Week 1 – Week 2 Progress Summary

Jin Hu
July 7, 2005

Adders

Design Summary

Logic Utilization	8-bit Adder Used	16-bit Adder Used	32-bit Adder Used	64-bit Adder Used
# Slices	5	9	17	33
# Slice Flip-Flops	9	17	33	65
# 4-Input LUTs	8	16	32	64
# Bonded IOBs	27	51	99	195

Table 1: Adders' Space Requirements

Graphs

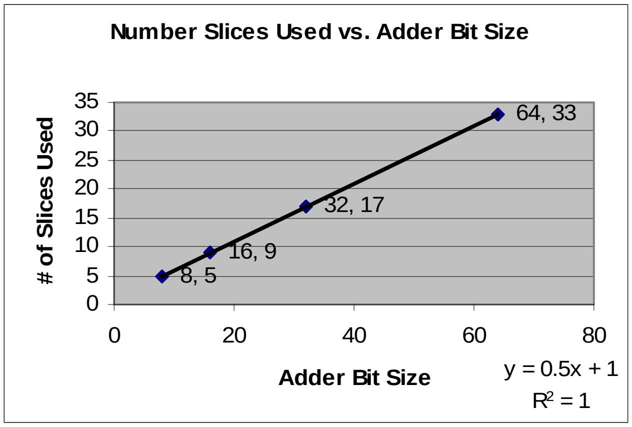

Figure 1: Graph of Number of Slices Used vs. Adders' Bit Size

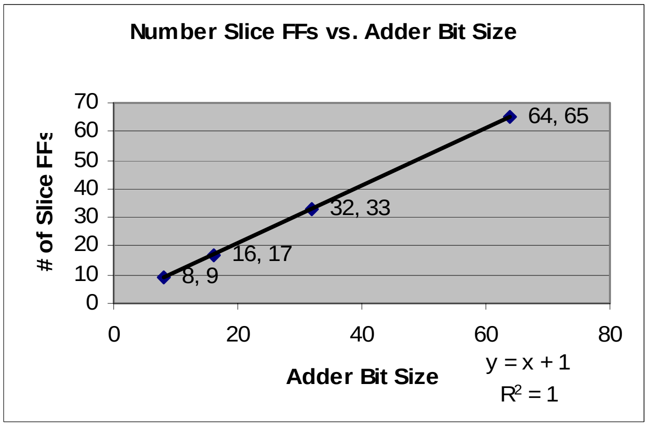

Figure 2: Graph of Number of Slice FFs vs. Adders' Bit Size

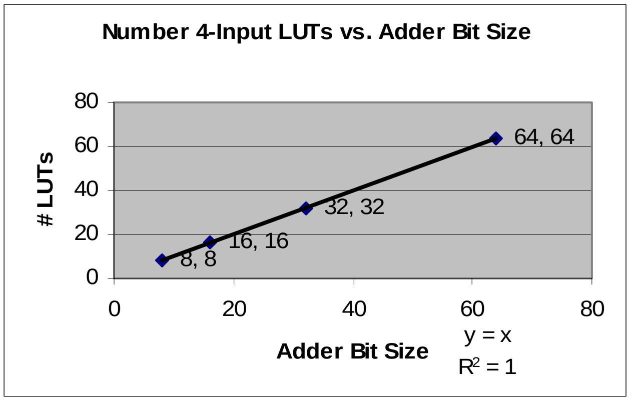

Figure 3: Graph of Number of 4-Input LUTs vs. Adders' Bit Size

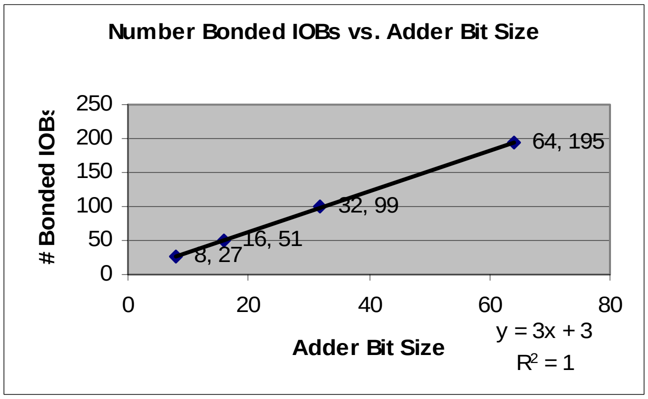

Figure 4: Graph of Number of Bonded IOBs vs. Adders' Bit Size

From the graphs, notice that the relationship between any of the parameters and the adder bit-size (8-bit, 16-bit adder) is **linear**. On each of the graph shows the linear regression as well as the correlation coefficient.

Logic Distribution Tables

Logic Distribution (8-bit Adder)	Used	Available	Utilization
# occupied Slices	5	depends on board	varies
# Slices containing only related logic	5	5	100%
# Slices containing unrelated logic	0	5	0%

Table 2: 8-bit Adder Logic Distribution Statistics

Logic Distribution (16-bit Adder)	Used	Available	Utilization
# occupied Slices	9	depends on board	varies
# Slices containing only related logic	9	9	100%
# Slices containing unrelated logic	0	9	0%

Table 2: 16-bit Adder Logic Distribution Statistics

Logic Distribution (32-bit Adder)	Used	Available	Utilization
# occupied Slices	17	depends on board	varies
# Slices containing only related logic	17	17	100%
# Slices containing unrelated logic	0	17	0%

Table 3: 32-bit Adder Logic Distribution Statistics

Logic Distribution (32-bit Adder)	Used	Available	Utilization
# occupied Slices	33	depends on board	varies
# Slices containing only related logic	33	33	100%
# Slices containing unrelated logic	0	33	0%

Table 4: 64-bit Adder Logic Distribution Statistics

This data on this page shows that out of all the slices used, all the slices contain relevant logic. In other words, all resources used are devoted to one module synthesis.

