FPGA Resource Utilization Estimates for NI cRIO-9074

LabVIEW FPGA Version: 8.6
NI-RIO Version: 3.0
Date: 8/5/2008

Note: The numbers presented in this document are estimates. Actual resource usage for your designs may be different (smaller or greater) than the resource usage computed using the information contained in this document.

1. How to use the table

The tables in this spreadsheet contain approximations of resource usage.

To obtain an estimate for your design, add the corresponding numbers for each function/VI you use. Some of the functions on the LabVIEW FPGA palette consume no logic resources on the FPGA because they are purely wiring operations:

Insert Into Array			0	0	0	0
Delete From Array			0	0	0	0
Initialize Array			0	0	0	0
Build Array			0	0	0	0

For more FPGA-specific information about the objects on the palette see LabVIEW Help:
VI and Function Reference > FPGA Module VIs and Functions > FPGA VI and Function details.

To allow for target overhead, you should add the following base numbers to the sum of flip flops and the sum of LUTs:

Flip Flops	LUTs
388	440

Also add the corresponding numbers for the controls and indicators in your VI(s):

Data type	Resource usage	
	Flip Flops	LUTs
Numeric controls and indicators		
U8	14	13
U16	27	21
U32	52	39
U64	102	90
I8	14	13
I16	27	21
I32	52	39
I64	102	90
FXP<±,8,4>	14	13
FXP<±,16,8>	27	21
FXP<±,32,16>	52	39
FXP<±,64,32>	102	90
Boolean controls and indicators		
Boolean	2	5

Data type	Fixed array size	Resource usage	
		Flip Flops	LUTs
Array controls and indicators			
U8	5	70	69
	10	132	116
	15	194	174
U16	5	132	116
	10	256	229
	15	382	341
U32	5	256	229
	10	505	444
	15	755	672
U64	5	506	415
	10	1004	853
	15	1509	1298
I8	5	70	68
	10	132	116
	15	194	174
I16	5	132	116
	10	256	229
	15	382	341
I32	5	256	229
	10	506	415
	15	755	672
I64	5	505	428
	10	1004	853
	15	1509	1298
FXP<±,8,4>	5	70	68
	10	132	116
	15	194	174
FXP<±,16,8>	5	132	116
	10	256	229
	15	382	341
FXP<±,32,16>	5	256	229
	10	506	415
	15	755	672
FXP<±,64,32>	5	506	415
	10	1004	853
	15	1509	1298
Boolean	5	10	11
	10	19	16
	15	28	22

2. An example

To illustrate the difference between actual resource usage and the estimate obtained using the spreadsheet, we use an example from the NI Example Finder, *Counters – R Series.lvproj*
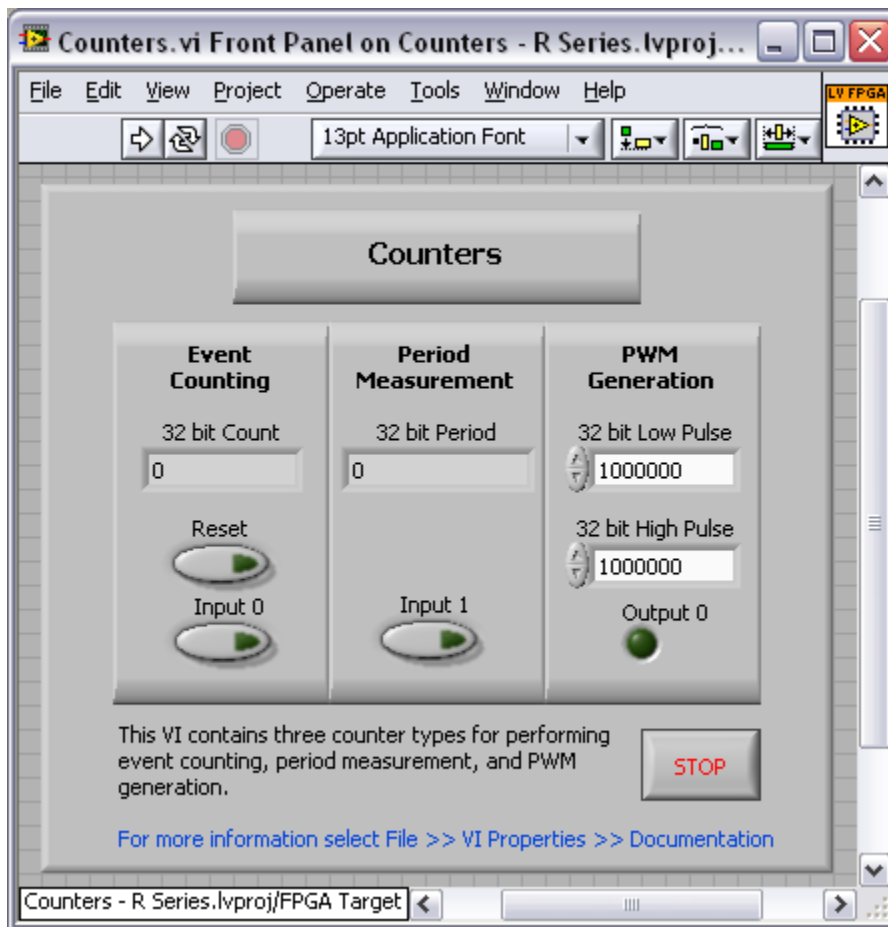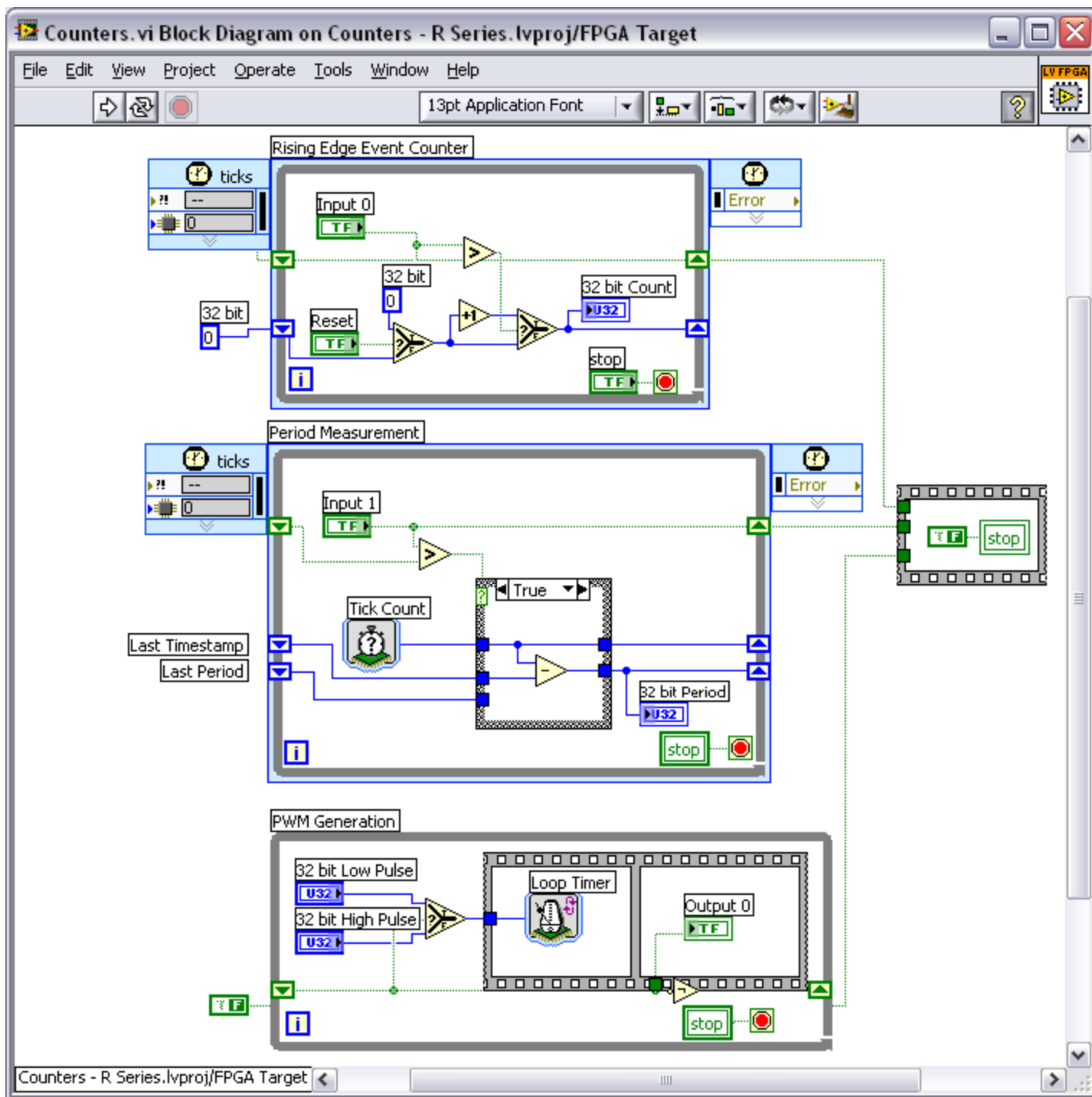
Actual resource usage:

