

IP Core  
Specification

*Author: Tom Burke*

*tomburkeii@gmail.com*

**Rev. 0.1**

**January 5, 2014**

*This page has been intentionally left blank.*

**Revision History**

| **Rev.** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 0.1 | 5 Jan 2014 | T. Burke | First Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contents**

Introduction 1

Architecture 3

qadd.v 3

qmult.v 4

qmults.v 5

qdiv.v 5

Operation 6

qadd.v 6

qmult.v 7

qmults.v 8

qdiv.v 10

Registers 12

List of Registers 12

Clocks 15

IO Ports 16

qadd.v 16

qmult.v 16

qmults.v 17

qdiv.v 18

Introduction

Verilog Fixed point math library

Original work by Sam Skalicky, originally found [here](http://opencores.org/project,fixed_point_arithmetic_parameterized)

Extended, updated, and heavily commented by Tom Burke

This library includes the basic math functions for the Verilog Language,   
for implementation on FPGAs.

All units have been simulated and synthesized for Xilinx Spartan 3E devices  
using the Xilinx ISE WebPack tools v14.7

These math routines use a signed magnitude Q,N format, where N is the total   
number of bits used, and Q is the number of fractional bits used. For instance,   
15,32 would represent a 32-bit number with 15 fractional bits, 16 integer bits,   
and 1 sign bit as shown below:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

This library contains the following modules:

|  |  |  |
| --- | --- | --- |
| qadd.v | - | Addition module; adds 2 numbers of any sign. |
| qdiv.v | - | Division module; divides two numbers using a right-shift and subtract algorithm - requires an input clock |
| qmult.v | - | Multiplication module; purely combinational for systems that will support it |
| qmults.v | - | Multiplication module; uses a left-shift and add algorithm - requires an input clock (for systems that cannot support the synthesis of a combinational multiplier) |
| Test\_add.v | - | Test fixture for the qadd.v module |
| Test\_mult.v | - | Test fixture for the qmult.v module |
| TestDiv.v | - | Test fixture for the qdiv.v module |
| TestMultS.v | - | Test fixture for the qmults.v module |

These math routines default to a (Q,N) of (15,32), but are easily customizable   
to your application by changing their input parameters. For instance, an   
unmodified use of (15,32) would look something like this:

qadd my\_adder(

.a(addend\_a),

.b(addend\_b),

.c(result)

);

To change this to an (8,23) notation, for instance, the same module would be   
instantiated thusly:

qadd #(8,23) my\_adder(

.a(addend\_a),

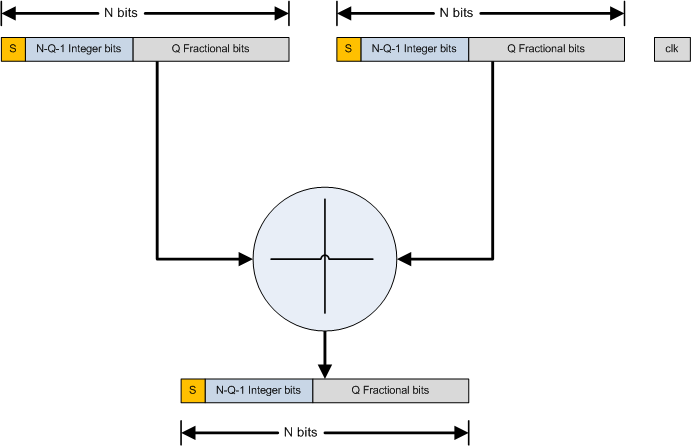
.b(addend\_b),

.c(result)

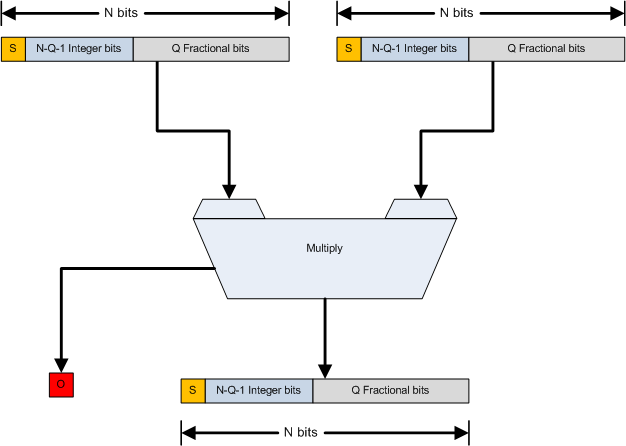
);