Constraint Testing

Type	Constraints	range* [slice coordinates]				xc4vfx12-11ff668		
	Passed?	top left x	top left y	bottom right x	bottom right y	x side length	y side length	total area [slices]
8-bit	**yes**	34	119	35	116	2	4	8
16-bit	**yes**	34	123	35	116	2	8	16
32-bit	**yes**	26	121	27	106	2	16	32
64-bit	**yes**	28	115	29	84	2	32	64

Table 5: Areas Constrained with Various Adders on a Virtex-4™ xc4vfx12 Model

Type	Constraints	range* [slice coordinates]				xc4vfx15-11ff668		
	Passed?	top left x	top left y	bottom right x	bottom right y	x side length	y side length	total area [slices]
8-bit	**yes**	36	99	37	96	2	4	8
16-bit	**yes**	34	123	35	116	2	8	16
32-bit	**yes**	28	117	29	102	2	16	32
64-bit	**yes**	34	119	35	88	2	32	64

Table 6: Areas Constrained with Various Adders on a Virtex-4™ xc4vfx15 Model

Type	Constraints	range* [slice coordinates]				xc4vfx20-11ff672		
	Passed?	top left x	top left y	bottom right x	bottom right y	x side length	y side length	total area [slices]
8-bit	**yes**	40	105	41	102	2	4	8
16-bit	**yes**	30	123	31	116	2	8	16
32-bit	**yes**	46	123	47	108	2	16	32
64-bit	**yes**	24	119	25	88	2	32	64

Table 7: Areas Constrained with Various Adders on a Virtex-4™ xc4vfx20 Model

range* refers to the area range constrained manually. The four columns denote the coordinates (in slices) of the top left and bottom right corner of the rectangle. The slice coordinates have the same properties as a XY plane in a Cartesian plane. For example, taking *Table 7*'s data for the 8-bit adder, the area would look like the following:

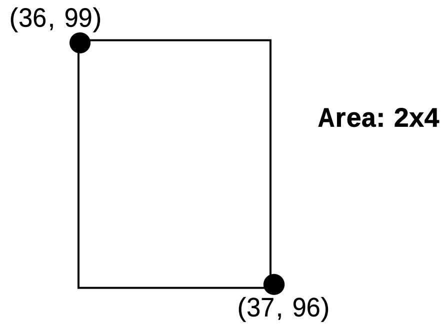

Figure 5: Example of Constraint Area

The side lengths of the rectangle (x side length, y side length) are then calculated along with the area in units of slices covered (total area).

As noticed in the tabulated data, all the adders fit within the constrained areas.

Constraint Graphs

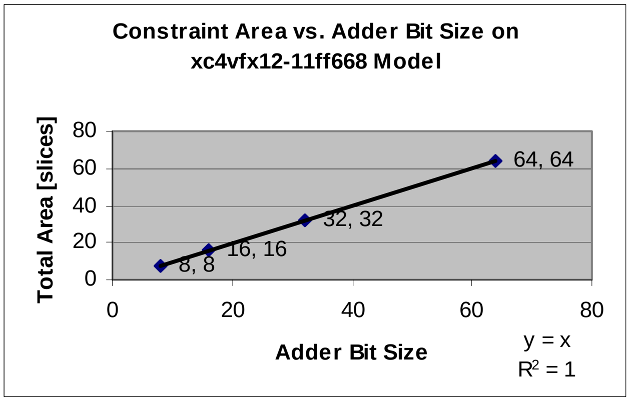

Figure 6: Graph of Area needed vs. Adders' Bit Size on the xc4vfx12-11ff668 board

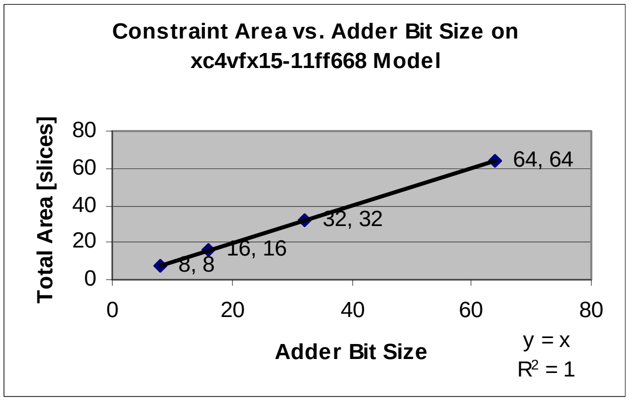

Figure 7: Graph of Area needed vs. Adders' Bit Size on the xc4vfx15-11ff668 board

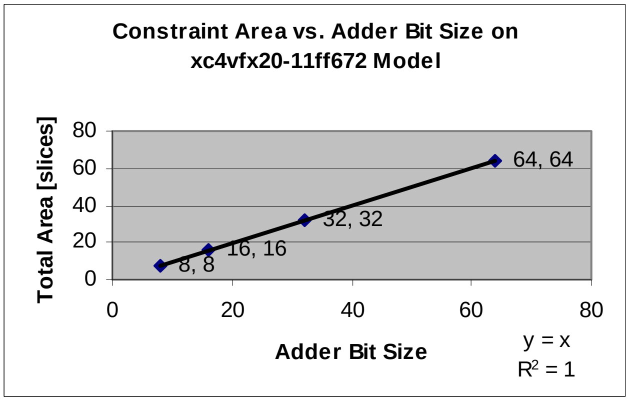

Figure 8: Graph of Area needed vs. Adders' Bit Size on the xc4vfx20-11ff672 board

From the graphs, the space needed to implement adders is just the adder's bit-width.

Pictures of Implemented Adders

One peculiar observation that I had noticed was that a given adder (e.g. 8-bit) generated under different Virtex-4™ boards yielded different results. Namely, if an adder is generated under a smaller board model, the adder will look differently than generated under a larger model. Below are two pictures that show the different more clearly.