Logic Utilization:

Number of Slice Flip Flops:	936 out of 40,960	2%
Number of 4 input LUTs:	1,027 out of 40,960	2%
Number used as logic:	943	
Number used as a route-thru:	80	
Number used as Shift registers:	4	

Logic Distribution:

Number of occupied Slices:	767 out of 20,480	3%

Estimated resource usage:

Name of VI/Element	Resource Usage	
	Flip Flops	LUTs
Functions		
Select I32/U32 SCTL 1	0	32
Select I32/U32 SCTL 2	0	32
Select I32/U32 SCTL 3	0	32
Greater? Boolean SCTL 1	0	2
Greater? Boolean SCTL 2	0	2
Not SCTL	0	1
Increment I32/U32 SCTL	0	32
Subtract I32/U32 SCTL	0	33
Tick count 32-bit SCTL	38	40
Loop timer 32-bit	73	126
Structures		
Timed Loop 1	0	0
Timed Loop 2	0	0
Case structure SCTL	0	0
Flat sequence SCTL	0	0
While loop, i terminal unwired	34	50
Stacked sequence	0	0
Controls and indicators		
7x Boolean	14	35
4x U32	208	156
Base	**388**	**440**
Total	**755**	**1013**

3. Resource utilization tables for the functions on the FPGA palette

3.1. Structures

Name of VI/Element	Data Type	Outside SCTL		Inside SCTL	
		Flip Flops	LUTs	Flip Flops	LUTs
Structures					
For Loop, *i* terminal wired		35	133	Not supported	
For Loop, *i* terminal unwired		35	101	Not supported	
While Loop, *i* terminal wired		2	6	Not supported	
While Loop, *i* terminal unwired		34	50	Not supported	
Timed Structures		0	0		
Timed Loop				Not supported	
FPGA Clk Const		0	0	0	0
Case Structure		5	15	0	0
Flat Sequence Structure		0	0	No sequencing effect	
Stacked Sequence Structure		0	0	No sequencing effect	
Diagram Disable Structure		0	0	0	0
Conditional Disable Structure		0	0	0	0
Local Variable	Boolean	3	3	0	0
	I8/U8	3	3	0	0
	I16/U16	3	3	0	0
	I32/U32	3	3	0	0
	I64/U64	3	3	0	0
Decorations		0	0	0	0
Free Label					
Thin Line		0	0	0	0
Thin Line with Arrow		0	0	0	0
Flat Frame		0	0	0	0
Thick Line		0	0	0	0
Thick Line with Arrow		0	0	0	0
Feedback Node	Boolean	3	5	2	2
	I8/U8	10	12	9	9
	I16/U16	18	20	17	17
	I32/U32	35	36	34	33
	I64/U64	67	68	68	65

