Calc3 Overview

- Design now has 16 internal data registers
  - Arithmetic operands no longer sent by requestor
  - Operand data is read internally from registers

Calc3 Overview

- Two new commands added to access registers
  - Fetch from register x
  - Store to register x

Calc3 Overview

- Two new branch commands
  - Successful branch causes next command from port to be skipped

Calc3 Overview

- Each requestor can still send in up to 4 commands
  - 2-bit tag on request
  - Using same tag simultaneously is not supported

Calc3 Overview

- Each port requestor is sending an instruction stream
  - In the first two Calc designs, the data accompanied the command. But in this design, the arithmetic ops reference operand registers internal to the design. Therefore, instruction ordering ("instruction stream") concepts must be obeyed by the design so that within in each port, the commands may only proceed out-of-order when the operand registers do not conflict.
  - The ordering rules are shown in the following slides.

I/O Protocols

For each requestor X:

Inputs:

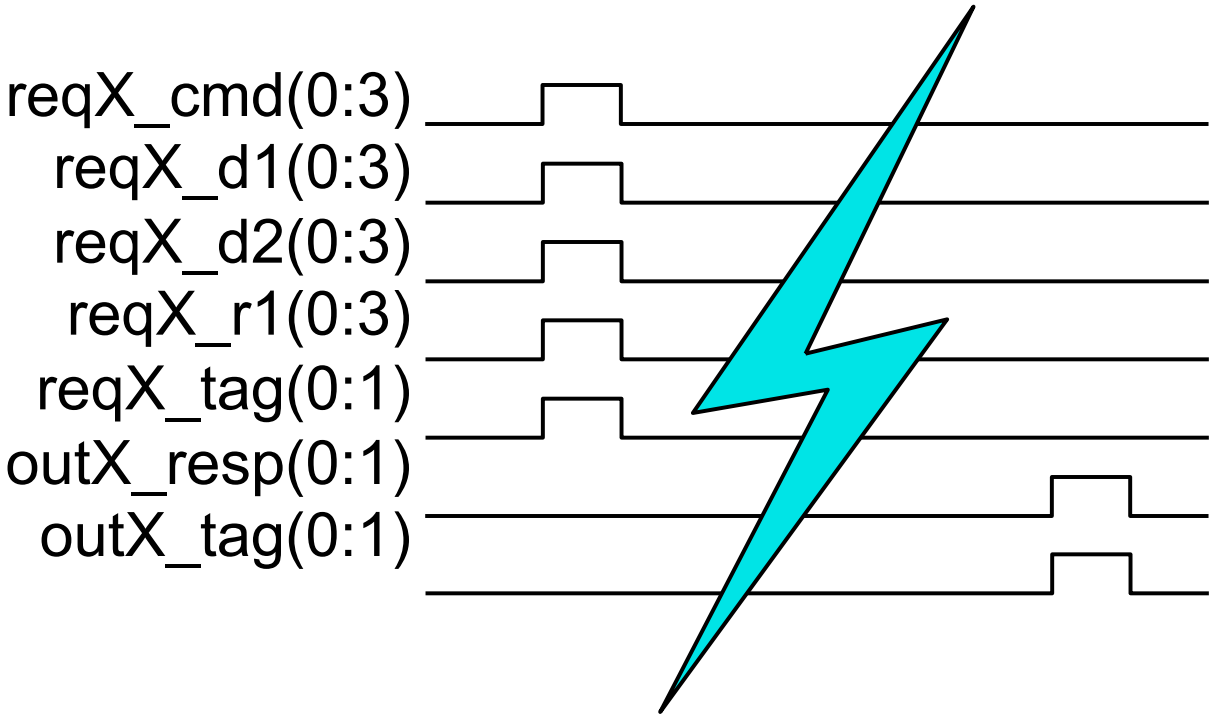
- reqX_cmd(0:3)
  - add: 0001 adds contents of d1 to d2 and stores in r1
  - subtract: 0010 subtracts contents of d2 from d1 and stores in r1
  - shift left: 0101 shifts contents of d1 to the left d2(27:31) places and stores in r1
  - shift right:0110 shifts contents of d1 to the right d2(27:31) places and stores in r1
  - store: 1001 stores reqX_data(0:31) into r1
  - fetch:1010 fetches contents of d1 and outputs it on out_dataX(0:31)
  - branch if zero: 1100 skip next valid command if contents of d1 are 0
  - branch if equal: 1101 skip next valid command if contents of d1 and d2 are equal
- reqX_d1(0:3)
- reqX_d2(0:3)
- reqX_r1(0:3)
- reqX_tag(0:1)
- reqX_data(0:31)