The left 32-bit adder was generated under the xc4vfx12-11ff668 board while the right 32-bit adder was generated under the xc4vfx20-11ff672 board. The smaller board's model has an extra loop of wiring at the top while the bigger board's model is just a straight line. Both adders are representative of their type (i.e. an adder generated under those conditions will take on similar shape). The difference in the two pictures is highlighted in red.

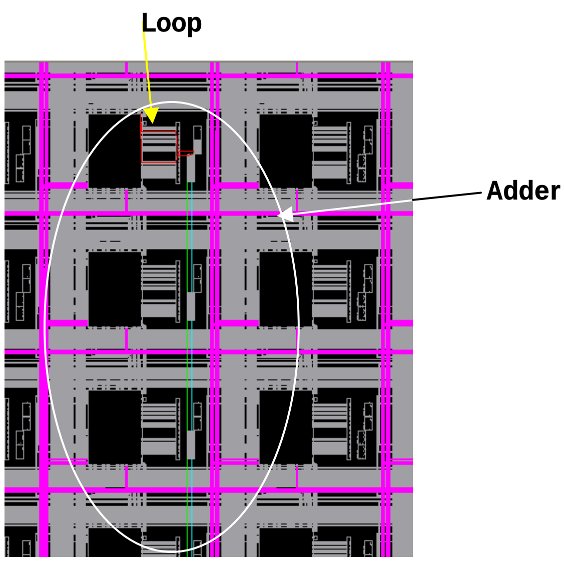

Figure 9: 32-bit Adder Generated Under xc4vfx12-11ff668 Parameters

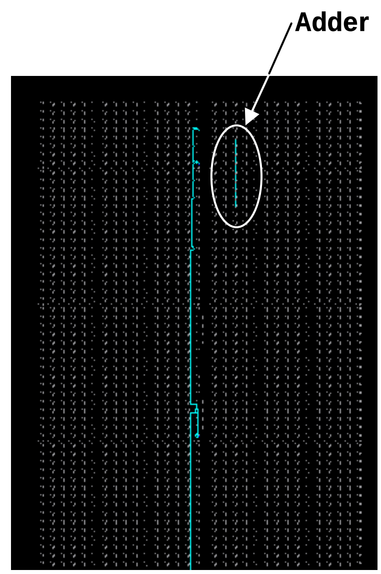

Figure 10: 32-bit Adder Generated Under xc4vfx20-11ff672 Parameters

Multipliers

Design Summary

Logic Utilization	8-bit Multiplier Used	16-bit Multiplier Used	32-bit Multiplier Used	64-bit Multiplier Used
# Slices	45	160	588	2222
# Slice Flip-Flops	32	64	128	256
# 4-Input LUTs	73	285	1104	4298
# bonded IOBs	33	65	129	257
Period [ns] (no constraints**)	5.601	7.876	11.088	15.651
Maximum Frequency [MHz]	178.54	126.97	90.2	63.892