3.2. Array

Name of VI/Element	Data Type	Array Size	Outside SCTL		Inside SCTL	
			Flip Flops	LUTs	Flip Flops	LUTs
Array						
Array Size			**0**	**0**	**0**	**0**
Index Array	I32/U32	10	**33**	**262**	**0**	**262**
		20	**33**	**483**	**0**	**481**
		30	**33**	**591**	**0**	**589**
		40	**33**	**951**	**0**	**949**
		50	**33**	**879**	**0**	**877**
Replace Array Subset	I32/U32	10	**321**	**341**	**0**	**339**
		20	**641**	**673**	**0**	**671**
		30	**961**	**1009**	**0**	**1007**
		40	**1281**	**1343**	**0**	**1341**
		50	**1601**	**1677**	**0**	**1675**
Insert Into Array			**0**	**0**	**0**	**0**
Delete From Array			**0**	**0**	**0**	**0**
Initialize Array			**0**	**0**	**0**	**0**
Build Array			**0**	**0**	**0**	**0**
Array Subset			**0**	**0**	**0**	**0**
Rotate 1D Array	I32/U32	10	**0**	**0**	Not supported	
		20	**0**	**0**		
		30	**0**	**0**		
		40	**0**	**0**		
		50	**0**	**0**		
Reverse 1D Array			**0**	**0**	**0**	**0**
Split 1D Array			**0**	**0**	**0**	**0**
Interleave 1D Arrays			**0**	**0**	**0**	**0**
Decimate 1D Array			**0**	**0**	**0**	**0**
Array Constant			**0**	**0**	**0**	**0**
Array to Cluster			**0**	**0**	**0**	**0**
Cluster to Array			**0**	**0**	**0**	**0**
Reshape Array			**3**	**3**	**0**	**0**

3.3. Cluster & Variant

Name of VI/Element	Outside SCTL		Inside SCTL	
	Flip Flops	LUTs	Flip Flops	LUTs
Cluster & variant				
Unbundle	**0**	**0**	**0**	**0**
Bundle	**0**	**0**	**0**	**0**
Unbundle by Name	**0**	**0**	**0**	**0**
Bundle by Name	**0**	**0**	**0**	**0**
Cluster Constant	**0**	**0**	**0**	**0**
Cluster to Array	**0**	**0**	**0**	**0**
Array to Cluster	**0**	**0**	**0**	**0**

3.4. Numeric

Name of VI/Element	Input (s) Data Type	Output Data Type	Outside SCTL			Inside SCTL		
			Flip Flops	LUTs	MULT 18x18s	Flip Flops	LUTs	MULT 18x18s
Numeric								
Add	I8/U8	I8/U8	**9**	**11**	**0**	**0**	**9**	**0**
	I16/U16	I16/U16	**17**	**19**	**0**	**0**	**17**	**0**
	I32/U32	I32/U32	**33**	**35**	**0**	**0**	**33**	**0**
	I64/U64	I64/U64	**65**	**67**	**0**	**0**	**65**	**0**
	FXP<±,16,8>	FXP<±,16,8>	**18**	**20**	**0**	**0**	**18**	**0**
	FXP<±,32,16>	FXP<±,32,16>	**34**	**36**	**0**	**0**	**34**	**0**
	FXP<±,64,32>	FXP<±,64,32>	**65**	**196**	**0**	**0**	**194**	**0**
Subtract	I8/U8	I8/U8	**9**	**11**	**0**	**0**	**9**	**0**
	I16/U16	I16/U16	**17**	**19**	**0**	**0**	**17**	**0**
	I32/U32	I32/U32	**33**	**35**	**0**	**0**	**33**	**0**
	I64/U64	I64/U64	**65**	**67**	**0**	**0**	**65**	**0**
	FXP<±,16,8>	FXP<±,16,8>	**18**	**20**	**0**	**0**	**18**	**0**
	FXP<±,32,16>	FXP<±,32,16>	**34**	**36**	**0**	**0**	**34**	**0**
	FXP<±,64,32>	FXP<±,64,32>	**65**	**259**	**0**	**0**	**257**	**0**
Multiply	I8/U8	I16/U16	**18**	**3**	**1**	**0**	**1**	**1**
	I16/U16	I32/U32	**34**	**3**	**1**	**0**	**1**	**1**
	I32/U32	I64/U64	**33**	**33**	**3**	**0**	**31**	**3**
	FXP<±,16,8>	FXP<±,32,16>	**66**	**3**	**1**	**0**	**1**	**1**
	FXP<±,32,16>	FXP<±,64,32>	**65**	**114**	**4**	**0**	**112**	**4**

Quotient & Remainder	I8/U8	I8/U8	67	281	0			
	I16/U16	I16/U16	121	487	0			
	I32/U32	I32/U32	237	867	0		Not supported	
	I64/U64	I64/U64	459	1690	0			
Conversion To Byte Integer			0	1	0	0	1	0
To Word Integer			0	1	0	0	1	0
To Long Integer			0	1	0	0	1	0
To Quad Integer			0	1	0	0	1	0
To Unsigned Byte Integer			0	1	0	0	1	0
To Unsigned Word Integer			0	1	0	0	1	0
To Unsigned Long Integer			0	1	0	0	1	0
To Unsigned Quad Integer			0	1	0	0	1	0
To Fixed Point	I16	FXP<±,16,16>	0	1	0	0	1	0
	I32	FXP<±,32,32>	0	1	0	0	1	0
	I64	FXP<±,64,64>	0	1	0	0	1	0
Number to Boolean Array			0	0	0	0	0	0
Boolean Array to Number			0	1	0	0	1	0
Boolean to (0,1)			0	1	0	0	1	0
Increment	I8/U8	I8/U8	9	12	0	0	10	0
	I16/U16	I16/U16	17	18	0	0	16	0
	I32/U32	I32/U32	33	34	0	0	32	0
	I64/U64	I64/U64	65	66	0	0	64	0
	FXP<±,16,8>	FXP<±,16,8>	18	14	0	0	12	0
	FXP<±,32,16>	FXP<±,32,16>	34	20	0	0	18	0
	FXP<±,64,32>	FXP<±,64,32>	65	164	0	0	162	0
Decrement	I8/U8	I8/U8	9	13	0	0	11	0
	I16/U16	I16/U16	17	19	0	0	17	0
	I32/U32	I32/U32	33	35	0	0	33	0
	I64/U64	I64/U64	65	67	0	0	65	0
	FXP<±,16,8>	FXP<±,16,8>	18	14	0	0	12	0
	FXP<±,32,16>	FXP<±,32,16>	34	20	0	0	18	0
	FXP<±,64,32>	FXP<±,64,32>	65	164	0	0	162	0
Data Manipulation Rotate Left with Carry	I8/U8	I8/U8	0	0	0	0	0	0
	I16/U16	I16/U16	0	0	0	0	0	0
	I32/U32	I32/U32	0	0	0	0	0	0
	I64/U64	I64/U64	0	0	0	0	0	0
Rotate Right with Carry	I8/U8	I8/U8	0	0	0	0	0	0
	I16/U16	I16/U16	0	0	0	0	0	0
	I32/U32	I32/U32	0	0	0	0	0	0
	I64/U64	I64/U64	0	0	0	0	0	0
Logical Shift	I8/U8	I8/U8	9	64	0	0	62	0
	I16/U16	I16/U16	17	153	0	0	156	0
	I32/U32	I32/U32	33	415	0	0	409	0