Architecture

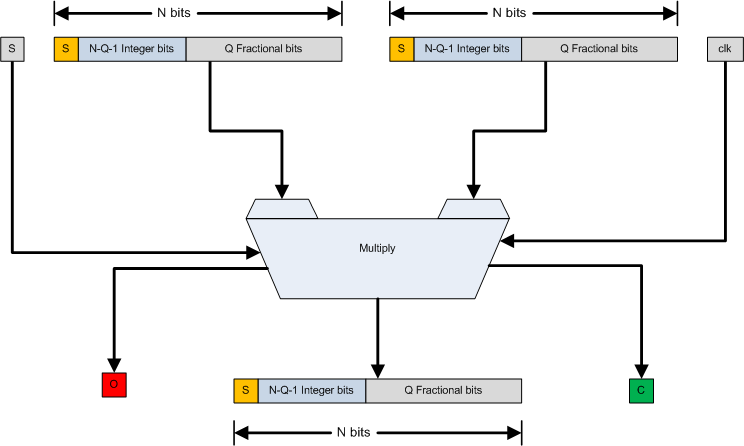
### qadd.v



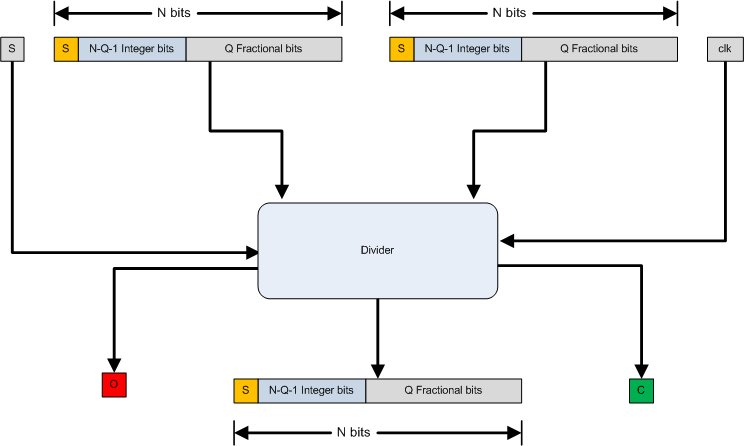
### qmult.v



### qmults.v



### qdiv.v



Operation

### qadd.v

A simple combinational addition module.

sum = addend + addend

Input format:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

Inputs:

a - addend 1

b - addend 2

Output format:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

Output:

c - result

NOTE: There is no error detection for an overflow. It is up to the designer to ensure that an overflow cannot occur!!

Example usage:

qadd #(Q,N) my\_adder(

.a(addend\_a),

.b(addend\_b),

.c(result)

);

For subtraction, set the sign bit for the negative number. (subtraction is the addition of a negative, right?)

### qmult.v

A simple combinational multiplication module.

Input format:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

Inputs:

|  |  |  |
| --- | --- | --- |
| i\_multiplicand | - | multiplicand |
| i\_multiplier | - | multiplier |

Output format:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

Output:

|  |  |  |
| --- | --- | --- |
| o\_result\_out | - | result |
| o\_complete | - | computation complete flag; asserted ("1") when the operation is completed |
| o\_overflow | - | overflow flag; asserted ("1") to indicate that an overflow has occurred. |

|  |  |
| --- | --- |
| **NOTE:** | This module assumes a system that supports the synthesis of combinational multipliers. If your device/synthesizer does not support this for your particular application, then use the "qmults.v" module. |
| **NOTE:** | Notice that the output format is identical to the input format! To properly use this module, you need to either ensure that you maximum result never exceeds the format, or incorporate the overflow flag into your design. |

Example usage:

qmult #(Q,N) my\_multiplier(

.i\_multiplicand(multiplicand),

.i\_multiplier(multiplier),

.o\_result(result),

.ovr(overflow\_flag)