Table 8: Multipliers' Space Requirements

Graphs

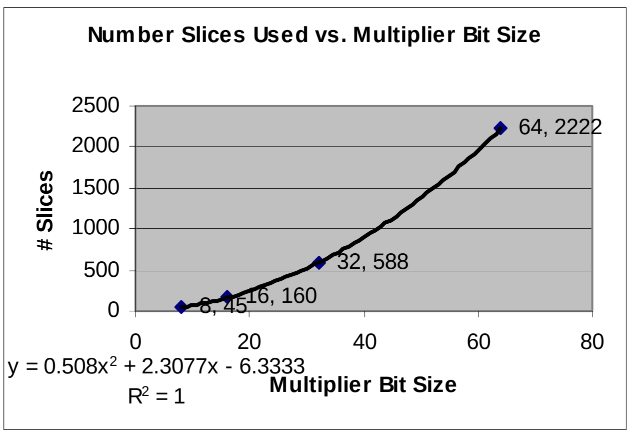

Figure 11: Graph of Number of Slices Used vs. Multiplier Bit Size

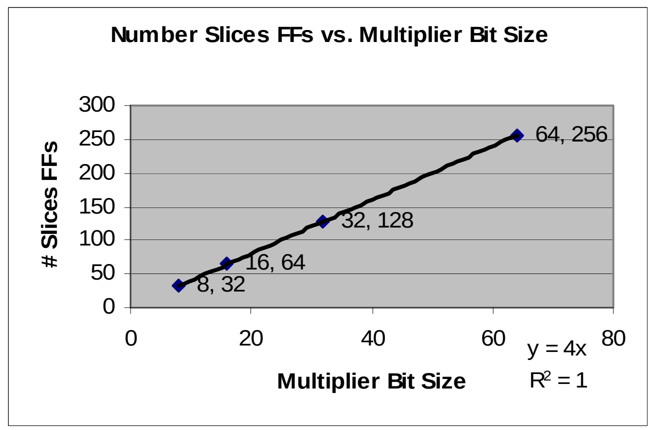

Figure 12: Graph of Number of Slice FFs Used vs. Multiplier Bit Size

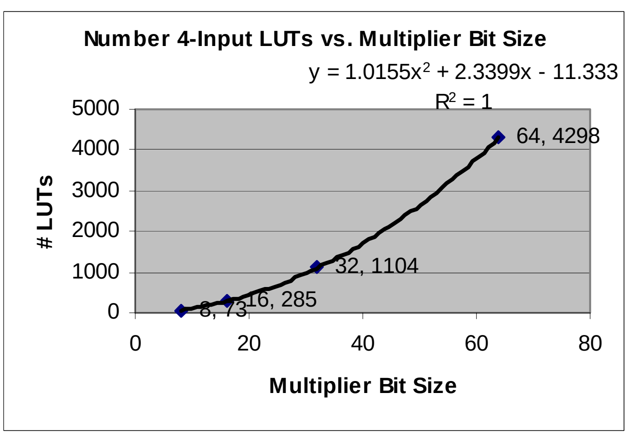

Figure 13: Graph of Number of 4-Input LUTs vs. Multiplier Bit Size

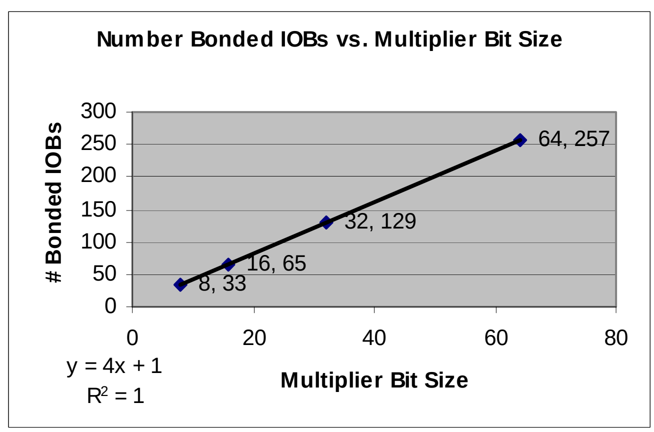

Figure 14: Graph of Number of Bonded IOBs vs. Multiplier Bit Size

From the graphs, it is apparent that the relationship for multipliers and the number of slices needed goes up nonlinearly. Thus, a polynomial regression (of order 2) has been used on the number of slices and the number of 4-Input LUTs.

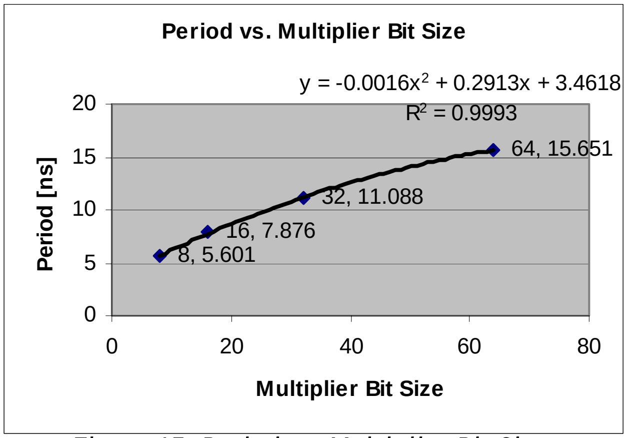

Figure 15: Period vs. Multiplier Bit Size

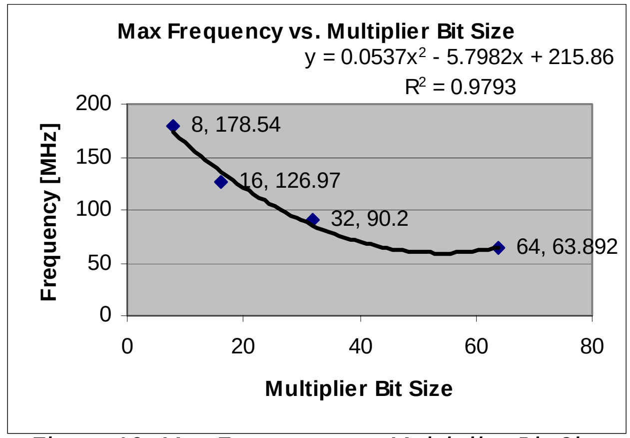

Figure 16: Max Frequency vs. Multiplier Bit Size

From the graphs, the minimum time needed for complete execution (one clock cycle) increases non-linearly. Note if we were to keep the constraint of 10 ns, then we would have to stop at:

$$y = -0.0016x^2 + 0.2913x + 3.4618$$
$$\rightarrow 10 = -0.0016x^2 + 0.2913x + 3.4618$$
$$\rightarrow x = 26 \text{ bits} \quad \text{or } x = 155 \text{ bits}$$

Since 155 bits does not apply, we know that we can only stop at the 26-bit multiplier.

**When constrained to a certain area, the amount of time the period increases by a maximum of 1 to 2 ns, but generally less than that.

Logic Distribution Tables

Logic Distribution (8-bit Multiplier)	Used	Available	Utilization
# occupied Slices	45	depends on board	varies
# Slices containing only related logic	45	45	100%
# Slices containing unrelated logic	0	45	0%

Table 9: 8-bit Multiplier Logic Distribution Statistics

Logic Distribution (16-bit Multiplier)	Used	Available	Utilization
# occupied Slices	160	depends on board	varies
# Slices containing only related logic	160	160	100%
# Slices containing unrelated logic	0	160	0%

Table 10: 16-bit Multiplier Logic Distribution Statistics

Logic Distribution (32-bit Multiplier)	Used	Available	Utilization
# occupied Slices	588	depends on board	varies
# Slices containing only related logic	588	588	100%
# Slices containing unrelated logic	0	588	0%

Table 11: 32-bit Multiplier Logic Distribution Statistics

Logic Distribution (32-bit Multiplier)	Used	Available	Utilization
# occupied Slices	2222	depends on board	varies
# Slices containing only related logic	2222	2222	100%
# Slices containing unrelated logic	0	2222	0%

Table 12: 64-bit Multiplier Logic Distribution Statistics

Similar to the adder logic distribution tables, this information shows that all the occupied slices contain only relevant logic.

Constraint Testing

Constraint testing for multipliers varied significantly from adders. The algorithms used to generate the wire mapping seem to use the area constraint specified more as a guideline than a requirement. In other words, all of the occupied slices were within the area but the wires were free to go about the board. The algorithm also seems to expand as much as possible to fill the specified area. This is evident when implementing 16-bit and 32-bit multipliers (only 32-bit is shown). Thus, every implementation I ran could never officially meet the constraints.

The following will be pictures of multipliers and the only the total area specified for constraint. The table below will showcase each multiplier, the area constrained (pictured in yellow), and my personal thought on if we adjusted manually what the space needed would be (pictured in white). All measurements are done in units of slices.

*64-bit multipliers were not implemented.