	I64/U64	I64/U64	65	871	0	0	869	0
Rotate	I8/U8	I8/U8	9	50	0	0	48	0
	I16/U16	I16/U16	17	182	0	0	180	0
	I32/U32	I32/U32	33	650	0	0	648	0
	I64/U64	I64/U64	65	712	0	0	710	0
Split Number			0	0	0	0	0	0
Join Numbers			0	1	0	0	1	0
Swap Bytes			0	0	0	0	0	0
Swap Words			0	0	0	0	0	0
Absolute Value	I8/U8	I8/U8	9	12	0	0	10	0
	I16/U16	I16/U16	17	34	0	0	32	0
	I32/U32	I32/U32	33	66	0	0	64	0
	I64/U64	I64/U64	65	130	0	0	128	0
	FXP<±,16,8>	FXP<±,16,8>	17	54	0	0	52	0
	FXP<±,32,16>	FXP<±,32,16>	33	106	0	0	104	0
	FXP<±,64,32>	FXP<±,64,32>	65	210	0	0	208	0
Round to Nearest	I8/U8	I8/U8	18	3	0	0	1	0
	I16/U16	I16/U16	34	3	0	0	1	0
	I32/U32	I32/U32	66	3	0	0	1	0
	I64/U64	I64/U64	130	3	0	0	1	0
	FXP<±,16,8>	FXP<±,9,9>	10	16	0	0	14	0
	FXP<±,32,16>	FXP<±,17,17>	18	62	0	0	60	0
	FXP<±,64,32>	FXP<±,33,33>	34	118	0	0	116	0
Round Toward -Infinity	I8/U8	I8/U8	0	1	0	0	1	0
	I16/U16	I16/U16	0	1	0	0	1	0
	I32/U32	I32/U32	0	1	0	0	1	0
	I64/U64	I64/U64	0	1	0	0	1	0
	FXP<±,16,8>	FXP<±,8,8>	18	3	0	0	1	0
	FXP<±,32,16>	FXP<±,16,16>	34	3	0	0	1	0
	FXP<±,64,32>	FXP<±,32,32>	66	3	0	0	1	0
Round Toward +Infinity	I8/U8	I8/U8	18	3	0	0	1	0
	I16/U16	I16/U16	34	3	0	0	1	0
	I32/U32	I32/U32	66	3	0	0	1	0
	I64/U64	I64/U64	130	3	0	0	1	0
	FXP<±,16,8>	FXP<±,9,8>	10	17	0	0	15	0
	FXP<±,32,16>	FXP<±,17,17>	18	48	0	0	46	0
	FXP<±,64,32>	FXP<±,33,33>	34	88	0	0	86	0
Scale by Power of 2	I8/U8	I8/U8	9	67	0	0	65	0
	I16/U16	I16/U16	17	174	0	0	172	0
	I32/U32	I32/U32	33	426	0	0	424	0
	I64/U64	I64/U64	65	917	0	0	913	0
Square	I8/U8	I16/U16	18	3	1	0	1	1
	I16/U16	I32/U32	34	3	1	0	1	1
	I32/U32	I64/U64	33	33	3	0	31	3
	FXP<±,16,8>	FXP<±,32,16>	64	55	1	0	51	1

Name	Data Type (in)	Data Type (out)	Flip Flops	LUTs	MULT 18x18s	Flip Flops	LUTs	MULT 18x18s
	FXP<±,32,16>	FXP<±,64,32>	128	219	4	0	209	4
Negate	I8/U8	I8/U8	9	12	0	0	10	0
	I16/U16	I16/U16	17	19	0	0	17	0
	I32/U32	I32/U32	33	35	0	0	33	0
	I64/U64	I64/U64	65	67	0	0	65	0
	FXP<±,15,7>	FXP<±,16,8>	17	19	0	0	17	0
	FXP<±,31,15>	FXP<±,32,16>	33	35	0	0	33	0
	FXP<±,63,31>	FXP<±,64,32>	65	67	0	0	65	0
Reciprocal	FXP<±,32,16>	FXP<±,33,18>	244	481	0	Not supported		
Sign	I8/U8	I8/U8	9	12	0	0	10	0
	I16/U16	I16/U16	17	19	0	0	17	0
	I32/U32	I32/U32	33	35	0	0	33	0
	I64/U64	I64/U64	65	67	0	0	65	0
	FXP<±,16,8>	FXP<±,2,2>	6	8	0	0	6	0
	FXP<±,32,16>	FXP<±,2,2>	3	11	0	0	9	0
	FXP<±,64,32>	FXP<±,2,2>	3	20	0	0	18	0
Fixed-Point Clear Overflow Status	FXP<±,32,16>	FXP<±,32,16>	0	0	0	0	0	0
Remove Overflow Status	FXP<±,32,16>	FXP<±,32,16>	0	0	0	0	0	0
Include Overflow Status	FXP<±,32,16>	FXP<±,32,16>	0	0	0	0	0	0
Numeric Constant			0	0	0	0	0	0
Enum Constant			0	0	0	0	0	0
Ring Constant			0	0	0	0	0	0

Name of VI/Element	Data Type	Number of inputs	Mode	Outside SCTL			Inside SCTL		
				Flip Flops	LUTs	MULT 18x18s	Flip Flops	LUTs	MULT 18x18s
Compound Arithmetic	I32/U32	2	Add	33	34	0	0	32	0
		4	Add	33	96	0	0	94	0
		8	Add	33	222	0	0	220	0
		2	Multiply	33	32	3	0	30	3
		4	Multiply	98	93	9	0	90	9
		8	Multiply	267	714	18	0	710	18
		2	AND	33	34	0	0	32	0
		4	AND	33	34	0	0	32	0
		8	AND	33	98	0	0	96	0
		2	OR	33	34	0	0	32	0
		4	OR	33	34	0	0	32	0
		8	OR	33	98	0	0	96	0
		2	XOR	33	34	0	0	32	0
		4	XOR	33	34	0	0	32	0
		8	XOR	33	98	0	0	96	0

