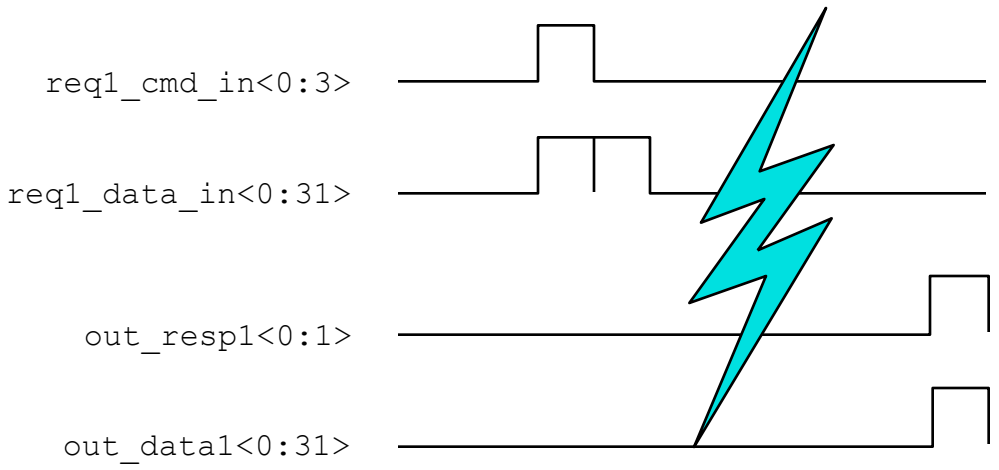
- Two designs are provided
  - Calc1
  - Calc2

Calc1

- Calc1 is a simple 4-port calculator.
- Each port can send in 1 command at a time.
- The commands consist of add, subtract, shift left, and shift right.
- The operand data accompanies the commands.
- Since there are 4 ports, there can be up to 4 outstanding commands at once.

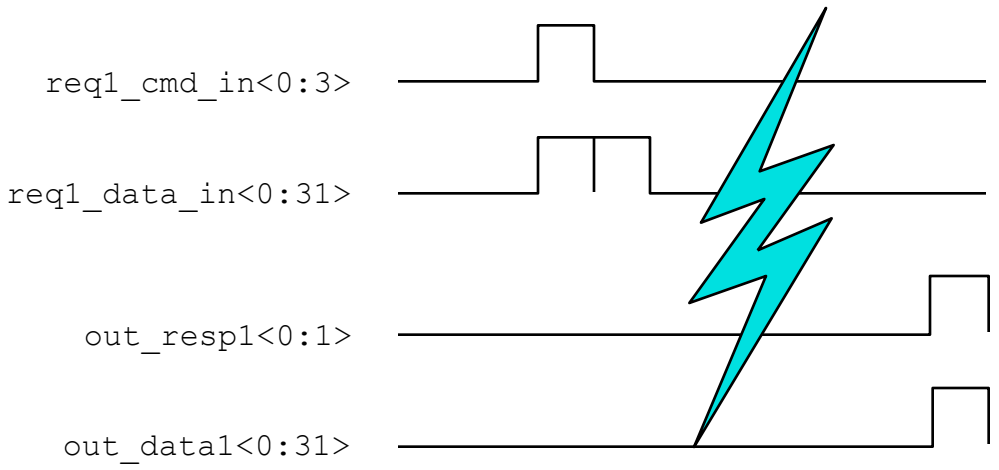
Calc1

- The input/Output timing on a single port is as follows:



