LogiCORE IP Divider Generator v3.0

DS530 March 1, 2011 **Product Specification**

Introduction

The LogiCORE™ IP Divider Generator core v3.0 creates a circuit for integer division based on Radix-2 non-restoring division, or High-Radix division with prescaling. The Radix-2 algorithm exploits fabric to achieve a range of throughput options that includes single cycle, and the High Radix algorithm exploits XtremeDSP™ slices at lower throughput, but with reuse to reduce resources.

Features

- Drop-in module for Kintex™-7, Virtex®-7, Virtex-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3/XA, Spartan-3E/XA, Spartan-3A/AN/3A DSP/XA FPGAs.
- Integer division with operands of up to 54 bits wide.
- Performs Radix-2 integer division for fabric-only implementation or High Radix division with prescaling to take advantage of XtremeDSP slices.
- Optional operand widths, synchronous controls, and selectable latency.
- For use with Xilinx CORE Generator™ and Xilinx System Generator for DSP v13.1 or later.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	Virtex-7, Kintex-7, Virtex-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3/XA, Spartan-3E/XA, Spartan-3A/3AN/3A DSP/XA
Supported User Interfaces	N/A
Provided with Core	
Documentation	Product Specification
Design Files	Netlist
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	N/A
Simulation Model	Verilog, VHDL
Tested Design Tools	
Design Entry Tools	CORE Generator 13.1, System Generator for DSP 13.1
Simulation	Mentor Graphics ModelSim 6.6d, Cadence Incisive Enterprise Simulator (IES) 10.2, Synopsys VCS and VCS MX 2010.06, ISIM 13.1
Synthesis Tools	Not Provided
Support	
Provided by Xilinx, Inc.	

1. For a complete listing of supported devices, see the release notes for this core.

Overview

Two implementations of division are supported by Divider Generator v3.0:

- **Radix-2.** Radix-2 non-restoring integer division using integer operands, allowing either a fractional or integer remainder to be generated. This is recommended for operand widths less than around 16 bits or for applications requiring high throughput. This is the same algorithm as (and is compatible with) the Serial Divider v3.1 and earlier cores which it supersedes. The implementation uses fabric.
- **High Radix**. High Radix division with prescaling. This is recommended for operand widths greater than around 16 bits. This implementation uses XtremeDSP slices and is not available for devices that do not have XtremeDSP slices.

A detailed explanation of each implementation is provided in Radix-2 Solution and High Radix Solution.

Applications

Division is the most complex of the four basic arithmetic operations. Because hardware solutions are correspondingly larger and more complex than the solutions for other operations, it is best to minimize the number of divisions in any algorithm. There are many forms of division implementation, each of which may offer the optimal solution in different circumstances.

The divider generator core provides two division algorithms, offering solutions targeted at small operands and large operands.

The Radix-2 non-restoring algorithm solves one bit of the quotient per cycle using addition and subtraction. The design is fully pipelined, and can achieve a throughput of one division per clock cycle. If the throughput required is smaller, the divisions per clock parameter allows compromises of throughput and resource use. This algorithm naturally generates a remainder, so is the choice for applications requiring integer remainders or modulus results.

The High Radix with prescaling algorithm resolves multiple bits of the quotient at a time. It is implemented by reusing the quotient estimation block, and so throughput is a function of the number of iterations required. The operands must be conditioned in preparation for the iterative operation. This overhead makes this algorithm less suitable for smaller operands. Although the iterative calculation is more complex than for Radix-2, taking more cycles to perform, the number of bits of quotient resolved per iteration and its use of XtremeDSP slices makes this the preferred option for larger operand widths.

Radix-2 Solution

Radix-2 Feature Summary

- Provides quotient with integer or fractional remainder
- Pipelined, parallel architecture for increased throughput
- Pipeline reduction for size versus throughput selections
- Dividend width from 2 to 32 bits
- Divisor width from 2 to 32 bits
- Independent dividend, divisor and fractional bit widths
- Fully synchronous design using a single clock
- Supports unsigned or two's complement signed numbers
- Can implement 1/X (reciprocal) function
- Fully registered outputs
- Compatible with Serial Divider v3.1, which it supersedes

Radix-2 Solution Overview

This parameterized solution divides an M-bit-wide variable dividend by an N-bit-wide variable divisor. The output consists of the quotient and either an integer remainder or fractional result (quotient continued past the binary point). In the integer remainder case, the result of the division is an M-bit-wide quotient with an N-bit-wide integer remainder (Equation 1). In the fractional case, the result is an M-bit-wide quotient with an F-bit-wide fractional remainder (Equation 2). When signed operation is selected, all operands and results employ a two's complement sign bit, resulting in one less bit of magnitude result (Equation 3).

Integer remainder case:

$$Dividend = Quotient * Divisor + IntRmd$$

Equation 1

F-bit-wide fractional remainder in the unsigned case:

$$FractRmd = \frac{IntRmd * 2^{F}}{Divisor}$$

Equation 2

F-bit-wide fractional remainder in the signed case:

$$FractRmd = \frac{IntRmd * 2^{(F-1)}}{Divisor}$$

Equation 3

For signed mode with integer remainder, the sign of the quotient and remainder correspond exactly to Equation 1.