Boolean	2	AND	4	3	0	0	1	0
	4	AND	4	3	0	0	1	0
	8	AND	4	5	0	0	3	0
	2	OR	4	3	0	0	1	0
	4	OR	4	3	0	0	1	0
	8	OR	4	5	0	0	3	0
	2	XOR	4	3	0	0	1	0
	4	XOR	4	3	0	0	1	0
	8	XOR	4	5	0	0	3	0

3.5. Boolean

Name of VI/Element	Data Type	Outside SCTL		Inside SCTL	
		Flip Flops	LUTs	Flip Flops	LUTs
Boolean					
And	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64
Or	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64
Exclusive Or	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64
Not	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64
Compound Arithmetic	see the second table in section 3.4. Numeric				
Not And	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64

		Outside SCTL		Inside SCTL	
		Flip Flops	LUTs	Flip Flops	LUTs
Not Or	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64
Not Exclusive Or	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64
Implies	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64
Number to Boolean Array	I8/U8	0	0	0	0
	I16/U16	0	0	0	0
	I32/U32	0	0	0	0
	I64/U64	0	0	0	0
Boolean Array to Number	I8/U8	0	1	0	1
	I16/U16	0	1	0	1
	I32/U32	0	1	0	1
	I64/U64	0	1	0	1
Boolean to (0,1)		0	1	0	1
True Constant		0	0	0	0
False Constant		0	0	0	0

Name of VI/Element	Array Size	Outside SCTL		Inside SCTL	
		Flip Flops	LUTs	Flip Flops	LUTs
And Array Elements	8	4	5	0	3
	16	4	7	0	5
	32	2	10	0	8
	64	2	18	0	16
Or Array Elements	8	4	5	0	3
	16	4	7	0	5
	32	2	10	0	8
	64	2	18	0	16

3.6. Comparison

Name of VI/Element	Data Type	Outside SCTL		Inside SCTL	
		Flip Flops	LUTs	Flip Flops	LUTs
Comparison					
Equal?	Boolean	4	4	0	2
	I8/U8	4	8	0	6
	I16/U16	2	11	0	9
	I32/U32	2	19	0	17
	I64/U64	2	35	0	33
	FXP<±,16,8>	2	11	0	9
	FXP<±,32,16>	2	19	0	17
	FXP<±,64,32>	2	35	0	33
Not Equal?	Boolean	4	4	0	2
	I8/U8	4	8	0	6
	I16/U16	2	12	0	10
	I32/U32	2	20	0	18
	I64/U64	2	36	0	34
	FXP<±,16,8>	2	12	0	10
	FXP<±,32,16>	2	20	0	18
	FXP<±,64,32>	2	36	0	34
Greater?	Boolean	4	4	0	2
	I8/U8	2	12	0	10
	I16/U16	2	20	0	18
	I32/U32	2	36	0	34
	I64/U64	2	68	0	66
	FXP<±,16,8>	2	20	0	18
	FXP<±,32,16>	2	36	0	34
	FXP<±,64,32>	2	68	0	66
Less?	Boolean	4	4	0	2
	I8/U8	2	12	0	10
	I16/U16	2	20	0	18
	I32/U32	2	36	0	34
	I64/U64	2	68	0	66
	FXP<±,16,8>	2	20	0	18
	FXP<±,32,16>	2	36	0	34
	FXP<±,64,32>	2	68	0	66
Greater or Equal?	Boolean	4	4	0	2
	I8/U8	2	13	0	11
	I16/U16	2	21	0	19
	I32/U32	2	37	0	35
	I64/U64	2	69	0	67
	FXP<±,16,8>	2	21	0	19
	FXP<±,32,16>	2	37	0	35
	FXP<±,64,32>	2	69	0	67

Less or Equal?	Boolean	4	4	0	2
	I8/U8	2	13	0	11
	I16/U16	2	21	0	19
	I32/U32	2	37	0	35
	I64/U64	2	69	0	67
	FXP<±,16,8>	2	21	0	19
	FXP<±,32,16>	2	37	0	35
	FXP<±,64,32>	2	69	0	67
Equal to 0?	I8/U8	4	6	0	4
	I16/U16	4	8	0	6
	I32/U32	2	11	0	9
	I64/U64	2	19	0	17
	FXP<±,16,8>	4	8	0	6
	FXP<±,32,16>	2	11	0	9
	FXP<±,64,32>	2	19	0	17
Not Equal to 0?	I8/U8	4	6	0	4
	I16/U16	4	8	0	6
	I32/U32	2	11	0	9
	I64/U64	2	20	0	18
	FXP<±,16,8>	4	8	0	6
	FXP<±,32,16>	2	11	0	9
	FXP<±,64,32>	2	20	0	18
Greater Than 0?	I8/U8	4	5	0	3
	I16/U16	2	20	0	18
	I32/U32	2	36	0	34
	I64/U64	2	68	0	66
	FXP<±,16,8>	2	20	0	18
	FXP<±,32,16>	2	36	0	34
	FXP<±,64,32>	2	68	0	66
Less Than 0?	I8/U8	0	1	0	1
	I16/U16	0	1	0	1
	I32/U32	0	1	0	1
	I64/U64	0	1	0	1
	FXP<±,16,8>	0	1	0	1
	FXP<±,32,16>	0	1	0	1
	FXP<±,64,32>	0	1	0	1
Greater or Equal to 0?	I8/U8	0	2	0	2
	I16/U16	0	2	0	2
	I32/U32	0	2	0	2
	I64/U64	0	2	0	2
	FXP<±,16,8>	0	2	0	2
	FXP<±,32,16>	0	2	0	2
	FXP<±,64,32>	0	2	0	2
Less or Equal to 0?	I8/U8	4	5	0	3
	I16/U16	2	21	0	19