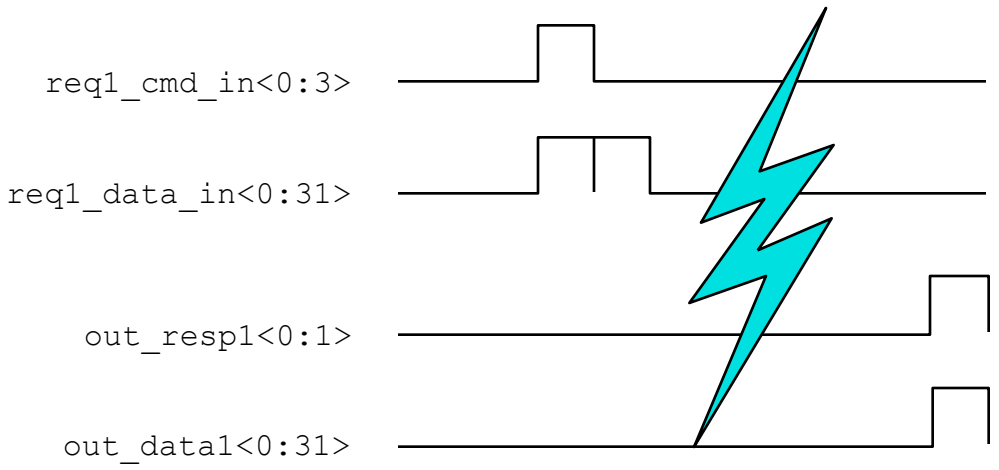
Calc1

- The lightning bolt represents “some number of cycles passing.”



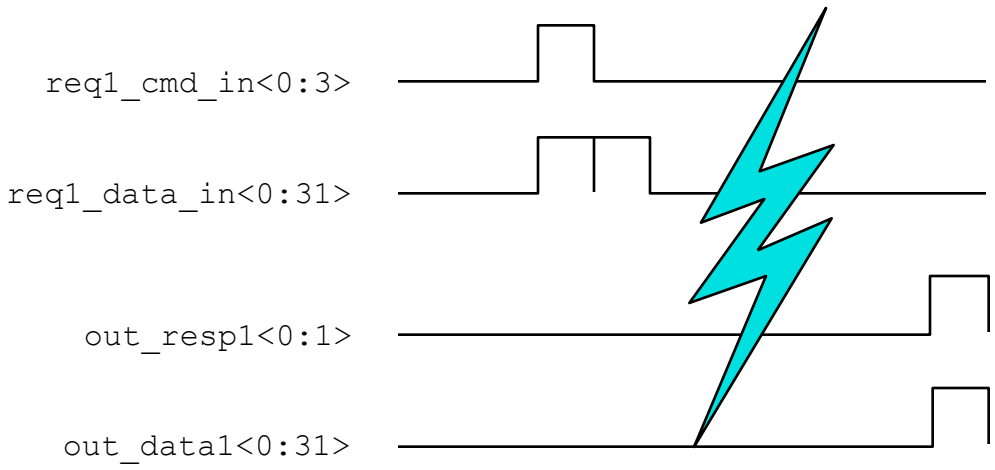
Calc1

- A second request from the same port is prohibited until the response from the first command is received.



Calc1

- Internally, there are two ALUs: one that processes all add commands, and the second for all shift commands. Priority logic dispatches the individual commands to the ALUs.



Calculator Design

- Calculator has 4 functions:
  - Add
  - Subtract
  - Shift left
  - Shift right
- Calculator can handle 4 requests in parallel
  - All 4 requestors use separate input signals
  - All requestors have equal priority
  - Each port must wait for its response prior to sending the next command