Thus,

6/-4 = -1 REMD 2

whereas

-6/4 = -1 REMD –2

For signed mode with fractional remainder, the sign bit is present both in the quotient and the fractional remainder. For example, for a five-bit dividend, divisor and fractional remainder we have:

-9/4 = 9/-4 = -(2 1/4)

This corresponds to:

10111/00100 or 01001/11100

Giving the result:

Quotient = 11110 (= -2)

Remainder = 11100 (= -1/4)

Note that for division by zero, the quotient, remainder, and fractional results are undefined.

The core is highly pipelined. The throughput of the core is configurable and can be reduced from 1 clock cycle per division to 2, 4 or 8 clock cycles per division to reduce resources.

The dividend and divisor bit widths can be set independently. The bit width of the quotient is equal to the bit width of the dividend. The bit width of the integer remainder is equal to the width of the divisor. For fractional output, the remainder bit width is independent of the dividend and divisor. The core handles data ranges of 2 to 32 bits for dividend, divisor, and fractional outputs.

The divider can be used to implement the reciprocal of X; that is the 1/X function. To do this, the dividend bit width is set to 2 and fractional mode is selected. The dividend input is then tied to 01 for both unsigned or signed operation, and the X value is provided via the divisor input.

Radix-2 Solution Pinout

The fixed-point core pinout and signal names are shown in Figure 1 and defined in Table 1.

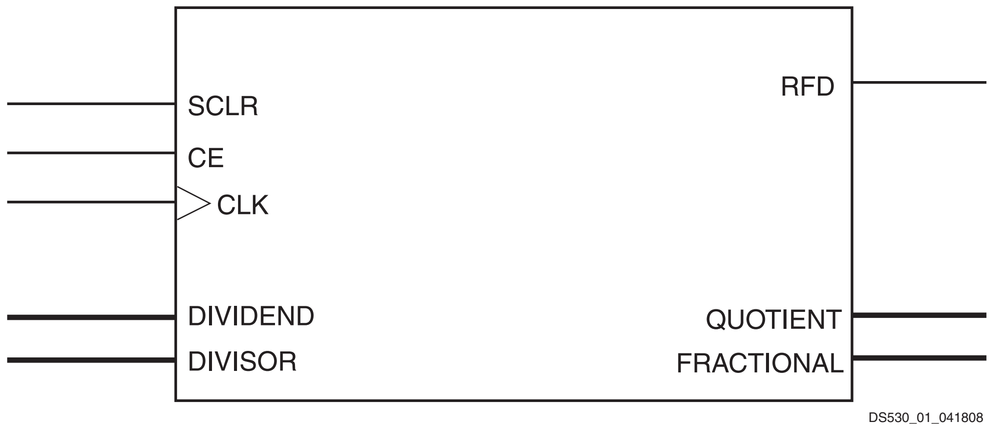

Figure 1: **Core Pinout Diagram**

Table 1: **Radix-2 Signal Pinout**

Signal	Direction[1]	Description
DIVIDEND[M-1:0]	Input	**Dividend** (parallel data in). Data bit width specified by Dividend and Quotient Width in GUI[2].
DIVISOR[N-1:0]	Input	**Divisor** (parallel data in). Data bit width specified by the Divisor Width in GUI.
CLK	Input	**Clock**. Control and data inputs are captured and new output data formed on rising-edge.
SCLR	Input (optional)	**Synchronous Clear (SCLR)**. Optional input pin. When asserted (High), all the core flip-flops are synchronously initialized (synchronous to the clock). The core remains in this state until SCLR is deasserted. When both SCLR and CE exist, the SCLR/CE Priority settings in the GUI determine whether SCLR is qualified by CE or whether SCLR overrides CE (that is, SCLR will clear the module on the clock edge even if CE is deasserted).
CE	Input (optional)	**Clock Enable (CE)**. Optional input pin. When deasserted (Low), all the synchronous inputs are ignored and the core remains in its current state.
RFD	Output	**Ready for Data (RFD)**. An output that indicates the cycle in which input data is sampled by the core. This is only applicable to cores where Clocks per Division is not 1. For the case of Clocks per Division is 1, the core samples the inputs on every enabled clock rising edge; hence, RFD will always be High. When Clocks per Division is either 2, 4, or 8, the core only samples data on every 2nd, 4th, or 8th enabled clock rising edge, respectively. The cycle on which data is sampled is indicated by RFD. This is important for the definition of latency, as shown in Figure 3. RFD only changes on enabled (CE input) clock rising edges for a core that has CE input.
QUOTIENT[M-1:0]	Output	**Quotient**. The result of the integer division of dividend by divisor (that is, dividend DIV divisor). The bit width of the quotient is equal to the dividend. For signed operation, the quotient is in two's complement form. Data bit width is determined by the Dividend and Quotient Width GUI parameter.
FRACTIONAL[N-1:0] or FRACTIONAL[F-1:0]	Output	**Fractional**. When Remainder Type is set to Remainder in the GUI, this output provides the remainder of the division of dividend by divisor (that is, dividend MOD divisor). When Remainder Type is set to Fractional, this output is the fractional part of the division result. For either case, if the core is signed, the output is in two's complement form. • **Integer Remainder.** Result data bit width determined by Divisor Width. • **Fractional Remainder.** Result data bit width determined by Fractional Width.