);

### qmults.v

A multi-clock multiplication module that uses a left-shift and add algorithm.

result = multiplicand x multiplier

Input format:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

Inputs:

|  |  |  |
| --- | --- | --- |
| i\_multiplicand | - | multiplicand |
| i\_multiplier | - | multiplier |
| i\_start | - | Start flag; set this bit high ("1") to start the operation when the last operation is completed. This bit is ignored until o\_complete is asserted. |
| i\_clk | - | input clock; internal workings occur on the rising edge |

Output format:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

Output:

|  |  |  |  |
| --- | --- | --- | --- |
| o\_result\_out | | - | result |
| o\_complete | | - | computation complete flag; asserted ("1") when the operation is completed |
| o\_overflow | | - | overflow flag; asserted ("1") to indicate that an overflow has occurred. |
| **NOTE:** | | This module is "time deterministic." - that is, it should always take the same number of clock cycles to complete an operation, regardless of the inputs (N+1 clocks) | | | |
| **NOTE:** | | Notice that the output format is identical to the input format! To properly use this module, you need to either ensure that you maximum result never exceeds the format, or incorporate the overflow flag into your design | | | |

Example usage:

qmults #(Q,N) my\_multiplier(

.i\_multiplicand(multiplicand),

.i\_multiplier(multiplier),

.i\_start(start),

.i\_clk(clock),

.o\_result(result),

.o\_complete(done),

.o\_overflow(overflow\_flag)

);

The qmults.v module begins computation when the start conditions are met:

o\_complete == 1'b1;

i\_start == 1'b1;

### qdiv.v

A multi-clock division module that uses a right-shift and add algorithm.

quotient = dividend / divisor

Input format:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

Inputs:

|  |  |  |
| --- | --- | --- |
| i\_dividend | - | dividend |
| i\_divisor | - | divisor |
| i\_start | - | Start flag; set this bit high ("1") to start the operation when the last operation is completed. This bit is ignored until o\_complete is asserted. |
| i\_clk | - | input clock; internal workings occur on the rising edge |

Output format:

|1|<- N-Q-1 bits ->|<--- Q bits -->|

|S|IIIIIIIIIIIIIIII|FFFFFFFFFFFFFFF|

Output:

|  |  |  |  |
| --- | --- | --- | --- |
| o\_quotient\_out | | - | result |
| o\_complete | | - | computation complete flag; asserted ("1") when the operation is completed |
| o\_overflow | | - | overflow flag; asserted ("1") to indicate that an overflow has occurred. |
| **NOTE:** | | This module is "time deterministic." - that is, it should always take the same number of clock cycles to complete an operation, regardless of the inputs (N+Q+1 clocks) | | | |
| **NOTE:** | | Notice that the output format is identical to the input format! To properly use this module, you need to either ensure that you maximum result never exceeds the format, or incorporate the overflow flag into your design | | | |

Example usage:

qdiv #(Q,N) my\_divider(

.i\_dividend(dividend),

.i\_divisor(divisor),

.i\_start(start),

.i\_clk(clock),

.o\_quotient\_out(result),

.o\_complete(done),

.o\_overflow(overflow\_flag)

);

The qdiv.v module begins computation when the start conditions are met:

o\_complete == 1'b1;

i\_start == 1'b1;

Registers

### List of Registers

#### qadd.v

None – combinational only

#### qmult.v

None – combinational only

#### qmults.v

| **Name** | **Address** | **Width** | **Access** | **Description** |
| --- | --- | --- | --- | --- |
| reg\_working\_result | N/A | 2N-1 | None | Accumulator for results of math operation |
| reg\_multiplier\_temp | N/A | 2N-1 | None | Working copy of multiplier |
| reg\_multiplicand\_temp | N/A | N | None | Working copy of multiplicand |
| reg\_count | N/A | N | None | Operation counter |
| reg\_done | N/A | 1 bit | None | Operation done flag |
| reg\_overflow | N/A | 1 bit | None | Result overflow flag |
| reg\_sign | N/A | 1 bit | None | Sign flag |