Calculator Design

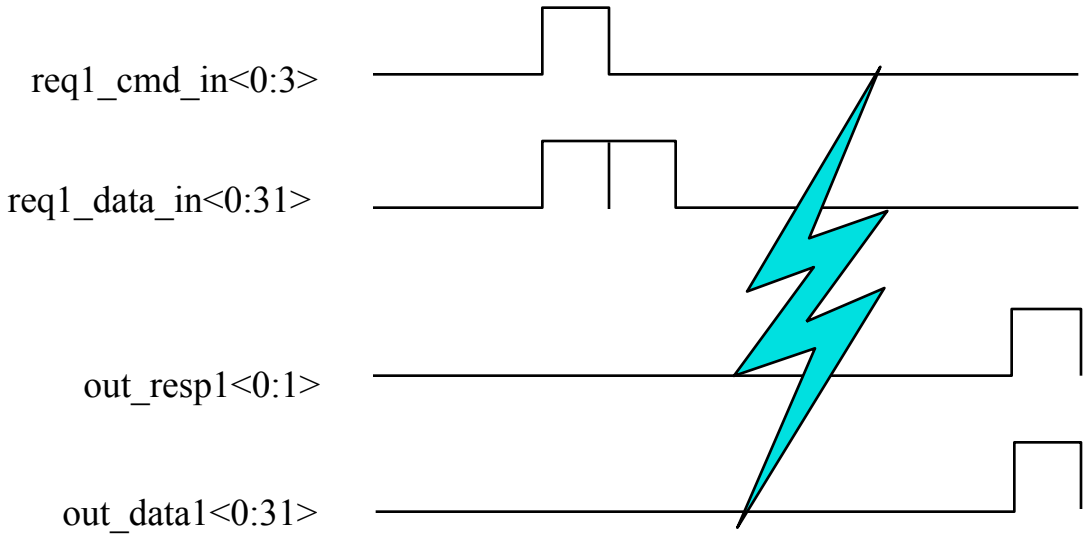
- I/O Description
  - Input commands:
    - 0 - No-op
    - 1 - Add operand1 and operand2
    - 2 - Subtract operand2 from operand1
    - 5 - Shift left operand1 by operand2 places
    - 6 - Shift right operand1 by operand2 places
  - Input Data
    - Operand1 data arrives with command
    - Operand2 data arrives on the following cycle

Calculator Design

- Outputs
  - Response line definition
    - 0 - no response
    - 1 - successful operation completion
    - 2 - invalid command or overflow/underflow error
    - 3 - Internal error
  - Data
    - Valid result data on output lines accompanies response (same cycle)

Calculator Design

- Input/Output timing



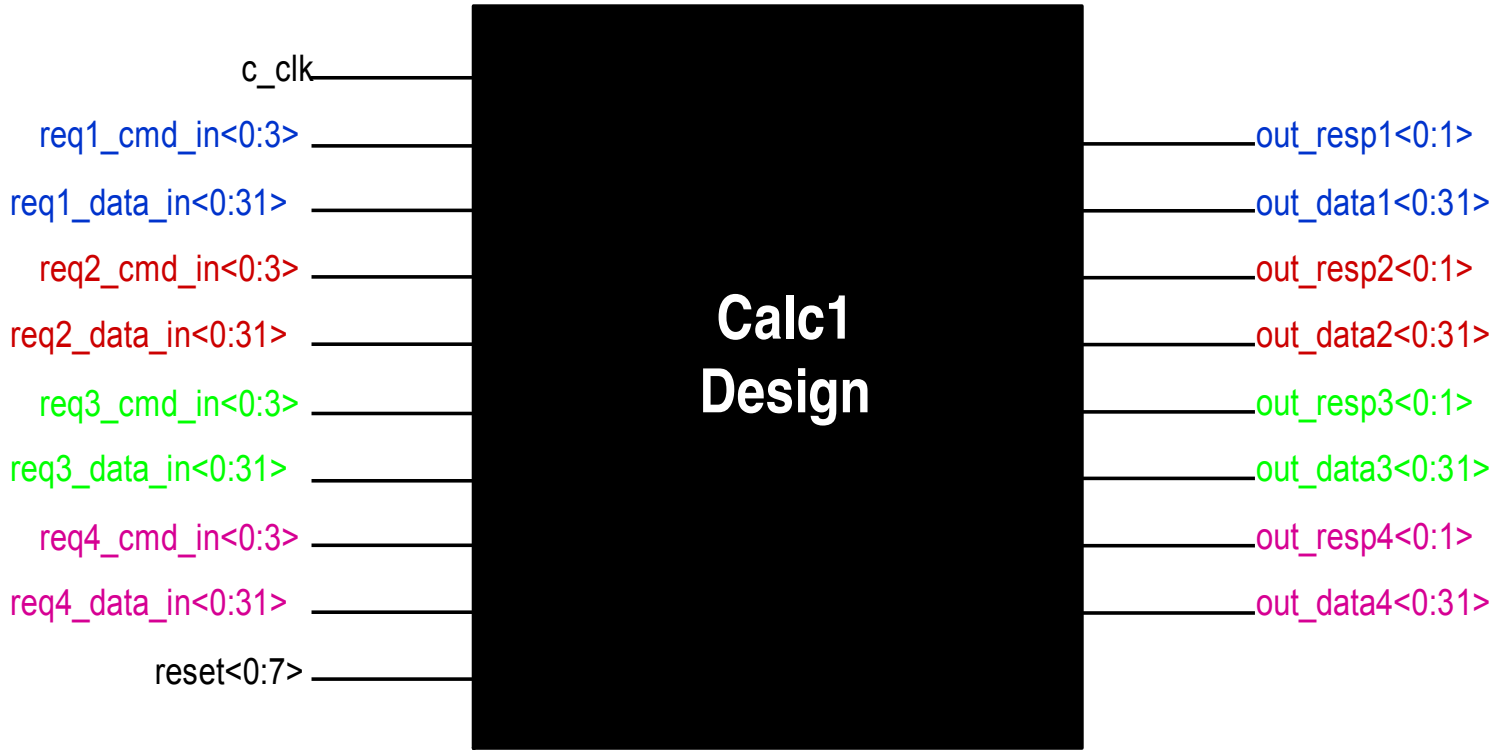
Each port must wait for its response prior to sending the next command!