Notes:

1. All control inputs are Active High. If an Active Low input is required for a particular control pin, an inverter must be placed in the path to the pin. The inverter is absorbed appropriately during synthesis and/or mapping.
2. Dividend and Quotient Width must be set to satisfy the largest possible quotient result. Note that due to the non-symmetry of two's complement representation bit growth from the dividend to quotient is possible, but only for the single combination of the most negative number divided by negative one (i.e., $-2^{(M-1)}/-1$). The width of dividend and quotient can be extended by 1 bit should this situation need to be accommodated.

Following a power-on reset or SCLR, the core outputs zeros on QUOTIENT and FRACTIONAL outputs until new results appear.

Radix-2 Solution Waveforms

The total latency (number of clocks required before the core generates the first valid output) is a function of the bit width of the dividend. If fractional output is required, the latency is also a function of the fractional bit width. If clock enable is selected, latency is in terms of enabled clock cycles.

When Clocks per Division is set to 2, 4, or 8, the RFD (Ready For Data) output indicates the cycle in which input data is sampled (Figure 2) and, therefore, from when latency is measured. RFD should be qualified by clock enable if used externally.

In general:

- Latency is of the order M for integer remainder dividers
- Latency is of the order M + F for fractional remainder dividers

Table 2 provides a list of the latency formula for divider selections, and Figure 2 illustrates how latency is defined. Latency is expressed in clock cycles for dividers with no clock enable input and otherwise in enabled clock cycles.

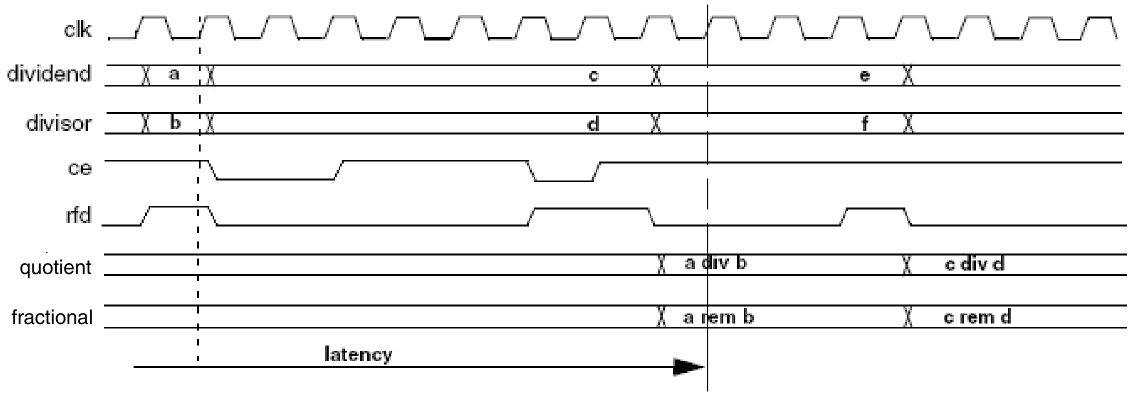

Figure 2: **Latency Example (Clocks per Division = 4)**

Table 2: **Latency of Radix-2 Solution Based on Divider Parameters**

Signed	Fractional	Clocks Per Division	Latency[(1)]
False	False	1	M+2
False	False	>1	M+3
False	True	1	M+F+2
False	True	>1	M+F+3
True	False	1	M+4
True	False	>1	M+5
True	True	1	M+F+4
True	True	>1	M+F+5

Notes:

1. M = Dividend and Quotient Width, F = Fractional Width.

The Clocks per Division parameter allows a range of choices of throughput versus resources. With Clocks per Division set to 1, the core is fully pipelined, so it will have maximal throughput of one division per clock cycle, but will use the most resources. Clock per Division settings of 2, 4, and 8 reduce the throughput by those respective factors for smaller core sizes.

Performance Characteristics of Radix-2 Solution

Table 3 defines performance characteristics for Radix-2 cases on Virtex-6 FPGA, speed grade 1. The tool arguments used were as follows:

```
map –ol high
par -ol high
```

The performance characteristics are intended to provide an indication of resources used and achievable clock speeds. For all cases the Operand Sign is unsigned and CE and SCLR are disabled.

Table 3: **Radix-2 Solution Performance Characteristics on Virtex-6 FPGA (Part = XC6VLX75T-1)**

Parameter/Result	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Dividend and Quotient Width	8	8	8	8	16	16	32	32
Divisor Width	8	8	8	8	8	16	32	32
Remainder Type	Rem	Rem	Rem	Rem	Rem	Rem	Rem	Frac
Fractional Width	8	8	8	8	8	16	32	32
Clocks per Division	1	2	4	8	1	1	1	1
LUT6-FF Pairs	159	134	85	66	372	578	2,176	4,487
LUTs	153	130	77	62	356	565	2,144	4,444
FFs	224	163	114	90	560	832	3,200	6,736
Block RAMs	0	0	0	0	0	0	0	0
DSP48 Blocks	0	0	0	0	0	0	0	0
Max Clock Freq(1)(2)	450	406	432	442	450	450	360	343

Notes:

1. Area and maximum clock frequencies are provided as a guide and might vary with new releases of the Xilinx implementation tools.
2. Maximum clock frequencies are shown in MHz. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

Table 4 defines performance characteristics for Radix-2 cases on Virtex-5 FPGA, speed grade 1.