Multiplier	Constraint Dimensions	Total Area	Possibility	Total Area	Total Slices Needed	Corresponding Figure #
8-bit	2x26	52	2x24	48	45	18
8-bit	8x8	64	8x8	64	45	19
8-bit	30x48	1140	4x14	56	45	20
16-bit	6x40	240	6x34	204	160	21
32-bit	12x60	720	10x60	600	588	22
32-bit	30x126	3780	n/a	n/a	588	23

Table 13: All Constraint Data on Multipliers

Multiplier	"Best" Dimensions	Total Area	Reasoning
8-bit	4x14	56	Fitting within 48 slices is a stretch; 4x14 area looks reasonable on Figure 16
16-bit	6x34	204	If lucky, this is possible. 8x34 (272) is more feasible
32-bit	10x60	600	This sizing looks feasible. Otherwise, a safe estimate is 12x60 (720)

Table 14: "Best" Constraint Data on Multipliers

The data in Table 14 merely portrays my personal thoughts with respect the wiring layout generated by the Xilinx® software package. This data is only meant to give the reader an idea of the general shape and size of the multipliers.

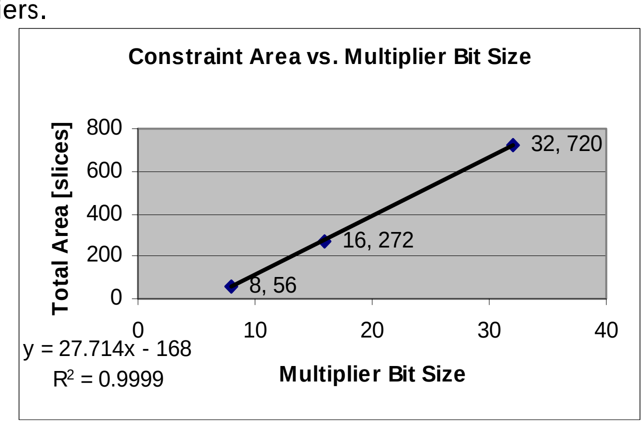

Figure 17: Graph of Area needed vs. Multipliers' Bit Size

From the graph, the area needed increases **linearly** with the multiplier bit-width.

Guide and Legend

Note that in the following pictures, one violet ‘grid’ sector contains four (4) slices – 2x2 slice sections. Also note that some sectors do not have slices at all (dividers on the FPGA board).

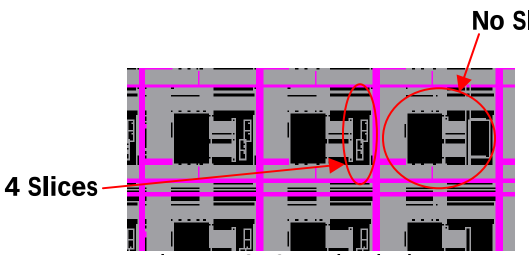

Figure 18: Standard Virtex-4™ FPGA Board

(white line) :	Personal Constraint
(yellow line) :	Specified Constraint (Sometimes not shown due to size)

Pictures of Implemented Multipliers

8-bit Multipliers

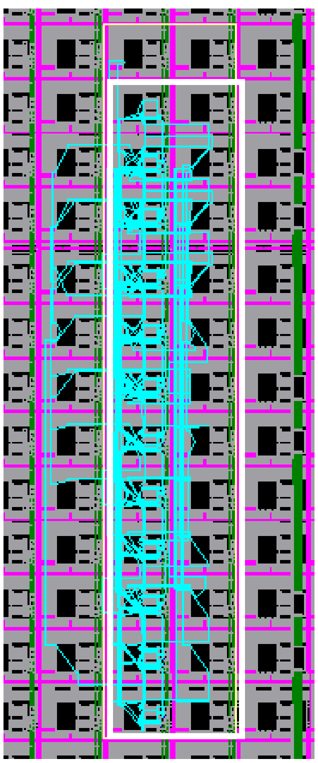

Figure 19: 8-bit Multiplier constrained at 2x26

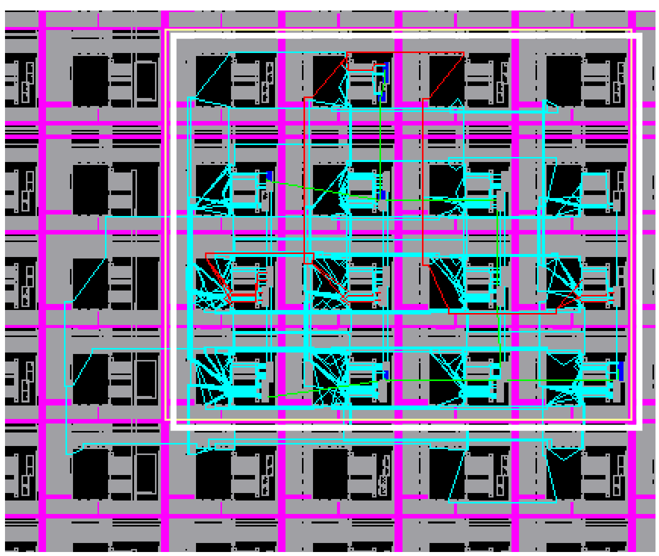

Figure 20: 8-bit Multiplier constrained at 8x8

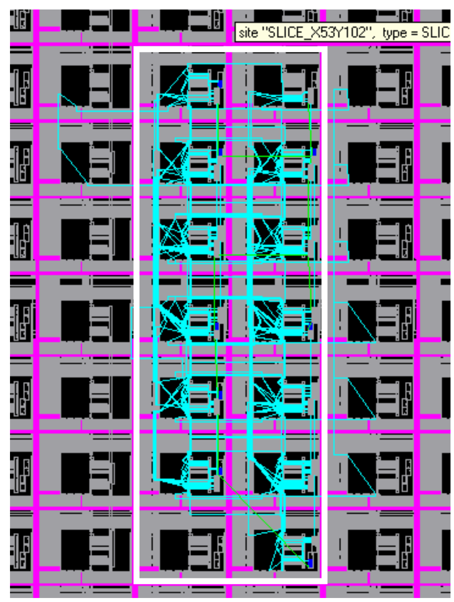

Figure 21: 8-bit Multiplier constrained at 20x38 (Actual constraint not shown)