Calculator Design

- Other information
  - Clocking
    - When using a cycle simulator, the clock should be held high (c_clk in the calculator model)
    - The clock should be toggled when using an event simulator
  - Calculator priority logic
    - Priority logic works on first come first serve algorithm
    - Priority logic allows for 1 add or subtract at a time and one shift operation at a time

Calculator Design

- Other information (con't)
  - Resets
    - Hold reset(1:7) to '1111111'b at start of testcase for seven cycles.
    - During the reset period, outputs of the calculator should be ignored
  - Shift operation
    - Only the low order 5 bits of the second operand are used
  - Arithmetic operations are unsigned


c_clk
req1_cmd_in<0:3>
req1_data_in<0:31>
req2_cmd_in<0:3>
req2_data_in<0:31>
req3_cmd_in<0:3>
req3_data_in<0:31>
req4_cmd_in<0:3>
req4_data_in<0:31>
reset<0:7>
Calc1
Design
out_resp1<0:1>
out_data1<0:31>
out_resp2<0:1>
out_data2<0:31>
out_resp3<0:1>
out_data3<0:31>
out_resp4<0:1>
out_data4<0:31>

Calc2

- The second design, Calc2, is much like the first, except that each port may now have up to 4 outstanding commands at a time.

Calc2

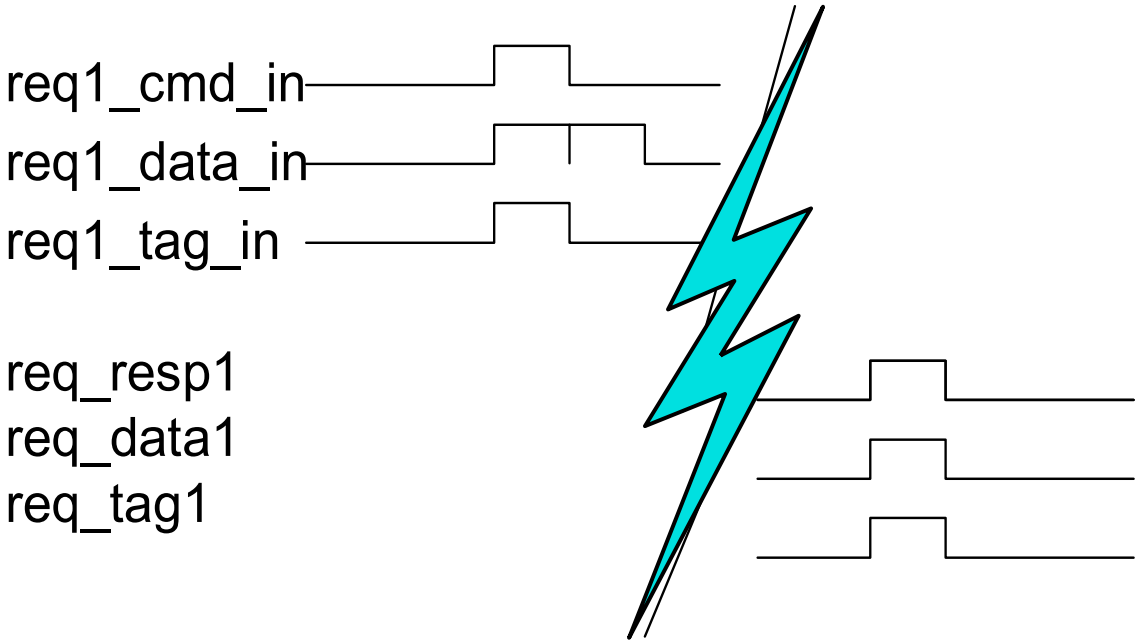
- Each command from a port is sent in serially (one at a time), but the calc log may respond "out of order," depending upon the internal state of the queues.