Table 4: **Radix-2 Solution Performance Characteristics on Virtex-5 FPGA (Part = XC5VSX35T-1)**

Parameter/Result	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Dividend and Quotient Width	8	8	8	8	16	16	32	32
Divisor Width	8	8	8	8	8	16	32	32
Remainder Type	Rem	Rem	Rem	Rem	Rem	Rem	Rem	Frac
Fractional Width	8	8	8	8	8	16	32	32
Clocks per Division	1	2	4	8	1	1	1	1
LUT6-FF Pairs	229	181	114	91	573	840	3,210	6,752
LUTs	95	88	50	32	183	319	1,151	2,239
FFs	224	163	114	90	560	832	3,200	6,736
Block RAMs	0	0	0	0	0	0	0	0
DSP48 Blocks	0	0	0	0	0	0	0	0
Max Clock Freq(1)(2)	439	373	389	388	450	375	330	298

Notes:

1. Area and maximum clock frequencies are provided as a guide and might vary with new releases of the Xilinx implementation tools.
2. Maximum clock frequencies are shown in MHz. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

Table 5 defines performance characteristics for Radix-2 cases on Spartan-6 FPGA, speed grade 2.

Table 5: **Radix-2 Solution Performance Characteristics on Spartan-6 FPGA (Part = XC6SLX16-2)**

Parameter/Result	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Dividend and Quotient Width	8	8	8	8	16	16	32	32
Divisor Width	8	8	8	8	8	16	32	32
Remainder Type	Rem	Rem	Rem	Rem	Rem	Rem	Rem	Frac
Fractional Width	8	8	8	8	8	16	32	32
Clocks per Division	1	2	4	8	1	1	1	1
LUT6-FF Pairs	161	124	81	61	368	573	2,182	4,486
LUTs	156	118	78	56	354	561	2,157	4,450
FFs	224	163	114	90	560	832	3,200	6,736
Block RAMs	0	0	0	0	0	0	0	0
DSP48 Blocks	0	0	0	0	0	0	0	0
Max Clock Freq(1)(2)	237	222	228	234	236	209	166	163

Notes:

1. Area and maximum clock frequencies are provided as a guide and might vary with new releases of the Xilinx implementation tools.
2. Maximum clock frequencies are shown in MHz. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

Table 6 defines performance characteristics for Radix-2 cases on a Spartan-3A DSP FPGA, speed grade 4.

Table 6: **Radix-2 Solution Performance Characteristics on Spartan-3A DSP FPGA (Part = XC3SD3400A-4)**

Parameter/Result	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Dividend and Quotient Width	8	8	8	8	16	16	32	32
Divisor Width	8	8	8	8	8	16	32	32
Remainder Type	Rem	Rem	Rem	Rem	Rem	Rem	Rem	Frac
Fractional Width	8	8	8	8	8	16	32	32
Clocks per Division	1	2	4	8	1	1	1	1
Slices	124	103	69	59	304	440	1,650	3,474
LUTs	79	75	41	24	151	287	1,087	2,111
FFs	224	163	114	90	560	832	3,202	6,738
Block RAMs	0	0	0	0	0	0	0	0
DSP48 Blocks	0	0	0	0	0	0	0	0
Max Clock Freq(1)(2)	229	156	168	225	239	201	147	156

Notes:

1. Area and maximum clock frequencies are provided as a guide and might vary with new releases of the Xilinx implementation tools.
2. Maximum clock frequencies are shown in MHz. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

High Radix Solution

High Radix Solution Feature Summary

- High Radix division enabled by prescaling
- Provides quotient and, optionally, fractional outputs
- Configurable widths, synchronous controls, selectable latency and detection of division by zero
- Uses XtremeDSP slices

High Radix Solution Overview

The High Radix implementation performs division by prescaling operands before employing an accelerated High Radix division algorithm. The design is fully pipelined for maximum clock frequency. First, the divisor is normalized, then an estimate of its reciprocal is made. Both operands are multiplied by this estimate to bring the divisor closer to 1. The precision and accuracy of the prescale determines how many bits of quotient can be resolved on each subsequent iteration. The fact that the prescaled divisor is close to one allows the estimate of new quotient bits to be just the top bits of the residue left from the previous iteration. The iterative operation itself is performed in carry-save notation, so that no long carry chains limit performance. Because only the top bits of the residue are used as the estimate and the divisor is not exactly 1, errors do occur in the internal result of each iteration; thus, the quotient bits resolved on each iteration overlap slightly with the previously resolved bits to allow correction of errors in subsequent iterations.

Because the iteration calculation consists of a carry-save multiplication and subtraction, it is ideally suited to the XtremeDSP (multiply-add) slices, providing an efficient, low-latency iteration.

Throughput Considerations

The iterative process is implemented as a loop rather than an unrolled data pipeline to reduce resources. This means that new input must be held off until previous calculations are finished within the iterative circuit. The maximum possible throughput is therefore 1/N divisions per clock, where N is the number of iterations required. However, to achieve this maximum throughput the input may be required to be bursty. This is because the iterative engine may be pipelined with each stage of the pipe offering a carousel place for interlaced divisions.