	I32/U32	2	37	0	35
	I64/U64	2	69	0	67
	FXP<±,16,8>	2	21	0	19
	FXP<±,32,16>	0	2	0	2
	FXP<±,64,32>	0	2	0	2
Select?	Boolean	4	3	0	1
	I8/U8	9	10	0	8
	I16/U16	17	18	0	16
	I32/U32	33	34	0	32
	I64/U64	65	66	0	64
	FXP<±,16,8>	17	18	0	16
	FXP<±,32,16>	33	34	0	32
	FXP<±,64,32>	65	66	0	64
Max & Min?	I8/U8	17	28	0	26
	I16/U16	33	52	0	50
	I32/U32	65	100	0	98
	I64/U64	129	196	0	194
	FXP<±,16,8>	33	52	0	50
	FXP<±,32,16>	65	100	0	98
	FXP<±,64,32>	129	196	0	194
In Range and Coerce?	I8/U8	10	38	0	36
	I16/U16	18	70	0	68
	I32/U32	34	134	0	132
	I64/U64	66	262	0	260
	FXP<±,16,8>	18	70	0	68
	FXP<±,32,16>	34	134	0	132
	FXP<±,64,32>	66	262	0	260
Fixed-Point Overflow?	FXP<±,16,8>	0	0	0	0
	FXP<±,32,16>	0	0	0	0
	FXP<±,64,32>	0	0	0	0

3.7. Timing

Name of VI/Element	Size of Internal Counter (bits)	Outside SCTL		Inside SCTL	
		Flip Flops	LUTs	Flip Flops	LUTs
Timing					
Loop Timer	8	25	48	Not supported	
	16	41	74		
	32	73	126		
Wait	8	39	58		
	16	63	75		
	32	112	122		
Tick Count	8	24	19	14	16
	16	40	27	22	24
	32	72	43	38	40

3.8. Memory & FIFO

3.8.1. Memory

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
Boolean	1024	if multiple requestors only	if multiple requestors only			**1**
Functions						
1 Read, 1 Write				**6**	**27**	
2 Reads, 2 Writes				**68**	**140**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I8	1024	if multiple requestors only	if multiple requestors only			**1**
Functions						
1 Read, 1 Write				**6**	**27**	
2 Reads, 2 Writes				**89**	**154**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I16	1024	if multiple requestors only	if multiple requestors only			**1**
Functions						
1 Read, 1 Write				**6**	**27**	
2 Reads, 2 Writes				**113**	**170**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I16	1024	always	always			**1**
Functions						
1 Read, 1 Write				**80**	**117**	
2 Reads, 2 Writes				**113**	**170**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I16	1024	never	never			**1**
Functions						
1 Read, 1 Write				**6**	**27**	
2 Reads, 2 Writes				**103**	**167**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I32	1024	if multiple requestors only	if multiple requestors only			**2**
Functions						
1 Read, 1 Write				**6**	**27**	
2 Reads, 2 Writes				**161**	**202**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I32	1024	always	always			**2**
Functions						
1 Read, 1 Write				**112**	**149**	
2 Reads, 2 Writes				**161**	**202**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I32	1024	never	never			**2**
Functions						
1 Read, 1 Write				**6**	**27**	
2 Reads, 2 Writes				**151**	**215**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I64	1024	if multiple requestors only	if multiple requestors only			**4**
Functions						
1 Read, 1 Write				**6**	**27**	
2 Reads, 2 Writes				**257**	**266**	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I64	1024	always	always			4
Functions						
1 Read, 1 Write				176	213	
2 Reads, 2 Writes				257	266	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I64	1024	never	never			4
Functions						
1 Read, 1 Write				6	27	
2 Reads, 2 Writes				248	311	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I64	4096	if multiple requestors only	if multiple requestors only			15
Functions						
1 Read, 1 Write				7	119	
2 Reads, 2 Writes				262	357	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I64	4096	always	always			15
Functions						
1 Read, 1 Write				181	306	
2 Reads, 2 Writes				262	357	

Data Type	Number of elements	Arbitration		Resources		
		Read	Write	Flip Flops	LUTs	Block RAMs
I64	4096	never	never			15
Functions						
1 Read, 1 Write				7	119	
2 Reads, 2 Writes				248	408	

3.8.2. FIFO

Data Type	Number of elements	Implementation	Arbitration		Resources		
			Read	Write	Flip Flops	LUTs	Block RAMs
I16	1028	Block memory	if multiple requestors only	if multiple requestors only			**1**
Functions							
1 Read, 1 Write					**300**	**302**	
1 Read, 1 Write, 1 Clear					**337**	**340**	
2 Reads, 2 Writes, 1 Clear					**481**	**491**	

Data Type	Number of elements	Implementation	Arbitration		Resources		
			Read	Write	Flip Flops	LUTs	Block RAMs
I16	1028	Block memory	always	always			**1**
Functions							
1 Read, 1 Write					**300**	**302**	
1 Read, 1 Write, 1 Clear					**337**	**340**	
2 Reads, 2 Writes, 1 Clear					**466**	**457**	

Data Type	Number of elements	Implementation	Arbitration		Resources		
			Read	Write	Flip Flops	LUTs	Block RAMs
I16	1028	Block memory	never	never			**1**
Functions							
1 Read, 1 Write					**300**	**302**	
1 Read, 1 Write, 1 Clear					**337**	**340**	
2 Reads, 2 Writes, 1 Clear					**466**	**457**	

Data Type	Number of elements	Implementation	Arbitration		Resources		
			Read	Write	Flip Flops	LUTs	Block RAMs
I32	1028	Block memory	if multiple requestors only	if multiple requestors only			**2**
Functions							
1 Read, 1 Write					**412**	**398**	
1 Read, 1 Write, 1 Clear					**449**	**442**	
2 Reads, 2 Writes, 1 Clear					**641**	**619**	

Data Type	Number of elements	Implementation	Arbitration		Resources		
			Read	Write	Flip Flops	LUTs	Block RAMs
I32	1028	Block memory	always	always			**2**
Functions							
1 Read, 1 Write					**559**	**563**	
1 Read, 1 Write, 1 Clear					**597**	**598**	
2 Reads, 2 Writes, 1 Clear					**641**	**619**	