16-bit Multiplier

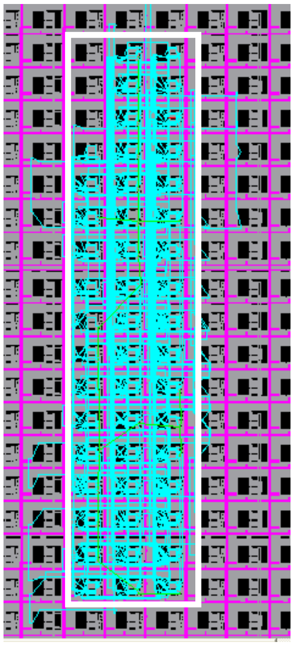

Figure 22: 16-bit Multiplier constrained at 6x40 (Actual Constraint not shown)

32-bit Multiplier

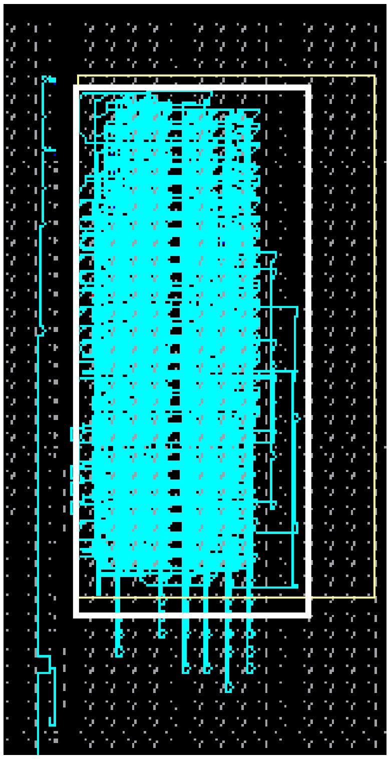

Figure 23: 32-bit Multiplier constrained at 12x60

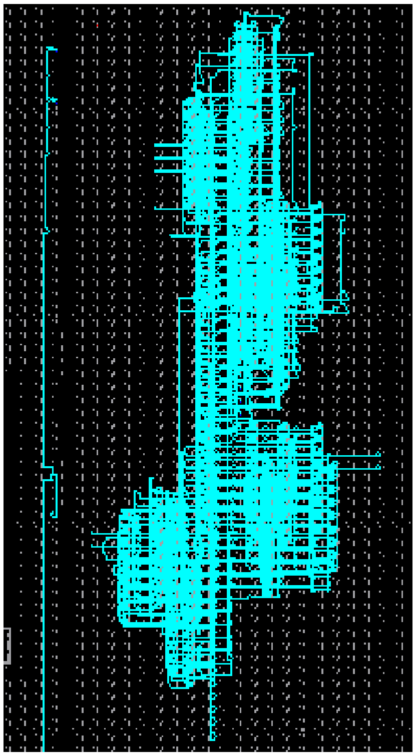

Figure 24: 32-bit Multiplier constrained at 30x126 (No constraints shown)

Pipeline Dividers

Design Summary

	8-bit Divider	16-bit Divider	32-bit Divider
Logic Utilization	**Used**	**Used**	**Used**
# Slices	116	482	1806
# Slice Flip-Flops	224	832	3200
# 4-Input LUTs	79	287	1087
# bonded IOBs	37	69	133
Period [ns] (no constraints**)	2.981	3.407	4.303
Maximum Frequency [MHz]	335.5	293.5	232.4