To achieve a constant number of clocks between inputs, the number of cycles between new input must be mutually prime with the number of pipe stages in the iterative engine. To support the maximum throughput, New Data (ND), Ready For Data (RFD), and output Ready (RDY) pins are provided so that the input may come from a FIFO for instance. To support a constant interval between inputs, the GUI provides feedback of the rate (1 in N) at which the divider can accept input on a continuous basis with a constant interval. This is expressed on the "Throughput" field of the GUI and is expressed in terms of 1 input every N enabled clock cycles.

The three handshaking signals provided are ND, RFD, and RDY. All these signals are subject to CE, so ND must be true when CE is true to be input to the core. Likewise, RFD and RDY will only change on a clock cycle where CE is true. ND is a request for new data input, but will only be accepted if RFD is true, else the input will be ignored. If ND and RFD are both true, then the core will accept the input and a result will appear with RDY after some delay. This latency is specified through the GUI and is represented in terms of the number of clock enabled clock cycles from input to output.

High Radix Solution Pinout

The High Radix core pinout and signal names are displayed in Figure 3 and defined in Table 7.

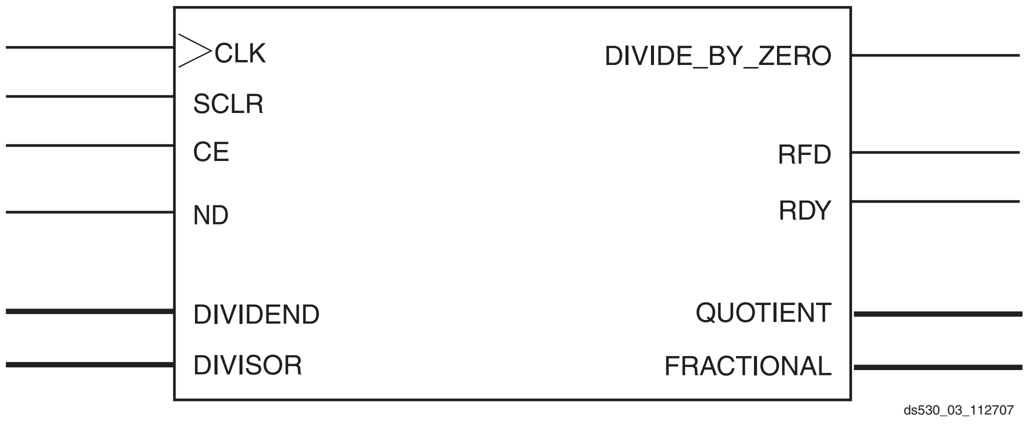

Figure 3: **High Radix Schematic Symbol**

Table 7: **High Radix Solution Pinout**

Signal	Direction[1]	Description
DIVIDEND[M-1: 0]	Input	**Signed integer dividend**[2]. Bit width specified by Dividend and Quotient Width in GUI.
DIVISOR[M-1:0]	Input	**Signed integer divisor.** Bit width specified by Divisor Width in GUI.
CLK	Input	**Clock**. Rising edge clock signal.
CE	Input (optional)	Clock Enable (CE).
SCLR	Input (optional)	Synchronous Clear (SCLR).
QUOTIENT[M-1:0]	Output	**Signed integer part of quotient.** Bit width specified by Dividend and Quotient Width in GUI.
FRACTIONAL[F-1:0]	Output	**Unsigned fractional part of quotient.** Does not contain a two's complement sign bit. Bit width specified by Fractional Width in GUI.
DIVIDE_BY_ZERO	Output (optional)	Detection of division by zero.
ND	Input	**New Data (ND)**. Used to signal to the core that new operands are present on the input to the core.
RFD	Output	**Ready for Data (RFD)**. Signals that the core is able to take new operands.
RDY	Output	**Ready**. Signals that a result is available at the output of the core.

Notes:

1. All control inputs are Active High. If an Active Low input is required for a particular control pin, an inverter must be placed in the path to the pin. The inverter is absorbed appropriately during synthesis and/or mapping.
2. Dividend and Quotient Width must be set to satisfy the largest possible quotient result. Note that due to the non-symmetry of two's complement representation bit growth from the dividend to quotient is possible, but only for the single combination of the most negative number divided by negative one (that is, $-2^{(M-1)}/-1$). The width of dividend and quotient can be extended by 1 bit should this situation need to be accommodated.

High Radix Solution Waveforms

The functional timing characteristics of the High Radix solution are illustrated in Figure 4. The DIVIDE_BY_ZERO output, when enabled, is provided on the same cycle as the QUOTIENT and FRACTIONAL outputs.