Data Type	Number of elements	Implementation	Arbitration		Resources		
			Read	Write	Flip Flops	LUTs	Block RAMs
I32	1028	Block memory	never	never			**2**
Functions							
1 Read, 1 Write					**412**	**398**	
1 Read, 1 Write, 1 Clear					**449**	**442**	
2 Reads, 2 Writes, 1 Clear					**626**	**585**	

3.8.3. FIFO – DMA host-to-target

Data Type	Number of elements	Read arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I16	1023	if multiple requestors only			**1**
Functions					
Read			**415**	**612**	

Data Type	Number of elements	Read arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I16	1023	always			**1**
Functions					
Read			**458**	**660**	

Data Type	Number of elements	Read arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I16	1023	never			**1**
Functions					
Read			**415**	**611**	

Data Type	Number of elements	Read arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I32	1023	if multiple requestors only			**2**
Functions					
Read			**608**	**758**	

Data Type	Number of elements	Read arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I32	1023	always			**2**
Functions					
Read			**667**	**823**	

Data Type	Number of elements	Read arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I32	1023	never			**2**
Functions					
Read			**609**	**759**	

3.8.4. FIFO – DMA target-to-host

Data Type	Number of elements	Write arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I16	1023	if multiple requestors only			**1**
Functions					
Write			**266**	**355**	

Data Type	Number of elements	Write arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I16	1023	always			**1**
Functions					
Write			**309**	**406**	

Data Type	Number of elements	Write arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I16	1023	never			**1**
Functions					
Write			**266**	**355**	

Data Type	Number of elements	Write arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I32	1023	if multiple requestors only			**2**
Functions					
Write			**363**	**397**	

Data Type	Number of elements	Write arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I32	1023	always			**2**
Functions					
Write			**422**	**461**	

Data Type	Number of elements	Write arbitration	Resources		
			Flip Flops	LUTs	Block RAMs
I32	1023	never			**2**
Functions					
Write			**362**	**396**	

3.9. FPGA Math & Analysis

Name of VI/element	Outside SCTL				Inside SCTL			
	Flip Flops	LUTs	Block RAMs	MULT 18x18s	Flip Flops	LUTs	Block RAMs	MULT 18x18s
FPGA Math & Analysis								

Control
  Discrete Nonlinear Systems
  Backlash
	83	**145**	**0**	**0**	**20**	**85**	**0**	**0**

Friction	72	89	0	1	0	1	0	0
Quantizer	2	36	0	0	0	0	0	0
Dead Zone	40	109	0	0	0	81	0	0
Rate Limiter	62	91	0	0	18	56	0	0
Relay	12	33	0	0	0	0	0	0
Saturate	66	3	0	0	0	1	0	0
Switch	19	37	0	0	0	34	0	0
Boolean Crossing	11	15	0	0	3	3	0	0
Zero Crossing	13	35	0	0	4	22	0	0
Memory Element	24	28	0	0	18	18	0	0
Trigger	27	49	0	0	18	37	0	0
Discrete Linear Systems Normalized Integrator	126	134	0	0	50	87	0	0
Unit Delay	24	28	0	0	18	18	0	0
Delay	67	38	0	0	Not supported			
Zero-Order Hold	26	31	0	0	0	0	0	0
Initial Condition	7	26	0	0	2	18	0	0
Control Filter	225	215	0	5	66	164	0	3
Utilities Linear Interpolation	17	36	0	1	0	34	0	1
Saturate	66	3	0	0	0	1	0	0
Zero Crossing	13	35	0	0	4	22	0	0
Boolean Crossing	11	15	0	0	3	3	0	0
Unit Delay	24	28	0	0	18	18	0	0
Discrete Delay	67	38	0	0	Not supported			
Generation Sine Wave	47	78	1	0	Not supported			
Square Wave	87	153	0	0	22	41	0	0
White Noise	52	99	0	3	35	96	0	3
Scaled Window	137	140	1	2	139	148	1	2
Analog Period Measurement	202	374	0	0	Not supported			

3.9.1. Look-up Table 1D

Look-Up Table Specifications				Resources			
Number of elements	Data type	Memory size	Interpolate data	Flip Flops	LUTs	Block RAMs	MULT 18x18s
1024	I16	2 KB	yes	**40**	**17**	**1**	**0**
1024	U16	2 KB	yes	**40**	**17**	**1**	**0**
1024	I8	1 KB	yes	**32**	**17**	**1**	**0**
1024	I32	4 KB	yes	**57**	**18**	**2**	**0**
2048	I16	4 KB	yes	**42**	**18**	**2**	**0**
4096	I16	8 KB	yes	**44**	**16**	**4**	**0**
2048	I8	2 KB	yes	**32**	**17**	**1**	**0**
4096	I8	4 KB	yes	**36**	**16**	**2**	**0**
2048	I32	8 KB	yes	**59**	**19**	**4**	**0**
4096	I32	16 KB	yes	**61**	**17**	**8**	**0**
16384	I32	64 KB	yes	**71**	**283**	**29**	**0**

3.9.2. Butterworth filter

Butterworth filter configuration options							Resources			
Channels	Input	Output	Type	Order	Cutoff frequency	Sample rate	Flip Flops	LUTs	Block RAMs	MULT 18x18s
1	I16	I16	Lowpass	1	1kHz	50kS/s	**120**	**321**	**0**	**3**
1	I16	I16	Lowpass	2	1kHz	50kS/s	**181**	**433**	**0**	**3**
1	I16	I16	Lowpass	4	1kHz	50kS/s	**318**	**797**	**0**	**3**
1	I32	I32	Lowpass	1	1kHz	50kS/s	**171**	**442**	**0**	**4**
1	I32	I32	Lowpass	2	1kHz	50kS/s	**238**	**584**	**0**	**4**
1	I32	I32	Lowpass	4	1kHz	50kS/s	**381**	**967**	**0**	**4**
2*	I16	I16	Lowpass	2	1kHz	50kS/s	**192**	**546**	**1**	**4**
4*	I16	I16	Lowpass	2	1kHz	50kS/s	**192**	**538**	**1**	**4**
8*	I16	I16	Lowpass	2	1kHz	50kS/s	**192**	**539**	**1**	**4**

* Multichannel support for the Butterworth filter is restricted to 16-bit resolution and 2nd order.