Table 15: Dividers' Space Requirements

Graphs

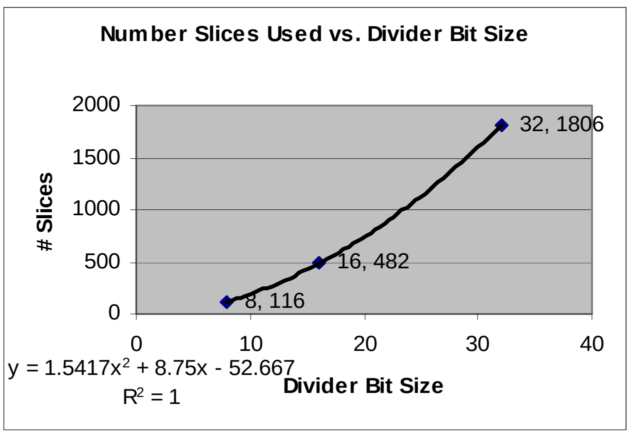

Figure 25: Graph of Number of Slices Used vs. Multiplier Bit Size

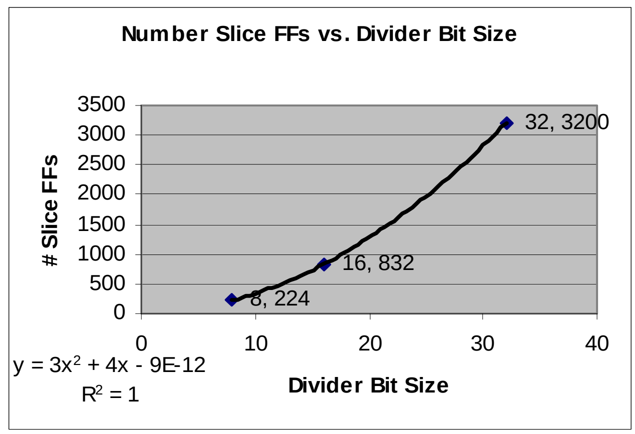

Figure 26: Graph of Number of Slice FFs vs. Divider Bit Size

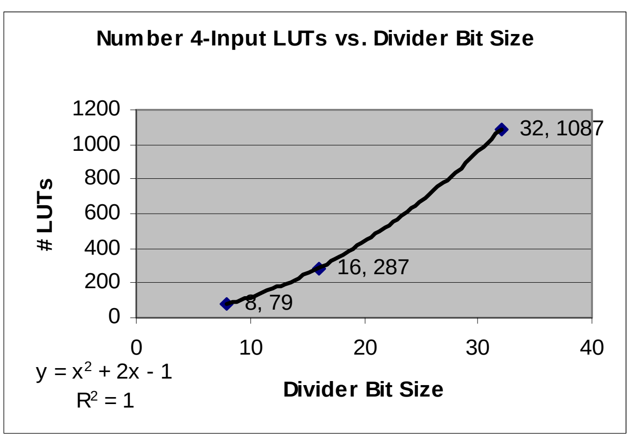

Figure 27: Graph of Number of 4-Input LUTs vs. Multiplier Bit Size

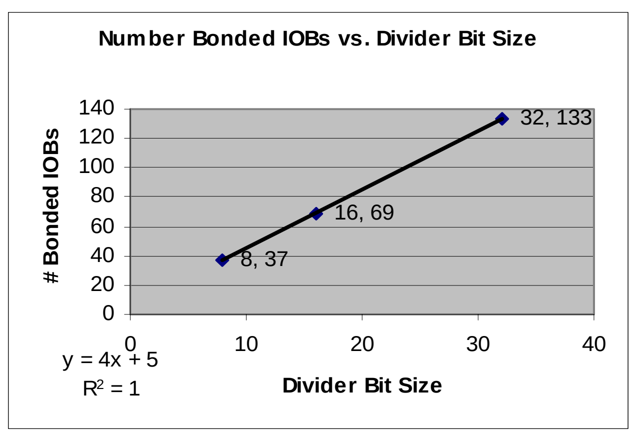

Figure 28: Graph of Number of Bonded IOBs vs. Divider Bit Size

Similar to multipliers, the above attributes increase non-linearly with the exception of bonded IOBs.

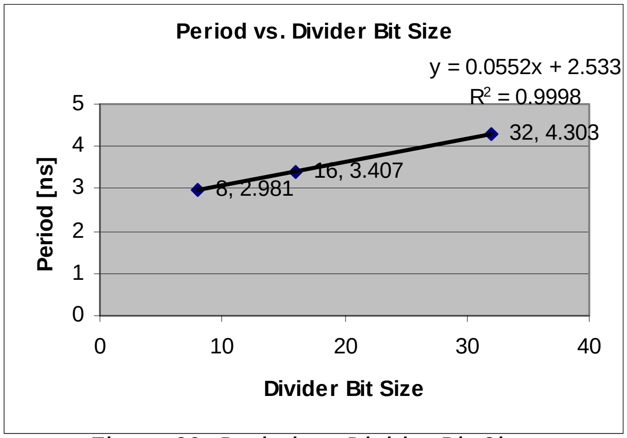

Figure 29: Period vs. Divider Bit Size

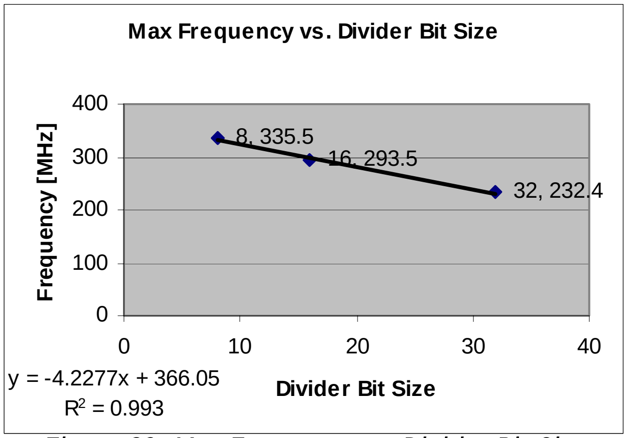

Figure 30: Max Frequency vs. Divider Bit Size

From the graphs, the time needed for complete execution (and the frequency) varies linearly with the number of bits on a divider.

**When constrained to a certain area, the amount of time the period increases by no more than 0.5 ns, usually less than that.

Logic Distribution Tables

Logic Distribution (8-bit Divider)	Used	Available	Utilization
# occupied Slices	116	depends on board	varies
# Slices containing only related logic	116	116	100%
# Slices containing unrelated logic	0	116	0%

Table 16: 8-bit Divider Logic Distribution Statistics

Logic Distribution (16-bit Divider)	Used	Available	Utilization
# occupied Slices	482	depends on board	varies
# Slices containing only related logic	482	482	100%
# Slices containing unrelated logic	0	482	0%

Table 17: 16-bit Divider Logic Distribution Statistics

Logic Distribution (32-bit Divider)	Used	Available	Utilization
# occupied Slices	1806	depends on board	varies
# Slices containing only related logic	1806	1806	100%
# Slices containing unrelated logic	0	1806	0%

Table 18: 32-bit Divider Logic Distribution Statistics

Similar to the adder and multiplier logic distribution tables, this information shows that all the occupied slices contain only relevant logic.

<u>Constraint Testing</u>

Constraint testing for dividers was similar to that of multipliers. The algorithms used to generate the wiring scheme used the area constraint only to place all the slices within and not the wires. Thus, even for a 'reasonable' constraint, wires have the potential to go outside the specified area.

Likewise, this algorithm also expands to fill as much as the space specified as possible. An example is shown in the 16-bit dividers section. Similarly to multipliers, the data in Table 20 only is to give the reader a general idea and should not be taken without first analyzing the corresponding figures.

*64-bit dividers are not available in the CORE Generator
**32-bit divider pictures not shown. If desired, please submit an email request.