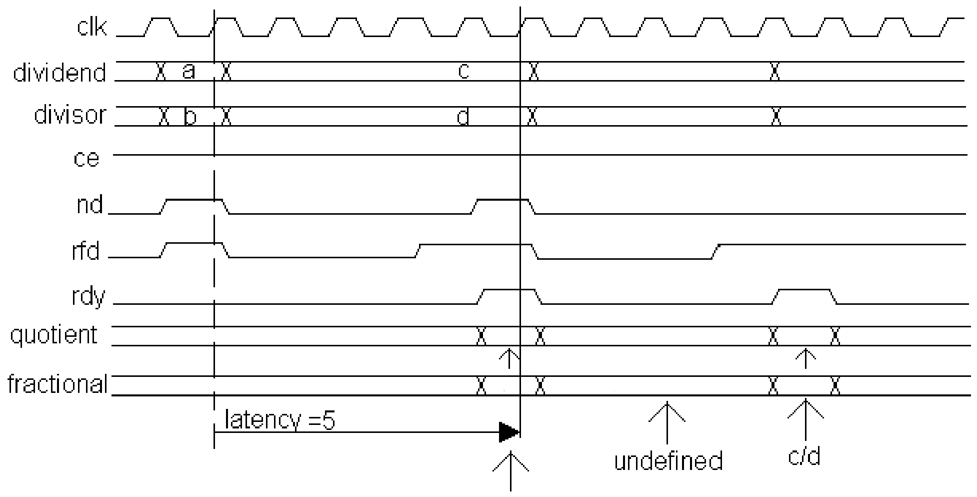

Figure 4: **High Radix Timing Diagram**

The minimum and maximum latency of the High Radix solution are summarized in Table 8 and Table 9 respectively.

Table 8: **Minimum Latency of High-Radix Solution Based on Divider Parameters**

Dividend and Quotient Width + Fractional Width					
4 to 12	**13 to 26**	**27 to 40**	**41 to 54**	**55 to 68**	**69 to 82[1]**
2	3	4	5	6	7

Notes:

1. Not applicable on Virtex-4 FPGA.

Table 9: **Maximum Latency of High-Radix Solution Based on Divider Parameters**

Divisor Width	Dividend and Quotient Width + Fractional Width					
	4 to 12	**13 to 26**	**27 to 40**	**41 to 54**	**55 to 68**	**69 to 82[1]**
4 to 8	16/16/16[2]	20/20/20	24/24/24	29/30/29	33/34/33	37/NA/37
9 to 18	17/17/17	21/21/21	25/25/25	30/31/30	34/35/34	38/NA/38
19 to 32	18/18/19	22/22/23	26/26/27	31/32/32	35/36/36	39/NA/40
33 to 35	19/19/20	23/23/24	27/27/28	32/33/33	36/37/37	40/NA/41
36 to 48	20/20/22	24/24/26	28/28/30	33/34/35	37/38/39	41/NA/43
49 to 52	22/22/24	26/26/28	30/30/32	35/36/37	39/40/41	43/NA/45
53 to 54	23/23/26	27/27/30	31/31/34	36/37/39	40/41/43	44/NA/47

Notes:

1. Not applicable on Virtex-4 FPGA.
2. Latency values are specific to family groups and are presented in the following way: Virtex-7, Kintex-7, Virtex-6, Virtex-5, Virtex-4, Spartan-3A DSP, and Spartan-6.

Performance Characteristics of High Radix Solution

Table 10 defines performance characteristics for High Radix cases on a Virtex-6 FPGA, speed grade 1. They are intended to provide an indication of resources used and achievable clock speeds. The tool arguments used were as follows:

```
map –ol high
par -ol high
```

Optional control signals CE and SCLR are disabled. The High Radix treats inputs and outputs as signed numbers.

Note: When the core does not have synchronous clear, use can be made of SRL16 primitives, leading to a substantial reduction in circuit size. For this reason, the use of SCLR is not recommended.

Table 10: **High Radix Solution Performance Characteristics on Virtex-6 FPGA (Part = XC6VLX75T-1)**

Parameter/Result	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
Dividend and Quotient Width	10	10	36	36	54	54	37
Divisor Width	14	14	36	36	50	50	24
Remainder Type	Frac	Frac	Frac	Frac	Frac	Frac	Frac
Fractional Width	2	2	2	2	28	28	0
Latency Configuration (latency)	Auto (17)	2	Auto (28)	4	Auto (43)	8	Auto (26)
LUT-FF Pairs	365	265	974	689	1,429	1,018	729
LUTs	358	262	963	685	1,405	1,011	720
FFs	412	70	1,105	187	1,652	266	818
RAMB18E1	1	1	1	1	1	1	1
DSP48E1s	7	7	13	13	17	17	10
Max Clock Freq(1)(2)	450	82	450	65	413	56	450

Notes:

1. Resources and maximum clock frequencies are provided as a guide and might vary with new releases of the Xilinx implementation tools.
2. Maximum clock frequencies are shown in MHz. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

Table 11 defines performance characteristics for cases run on a Virtex-5 FPGA, speed grade 1.

Table 11: **High Radix Solution Performance Characteristics on Virtex-5 FPGA (Part = XC5VSX35T-1)**

Parameter/Result	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
Dividend and Quotient Width	10	10	36	36	54	54	37
Divisor Width	14	14	36	36	50	50	24
Remainder Type	Frac	Frac	Frac	Frac	Frac	Frac	Frac
Fractional Width	2	2	2	2	28	28	0
Latency Configuration (latency)	Auto (17)	2	Auto (28)	4	Auto (43)	8	Auto (26)
LUT6-FF Pairs	424	279	1,140	732	1,693	1,091	842
LUTs	355	267	972	693	1,398	1,009	715
FFs	412	70	1,104	187	1,649	266	818
BlockRAM 18k	1	1	1	1	1	1	1
DSP48Es	7	7	13	13	17	17	10
Max Clock Freq(1)(2)	450	75	374	62	337	47	397

Notes:

1. Resources and maximum clock frequencies are provided as a guide and might vary with new releases of the Xilinx implementation tools.
2. Maximum clock frequencies are shown in MHz. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

Table 12 defines performance characteristics for cases run on a Spartan-6 FPGA.