Calc2

- As a result, each command is now accompanied by a 2-bit tag, which identifies the command when the response is received.

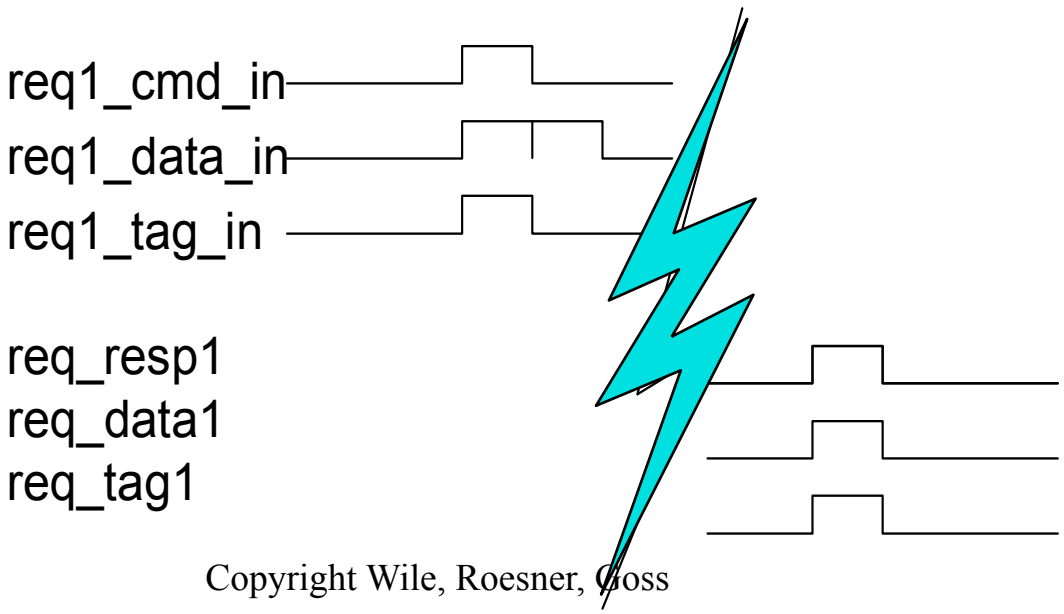
Calc2

- A possible timing diagram:



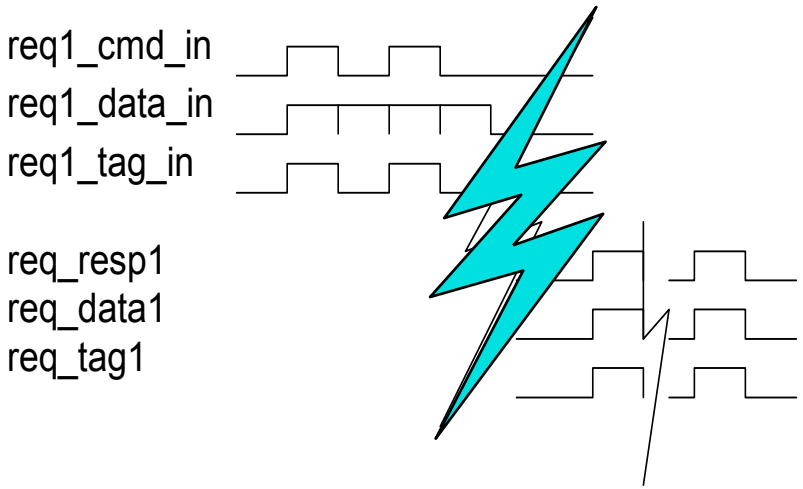
Calc2

- The lightning bolt represents “some number of cycles passing.” The data accompanies a successful response signal.