Divider	Constraint Dimensions	Total Area	Possibility	Total Area	Total Slices Needed	Corresponding Figure #
8-bit	8x22	176	8x18	144	116	32
8-bit	4x38	152	6x40	240	116	33
8-bit	28x114	3192	8x16	128	116	34
16-bit	20x28	560	22x30	660	482	35
16-bit	22x128	2816	n/a	n/a	482	36
32-bit	46x46	2116	46x46	2116	1806	**

Table 19: All Constraint Data on Multipliers

Multiplier	"Best" Dimensions	Total Area	Reasoning
8-bit	8x16	128	Regarding Figure 33, I believe that if some slices were moved within the white box, this would be feasible
16-bit	22x30	660	It can probably be done in either 22x28 or 20x30 as well
32-bit	46x46	2116	Even though wires come out of the area, I believe that it is possible with some manual wiring manipulation

Table 20: "Best" Constraint Data on Dividers

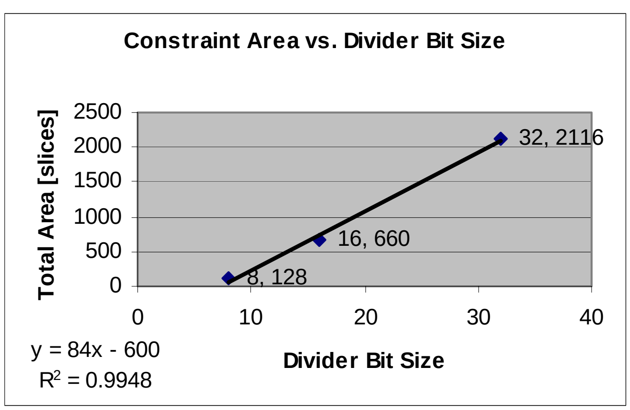

Figure 31: Graph of Area needed vs. Multipliers' Bit Size

From the graph, the area needed increases **linearly** with the divider bit-width.

Pictures of Implemented Dividers

8-bit Dividers

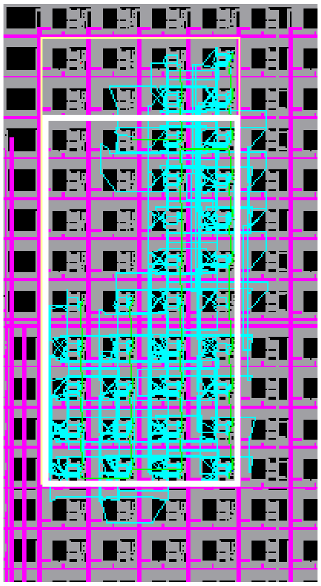

Figure 32: 8-bit Divider constrained at 8x22

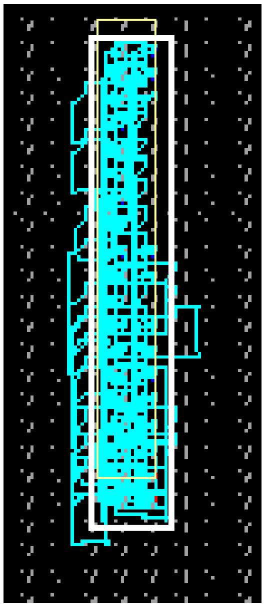

Figure 33: 8-bit Divider constrained at 4x38

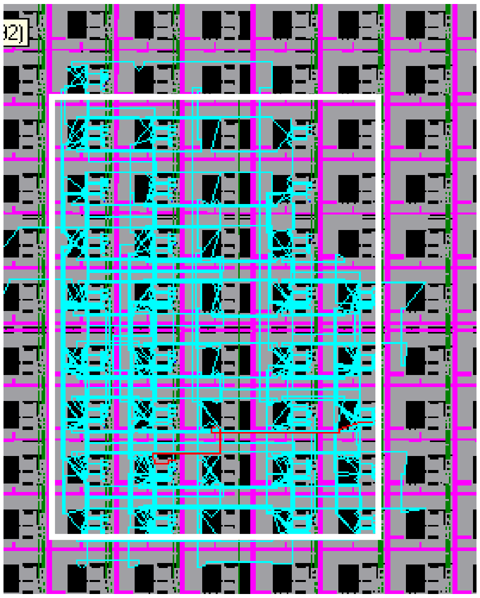

Figure 34: 8-bit Divider constrained at 28x114 (Actual constraint not shown)

16-bit Dividers

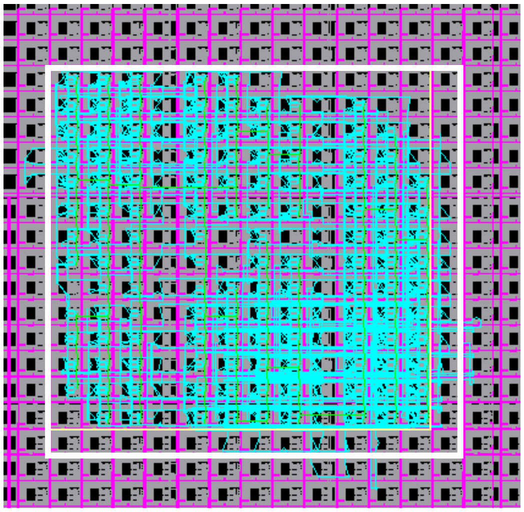

Figure 35: 16-bit Divider constrained at 20x28

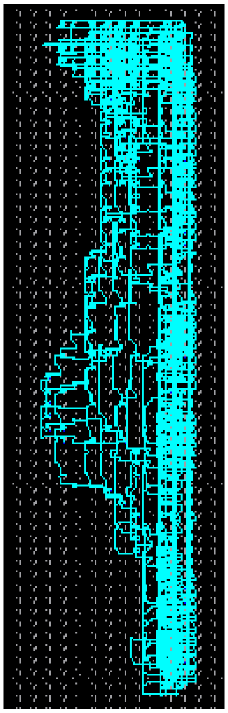

Figure 36: 16-bit Divider constrained at 22x128 (No constraints shown)