Table 12: **High Radix Solution Performance Characteristics on Spartan-6 FPGA (Part = XC6SLX16-2)**

Parameter/Result	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
Dividend and Quotient Width	10	10	36	36	54	54	37
Divisor Width	14	14	36	36	50	50	24
Remainder Type	Frac	Frac	Frac	Frac	Frac	Frac	Frac
Fractional Width	2	2	2	2	28	28	0
Latency Configuration (latency)	Auto (17)	2	Auto (30)	4	Auto (45)	8	Auto (27)
LUT6-FF Pairs	271	232	756	603	1,113	882	550
LUTs	257	229	744	601	1,088	869	540
FFs	412	70	1,121	189	1,682	268	824
RAMB16BWERs	1	1	1	1	1	1	1
DSP48A1s	7	7	13	13	17	17	10
Max Clock Freq(1)(2)	241	44	228	29	217	27	241

Notes:

1. Resources and maximum clock frequencies are provided as a guide and might vary with new releases of the Xilinx implementation tools.
2. Maximum clock frequencies are shown in MHz. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

Table 13 defines performance characteristics for cases run on a Spartan-3A DSP FPGA.

Table 13: **High Radix Solution Performance Characteristics on Spartan-3A DSP FPGA (Part = XC3SD3400A-4)**

Parameter/Result	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
Dividend and Quotient Width	10	10	36	36	54	54	37
Divisor Width	14	14	36	36	50	50	24
Remainder Type	Frac	Frac	Frac	Frac	Frac	Frac	Frac
Fractional Width	2	2	2	2	28	28	0
Latency Configuration (latency)	Auto (17)	2	Auto (30)	4	Auto (45)	8	Auto (27)
Slices	265	198	715	551	1,065	846	522
LUTs	417	297	1,211	885	1,722	1,328	908
FFs	431	76	1,140	193	1,699	267	844
RAMB16BWERs	1	1	1	1	1	1	1
DSP48As	7	7	13	13	17	17	10
Max Clock Freq(1)(2)	247	43	178	36	151	31	205

Notes:

1. Resources and maximum clock frequencies are provided as a guide and might vary with new releases of the Xilinx implementation tools.
2. Maximum clock frequencies are shown in MHz. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

CORE Generator Graphical User Interface

The Divider Generator core GUI provides one page split into sections to set parameter values for the particular instantiation required. This section provides a description of each GUI field. These fields are grouped as follows:

- **Common Options**: Describes parameters common to both implementations and allows the selection of the divider implementation.
- **Radix-2 Options**: Provides configuration options for the Radix-2 divider configuration.
- **High Radix Options**: Provides configuration options for the High Radix divider configuration.
- **Control Signals**: Provides configuration of SCLR and CE control inputs.

Common Options

- **Component Name**: The base name of the output files generated for the core. Names must begin with a letter and be composed of any of the following characters: a to z, 0 to 9 and "_".
- **Algorithm Type**: This selects between Radix-2 and High Radix division solutions.
- **Dividend and Quotient Width**: Specifies the number of integer bits provided on the DIVIDEND and QUOTIENT ports. This must be set to satisfy the largest possible quotient result. Note that due to the non-symmetry of two's complement representation bit growth from the dividend to quotient is possible, but only for the single combination of the most negative number divided by negative one (that is, $-2^{(M-1)}/-1$). The width of dividend and quotient can be extended by 1 bit should this situation need to be accommodated.
- **Divisor Width**: Specifies the number of integer bits provided on the DIVISOR port.

- **Remainder Type**: This selects between remainder types Fractional and Remainder presented on the FRACTIONAL port. Fractional remainder type is the only option for High Radix.
- **Fractional Width**: If Fractional remainder type is selected, this determines the number of bits provided on the FRACTIONAL port output. Note that when High Radix, the sum of Fractional Width and Dividend and Quotient Width is limited to 68 on Virtex-4 FPGA and 82 on other supported FPGA families.
- **Operand Sign**: Signed or unsigned. Determines the interpretation of the input and output buses. Operand sign for the High Radix solution is always signed, and for this solution the FRACTIONAL output is always unsigned.
- **Latency Configuration**: Automatic (fully pipelined) or manual (determined by following field). Latency Configuration for Radix-2 solution is always Automatic.
- **Latency**: When Latency Configuration is set to Automatic, this field provides the latency from input to output in terms of clock enabled clock cycles. When Manual, this field is used to specify the latency required.

Radix-2 Options

- **Clocks Per Division**: Determines the interval in clocks between new data being input (and output).

High Radix Options

- **Detect Divide-by-Zero**: Check box. Determines if the core has a DIVIDE_BY_ZERO output port to signal when a division by zero has been performed.
- **Number of iterations:** Read-only text field that reports the number of iterations performed by the High-Radix engine for each divide. This sets the maximum throughput of the divider. To achieve this throughput, the operands must be supplied as soon as requested by the core RFD output.
- **Throughput:** Read-only text field that reports the maximum throughput that can be sustained by the divider when operands are supplied at a constant rate. This rate is controlled by ND and is not governed by RFD (that is, RFD will be high when ND is applied at this rate).