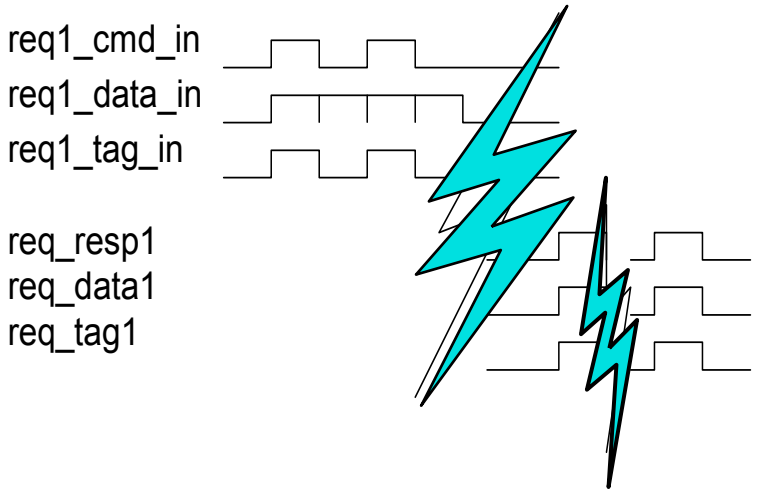
Calc2

- A timing diagram of back-to-back commands might look like:



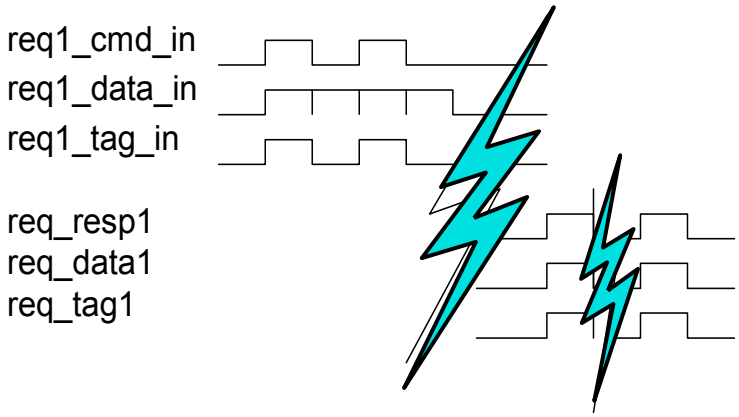
Calc2

- The lightning bolt represents “some number of cycles passing.”