Outputs:

- outX_resp(0:1)
  - Successful completion: 01
  - overflow/underflow error:10
  - Command skipped due to branch: 11
- outX_tag(0:1)
- outX_data(0:31)

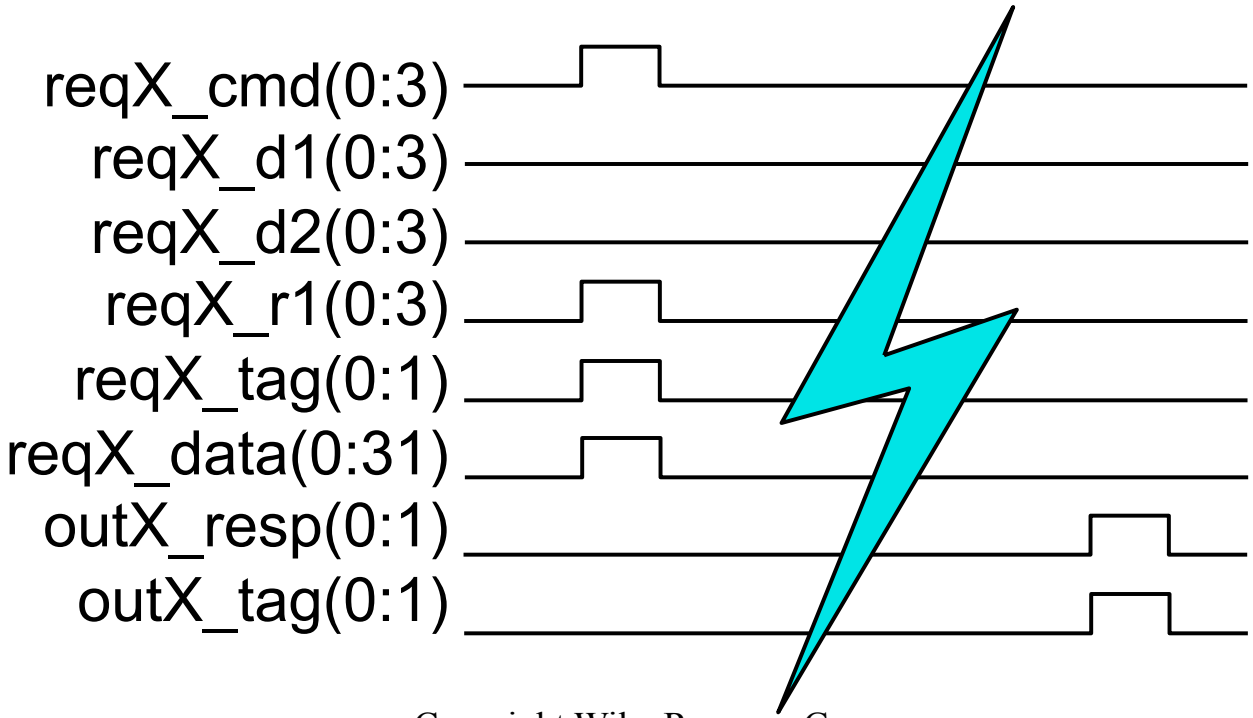
I/O Timing

- Basic timing of a single arithmetic command:



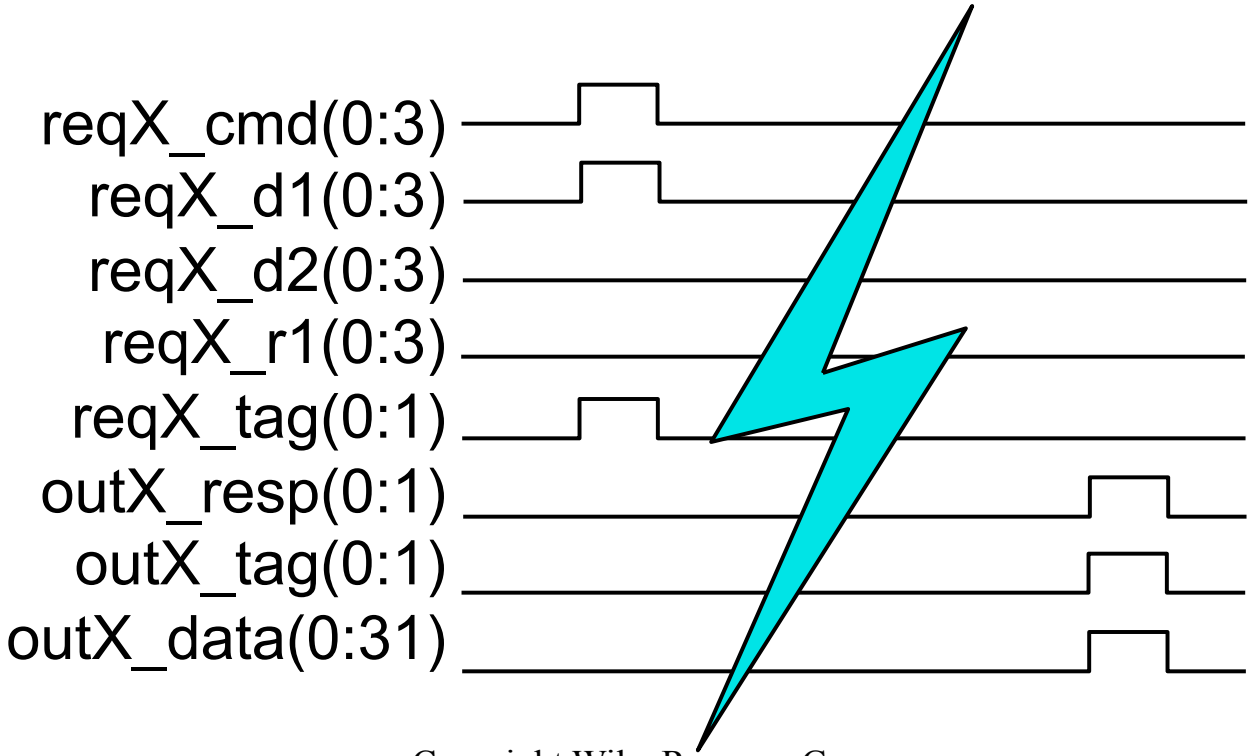
I/O Timing

- Basic timing of a store command:



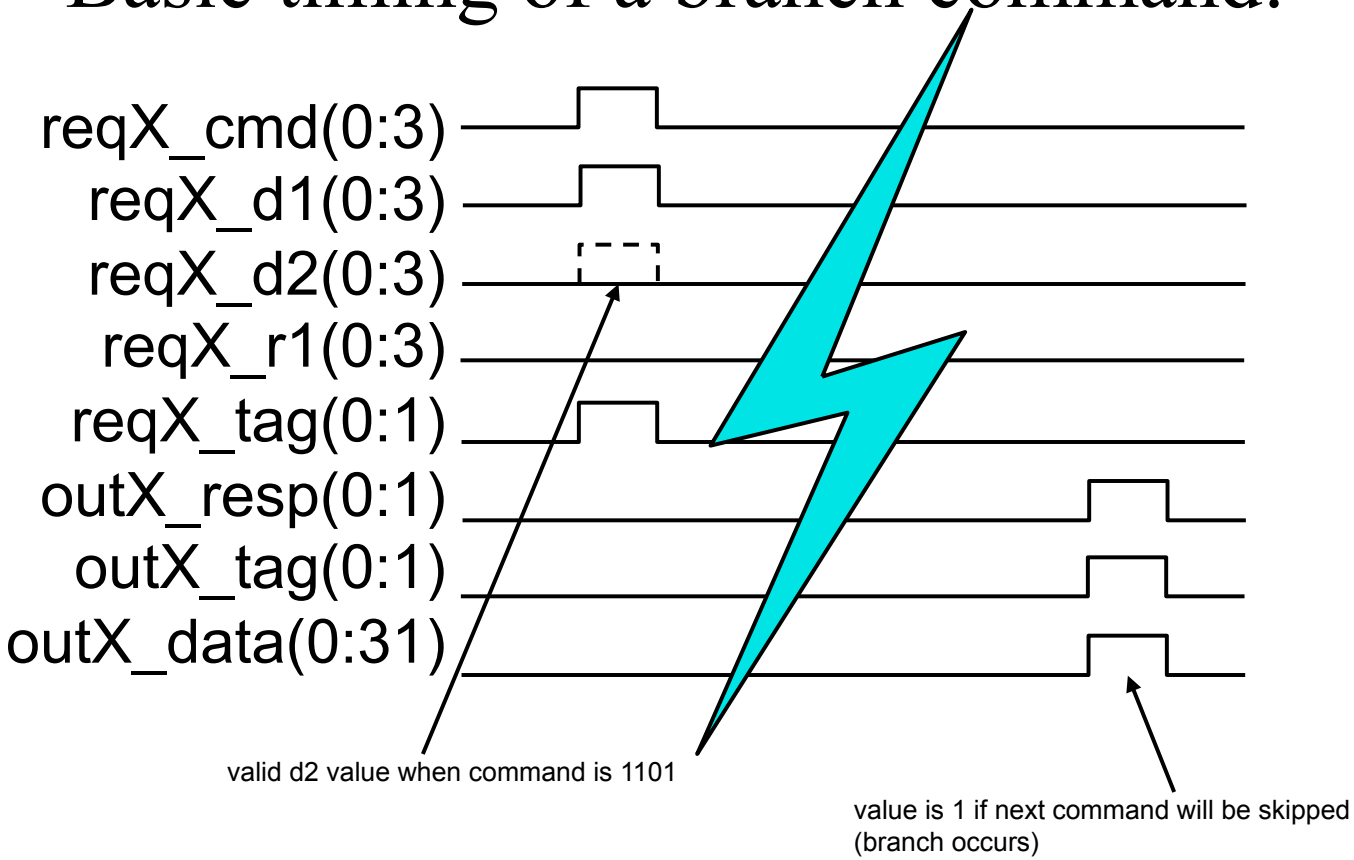
I/O Timing

- Basic timing of a fetch command:



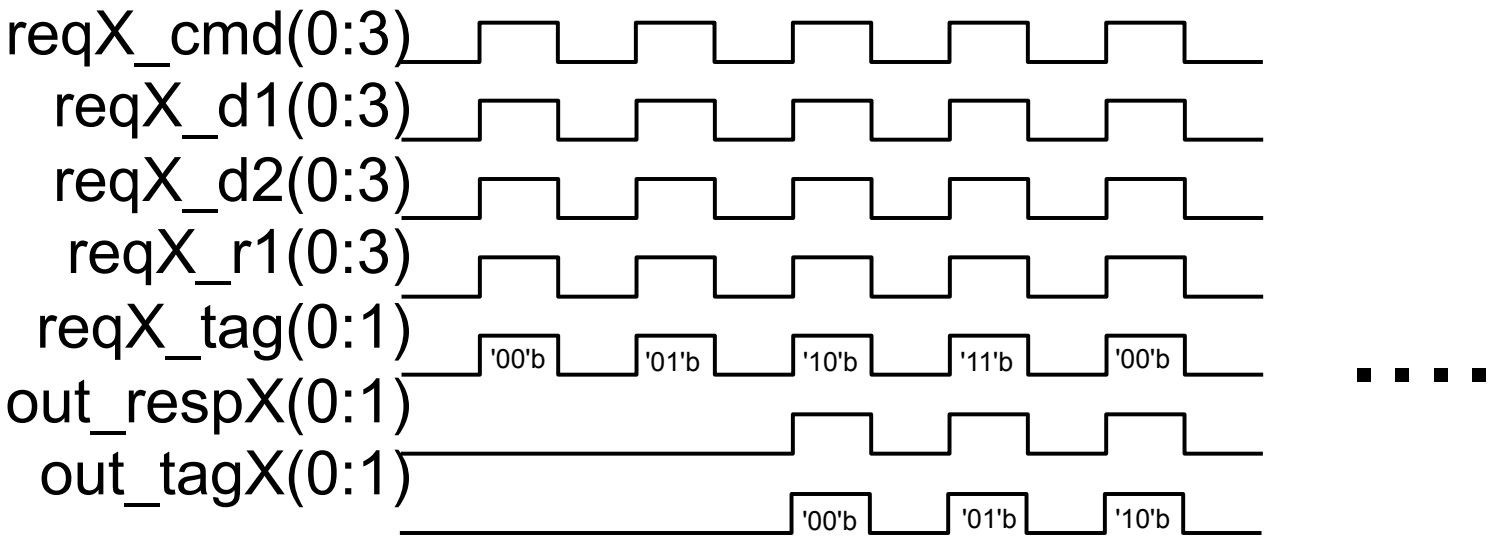
I/O Timing

- Basic timing of a branch command:



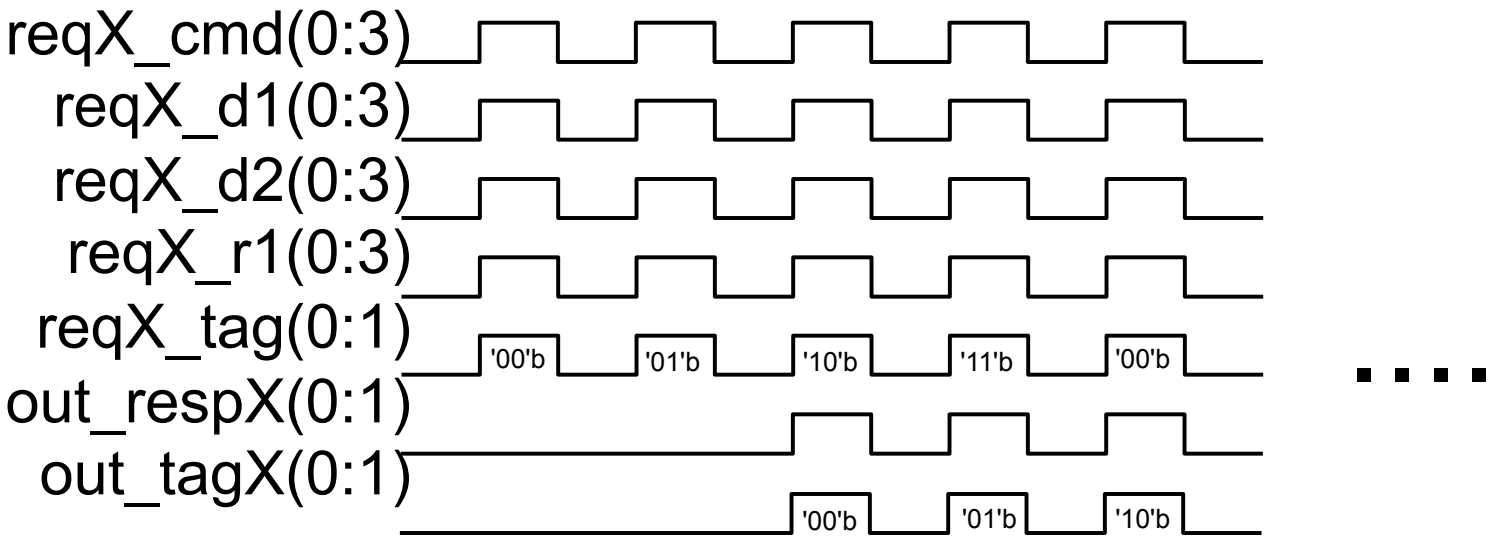
I/O Timing

- Fastest multiple command (any cmd type) timing (example is if only one requestor is sending commands):



I/O Timing

- Requestor must leave dead cycle between commands.
- Requestor may only have 4 commands outstanding at a time, each with a unique tag.



Command That Follows a Branch

- Any valid command can follow a branch.
- If the branch evaluates true, the following command will be “skipped”:
  - Add/Sub/SL/SR will not write to array
  - Store will not write to array
  - Fetch will not return data
  - Brach will evaluate to false (case of branch followed by branch)

Command That Follows a Branch

- Response code of ’11’b for follower indicating above action has occurred.
- Invalid OP codes are ignored and are NOT considered to “command that follows a branch.”

Command and Ordering Rules

- Within each requestor's (port's) instruction stream, operations can complete out of order with the follow restrictions:
  - Operands (d1, d2) cannot be used if prior instruction in stream writes (result r1) to the operand and prior instruction has not completed.
  - Results (r1) cannot be written if either of the prior command operands (d1, d2) use the same register as R1.
  - Same R1 (result) values from different instructions must complete in order.
- There are no restrictions of this type across different requestors.