Control Signals

- **CE**: Check box. Determines if the core has a clock enable input.
- **SCLR**: Determines if the core has a synchronous clear input.
- **SCLR/CE Priority**: Determines which of SCLR or CE has priority over the other.

XCO Parameters

Table 14 defines the mapping between GUI parameters and XCO parameters.

Table 14: **GUI and XCO Parameter Mapping**

GUI Parameter	Default Value	XCO Values	XCO Parameter
Common Parameters			
Algorithm Type	Radix2	Radix2, High_Radix	algorithm_type
Dividend and Quotient Width[1][2]	16	2 to 32 for Radix2, 4 to 54 for High_Radix	dividend_and_quotient width
Divisor Width	16	2 to 32 for Radix2, 4 to 54 for High_Radix	divisor_width
Remainder Type	Remainder	Remainder, Fractional	remainder _type
Operand Sign	Signed	Unsigned, Signed Must be Signed for High_Radix	operand_sign
Fractional Width[2]	16	2 to 32 for Radix2 0 to 54 for High Radix	fractional_width
Latency Configuration	Automatic	Automatic, Manual Must be Automatic for Radix2	latency_configuration
Latency	18	Range is a function of other parameters as summarized in Table 2, Table 8 and Table 9.	latency
Radix-2 Divider Parameter			
Clocks per Division	1	1, 2, 4, 8	clocks_per_division
High Radix Divider Parameter			
Detect Divide-By-Zero	false	false, true	divide_by_zero_detect
Control Signal Parameters			
CE	false	false, true	ce
SCLR	false	false, true	sclr
SCLR/CE Priority	SCLR_overrides_CE	SCLR_overrides_CE, CE_overrides_SCLR	sclr_ce_priority

Notes:

1. Dividend and Quotient Width must be set to satisfy the largest possible quotient result. Note that due to the non-symmetry of two's complement representation bit growth from the dividend to quotient is possible, but only for the single combination of the most negative number divided by negative one (that is, $-2^{(M-1)}/-1$). The width of dividend and quotient can be extended by 1 bit should this situation need to be accommodated.
2. When High Radix the sum of Fractional Width and Dividend and Quotient Width is limited to 68 on Virtex-4 FPGA and 82 on other supported FPGA families.

Using the Divider Generator IP Core

The CORE Generator GUI performs error-checking on all input parameters. Optimum latency information is also available.

Several files are produced when a core is generated, and customized instantiation templates for Verilog and VHDL design flows are provided in the `.veo` and `.vho` files, respectively. For detailed instructions, see the CORE Generator software documentation.

Simulation Models

The core has a number of options for simulation models:

- VHDL UniSim-based structural simulation model
- Verilog UniSim-based structural simulation model

The models required may be selected in the CORE Generator project options.

Xilinx recommends that simulations utilizing UniSim-based structural models are run using a resolution of 1 ps. Some Xilinx library components require a 1 ps resolution to work properly in either functional or timing simulation. The UniSim-based structural simulation models may produce incorrect results if simulated with a resolution other than 1 ps. See the "Register Transfer Level (RTL) Simulation Using Xilinx Libraries" section in *Chapter 6* of the Synthesis and Simulation Design Guide. This document is part of the ISE® Software Manuals set available at www.xilinx.com/support/software_manuals.htm.

Migrating to Divider Generator v3.0 from v2.0

The CORE Generator core update feature may be used to update an existing Divider XCO file from version 2.0 to version 3.0 of the Divider Generator core. The core may then be regenerated to create a new netlist. See the CORE Generator documentation for more information on this feature.

Port Changes

There are no differences in port naming conventions, polarities, priorities or widths between versions.

Latency Changes

There are no latency differences between versions. If Latency Configuration is Manual, the latency used for v2.0 will be re-used when regenerating for v3.0.

System Generator For DSP Graphical User Interface

This section describes the System Generator for DSP GUI and details the parameters that differ from the CORE Generator GUI.

The Divider Generator core may be found in the Xilinx Blockset in the Math section. The block is called "Divider Generator 3.0."

See the System Generator for DSP Help page for the "Divider Generator 3.0" block for more information on parameters not mentioned here.

The System Generator for DSP GUI offers the same parameters as the CORE Generator GUI, with the exception that the Operand Sign is inferred from the input operands.

References

1. *"Computer Arithmetic Algorithms and Hardware Designs,"* Behrooz Parhami. Oxford Press © 2000.
2. *"Proceedings 12th Symposium on Computer Arithmetic,"* IEEE Computer Society Press © 1995.

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

See the IP Release Notes Guide (XTP025) for further information on this core.

For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

Ordering Information

This core may be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator software v13.1 and later. The Xilinx CORE Generator system is shipped with Xilinx ISE Design Suite development software.

To order Xilinx software, contact your local Xilinx sales representative.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
01/18/06	1.0	Initial Xilinx release.
06/27/08	2.0	Added support for Virtex-5 and Spartan-3A DSP FPGAs, and replaced division by repeated multiplications with High Radix division.
06/24/09	3.0	Added support for Virtex-6 and Spartan-6.
03/01/11	4.0	Added support for Virtex-7 and Kintex-7.

Notice of Disclaimer