Table : List of registers for qmults.v

*Reset Value:*

|  |  |  |
| --- | --- | --- |
| reg\_working\_result | - | 2N-1’b0 |
| reg\_multiplier\_temp | - | 2N-1’b0 |
| reg\_multiplicand\_temp | - | N’b0 |
| reg\_count | - | N’b0 |
| reg\_done | - | 1’b0 |
| reg\_overflow | - | 1’b0 |
| reg\_sign | - | 1’b0 |

#### qdiv.v

| **Name** | **Address** | **Width** | **Access** | **Description** |
| --- | --- | --- | --- | --- |
| reg\_working\_quotient | N/A | 2N-2+Q | None | Accumulator for results of math operation |
| reg\_quotient | N/A | 2N-1 | None | Working copy of multiplier |
| reg\_working\_dividend | N/A | N | None | Working copy of multiplicand |
| reg\_working\_divisor | N/A | 2N-2+Q | None | Working copy of divisor |
| reg\_count | N/A | N | None | Operation counter |
| reg\_done | N/A | 1 bit | None | Operation done flag |
| reg\_overflow | N/A | 1 bit | None | Result overflow flag |
| reg\_sign | N/A | 1 bit | None | Sign flag |

Table : List of registers for qdiv.v

*Reset Value:*

|  |  |  |
| --- | --- | --- |
| reg\_working\_quotient | - | 2N-2+Q’b0 |
| reg\_quotient | - | 2N-1’b0 |
| reg\_working\_dividend | - | N’b0 |
| reg\_working\_divisor | - | 2N-2+Q’b0 |
| reg\_count | - | N’b0 |
| reg\_done | - | 1’b0 |
| reg\_overflow | - | 1’b0 |
| reg\_sign | - | 1’b0 |

Clocks

| **Name** | **Source** | **Rates (MHz)** | | | **Remarks** | **Description** |
| --- | --- | --- | --- | --- | --- | --- |
| **Max** | **Min** | **Resolution** |
| i\_clk | NHA | Solution dependent. Tested to 133MHz on Xilinx Spartan 3E | | | Duty cycle 50/50. | For internal use |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Table : List of clocks

IO Ports

### qadd.v

| **Port** | **Width** | **Direction** | **Description** |
| --- | --- | --- | --- |
| a | N | Input | Addend 1 |
| b | N | Input | Addend 2 |
| c | N | Output | Sum |

Table : List of qadd.v IO ports

### qmult.v

| **Port** | **Width** | **Direction** | **Description** |
| --- | --- | --- | --- |
| i\_multiplicand | N | Input | Multiplicand |
| i\_multiplier | N | Input | Multiplier |
| o\_result | N | Output | Product |
| ovr | 1 | Output | Overflow Flag |

Table : List of qmult.v IO ports

### qmults.v

| **Port** | **Width** | **Direction** | **Description** |
| --- | --- | --- | --- |
| i\_multiplicand | N | Input | Multiplicand |
| i\_multiplier | N | Input | Multiplier |
| i\_start | 1 | Input | Start flag; set this bit high ("1") to start the operation when the last operation is completed. This bit is ignored until o\_complete is asserted. |
| i\_clk | 1 | Input | input clock; internal workings occur on the rising edge |
| o\_result\_out | N | Output | Product |
| o\_complete | 1 | Output | computation complete flag; asserted ("1") when the operation is completed |
| o\_overflow | 1 | Output | overflow flag; asserted ("1") to indicate that an overflow has occurred. |

Table : List of qmults.v IO ports

### qdiv.v

| **Port** | **Width** | **Direction** | **Description** |
| --- | --- | --- | --- |
| i\_dividend | N | Input | Multiplicand |
| i\_divisor | N | Input | Multiplier |
| i\_start | 1 | Input | Start flag; set this bit high ("1") to start the operation when the last operation is completed. This bit is ignored until o\_complete is asserted. |
| i\_clk | 1 | Input | input clock; internal workings occur on the rising edge |
| o\_quotient\_out | N | Output | result |
| o\_complete | 1 | Output | computation complete flag; asserted ("1") when the operation is completed |
| o\_overflow | 1 | Output | overflow flag; asserted ("1") to indicate that an overflow has occurred. |

Table : List of qmults.v IO ports