3.9.3. Notch filter

Notch filter configuration options				Resources			
Number of channels	Input data type	Expected sample rate (kS/s)	Frequency (kHz)	Flip Flops	LUTs	Block RAMs	MULT 18x18s
1	I16	50	1	**179**	**498**	**0**	**4**
1	I32	50	1	**233**	**608**	**0**	**4**
2	I16	50	1	**173**	**522**	**1**	**4**
4	I16	50	1	**173**	**517**	**1**	**4**
8	I16	50	1	**173**	**517**	**1**	**4**
16	I16	50	1	**173**	**518**	**1**	**4**
32	I16	50	1	**173**	**519**	**1**	**4**
64	I16	50	1	**173**	**517**	**1**	**4**

3.9.4. DC & RMS Measurements

DC & RMS configuration options					Resources			
Function	Input data type	Hanning window?	Expected sample rate	Measurement time	Flip Flops	LUTs	Block RAMs	MULT 18x18s
DC	I16	no	50kS/s	20m	**112**	**189**	**0**	**2**
DC	I16	yes	50kS/s	20m	**177**	**307**	**2**	**4**
DC	I32	no	50kS/s	20m	**209**	**285**	**0**	**3**
DC	I32	yes	50kS/s	20m	**257**	**416**	**2**	**6**
RMS	I16	no	50kS/s	20m	**266**	**363**	**0**	**3**
RMS	I16	yes	50kS/s	20m	**332**	**482**	**2**	**5**
RMS	I32	no	50kS/s	20m	**501**	**758**	**0**	**8**
RMS	I32	yes	50kS/s	20m	**549**	**889**	**2**	**11**
Sum	I16	no	50kS/s	20m	**105**	**164**	**0**	**0**
Sum	I32	no	50kS/s	20m	**186**	**230**	**0**	**0**
Mean Square	I16	no	50kS/s	20m	**196**	**238**	**0**	**3**
Mean Square	I32	no	50kS/s	20m	**367**	**520**	**0**	**8**
Square Sum	I16	no	50kS/s	20m	**147**	**221**	**0**	**1**
Square Sum	I32	no	50kS/s	20m	**367**	**520**	**0**	**8**

3.9.5. Fast Fourier Transform

The Fast Fourier Transform function is configurable for a range of input and output parameters. The resource usage table below is not exhaustive.

Using the FFT VI inside of a SCTL enables an option to select the throughput. Throughput for all tests shown below is set not equal to one. Additionally, all tests (inside and outside SCTL) were performed using an output type of adapt to source.

FFT configuration options				Resources			
Real data in data type	Length	Direction	Execution	Flip Flops	LUTs	Block RAMs	MULT 18x18s
FXP<±,16,8>	1024	Forward	Inside SCTL	697	2368	6	16
FXP<±,16,8>	1024	Forward	Outside SCTL	586	2671	10	16
FXP<±,16,8>	1024	Inverse	Inside SCTL	697	2363	6	16
FXP<±,16,8>	1024	Inverse	Outside SCTL	586	2671	10	16
FXP<±,16,8>	2048	Forward	Inside SCTL	680	2318	10	16
FXP<±,16,8>	2048	Forward	Outside SCTL	567	2647	18	16
FXP<±,16,8>	2048	Inverse	Inside SCTL	680	2318	10	16
FXP<±,16,8>	2048	Inverse	Outside SCTL	568	2647	18	16
FXP<±,32,16>	1024	Forward	Inside SCTL	688	2396	6	16
FXP<±,32,16>	1024	Forward	Outside SCTL	609	2757	10	16
FXP<±,32,16>	1024	Inverse	Inside SCTL	688	2398	6	16
FXP<±,32,16>	1024	Inverse	Outside SCTL	609	2758	10	16
FXP<±,32,16>	2048	Forward	Inside SCTL	674	2370	10	16
FXP<±,32,16>	2048	Forward	Outside SCTL	598	2757	18	16
FXP<±,32,16>	2048	Inverse	Inside SCTL	683	2411	10	16
FXP<±,32,16>	2048	Inverse	Outside SCTL	598	2744	18	16

3.9.6. Rational Resampler

Rational resampler configuration options					Resources			
Input data type	Number of channels	Execution	L	M	Flip Flops	LUTs	Block RAMs	MULT 18x18s
I32	1	Inside SCTL	1	2	337	566	2	4
I32	1	Inside SCTL	2	1	327	859	2	4
I32	1	Outside SCTL	1	2	205	387	2	4
I32	4	Inside SCTL	1	2	380	815	2	4
I32	4	Inside SCTL	2	1	333	878	2	4

I32	4	Outside SCTL	1	2	**220**	**422**	**2**	**4**
I32	8	Inside SCTL	1	2	**388**	**859**	**2**	**4**
I32	8	Inside SCTL	2	1	**339**	**900**	**2**	**4**
I32	8	Outside SCTL	1	2	**225**	**463**	**2**	**4**
FXP<±,32,16>	1	Inside SCTL	1	2	**337**	**566**	**2**	**4**
FXP<±,32,16>	1	Outside SCTL	1	2	**205**	**387**	**2**	**4**
FXP<±,32,16>	4	Inside SCTL	1	2	**380**	**815**	**2**	**4**
FXP<±,32,16>	4	Outside SCTL	1	2	**220**	**422**	**2**	**4**
FXP<±,32,16>	8	Inside SCTL	1	2	**388**	**859**	**2**	**4**
FXP<±,32,16>	8	Outside SCTL	1	2	**225**	**463**	**2**	**4**

3.10.   Synchronization

Name of VI/Element	Outside SCTL		Inside SCTL	
	Flip Flops	LUTs	Flip Flops	LUTs
Synchronization				
FIFO	see section 3.8 Memory & FIFO			
Occurrences Generate Occurrence	**0**	**1**	**0**	**1**
Wait on Occurrence	**1**	**3**	Not supported	
Wait on Occurrence with Timeout in Ticks	**102**	**103**	Not supported	
Set Occurrence	**2**	**5**	Not supported	
First Call?	**4**	**5**	**2**	**2**
Interrupt	**1**	**20**	Not supported	

3.11.   Advanced

Advanced FPGA programming techniques could include either the HDL Interface Node or CLIP node. These methods also utilize FPGA resources on your target; however, the amount of resources will vary based on how the code is written.