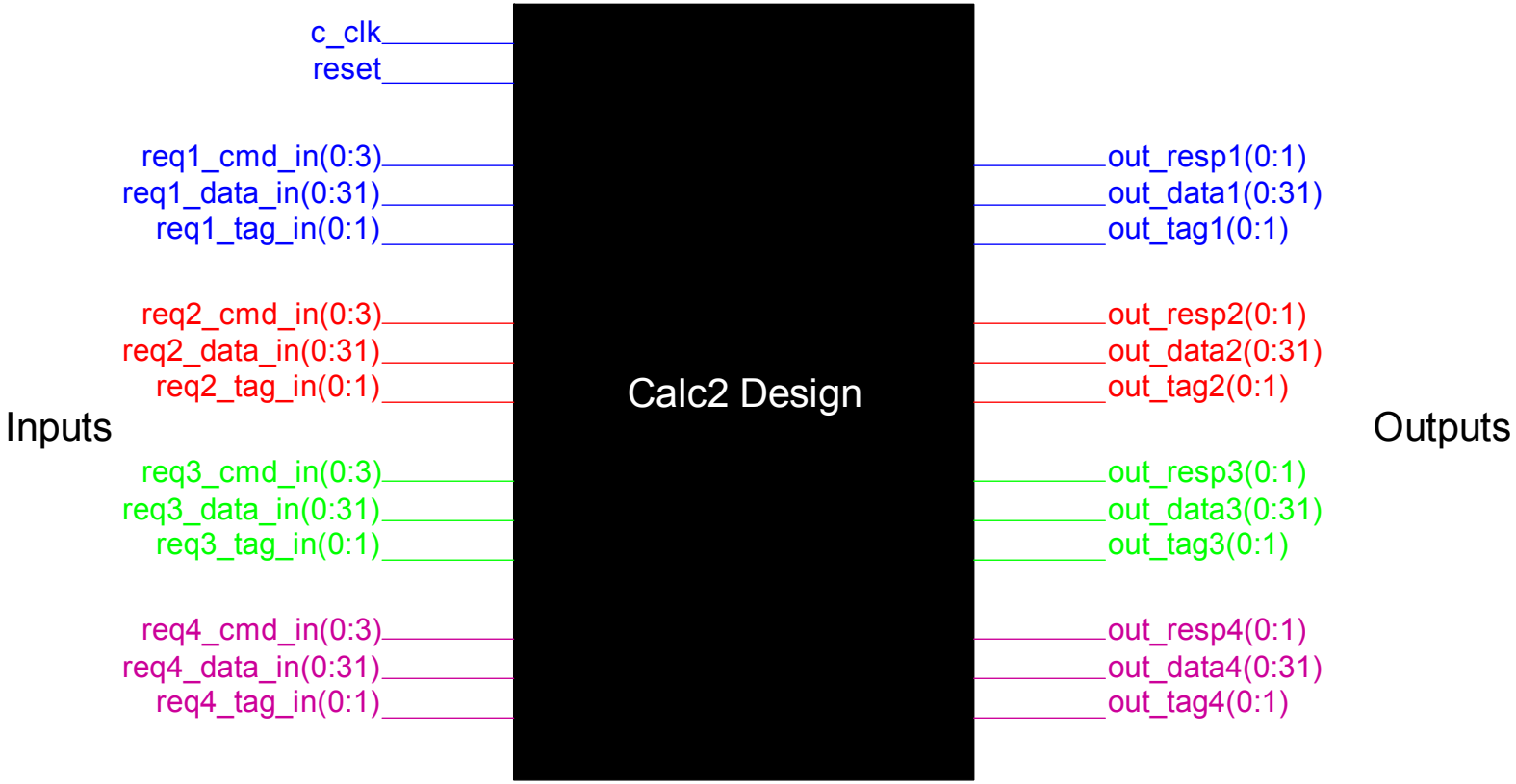
Calc2

- The data accompanies a successful response signal. Responses may or may not be in order, but the tags will identify which command the response is for.



Calc2

- There may be up to 16 commands outstanding at once (4 ports, 4 commands each).
- As in Calc1, the commands are add, subtract, shift left, and shift right.


Inputs
c_clk
reset
req1_cmd_in(0:3)
req1_data_in(0:31)
req1_tag_in(0:1)
req2_cmd_in(0:3)
req2_data_in(0:31)
req2_tag_in(0:1)
req3_cmd_in(0:3)
req3_data_in(0:31)
req3_tag_in(0:1)
req4_cmd_in(0:3)
req4_data_in(0:31)
req4_tag_in(0:1)
Calc2 Design
Outputs
out_resp1(0:1)
out_data1(0:31)
out_tag1(0:1)
out_resp2(0:1)
out_data2(0:31)
out_tag2(0:1)
out_resp3(0:1)
out_data3(0:31)
out_tag3(0:1)
out_resp4(0:1)
out_data4(0:31)
out_tag4(0:1)

Functional Verification

- Test Plan
- Use SystemVerilog to build your verification environment for Calc1,Calc2.
- Your environment should be class based.
- You should use:
  - Interface
  - Clocking blocks that is sensitive to the falling edge of the clock, and all I/O that are synchronous to the clock
  - A modport for the testbench called master, and modport for the DUT called slave

Functional Verification

- Random verification
- Functional Coverage
- You need to report all the bugs founds in Calc1,Calc2.

Functional verification

- Submit:
  - Report for all students
  - All SV code