Summary and Analysis of Multiplier-Adders

Jin Hu

July 20, 2005

Introduction

A common component used on FPGA circuit boards is the multiplier-adder or the multiply-accumulator, more commonly known as the MAC. This merged module consists of two basic components: a multiplier whose output is then fed into one of the input ports of the adder or accumulator. These two single modules also have the same bit width; that is, a 16-bit multiplier feeds into a 16-bit adder to make a 16-bit MAC.

To implement a MAC, there are a few important aspects of performance and resource usage to consider. In particular, the following three are considered: area (in terms of slices), the number of frames needed during reconfiguration, and timing (clock period). In the following, the 8-bit, 16-bit, and the 32-bit MACs are analyzed with respect to the previously noted constraints. The discussion of the results will be in part with respect to the Xilinx® software package used and how Xilinx® performs when given different types of area constraints, including an unconstrained case.

Constraints and Considerations

To further clarify the terminology and the different constraints put forth on the three merged modules, the following explains in further detail what each constraint means and why it is important to performance.

Area

The amount of real estate needed on a board is perhaps the easiest to visualize. The physical size of the module must be small enough to fit onto a given FPGA board. As one module is typically not the only component used, the amount of room needs to be minimized. Note that this area refers to the implemented version and not only the physical composition of the MAC. A given MAC needs a number of slices on the board purely for logic – the variation occurs during the actual placing and routing of the module. The area is measured in terms of slices or configurable logic blocks [CLBs] where 1 CLB = 2 slices. Please note that all the areas specified are the constraints imposed. The actual area is slightly greater due to the routing of the wires during implementation.

Frames

During partial reconfiguration, the number of frames directly correlates to the number of bits that must be reprogrammed. Thus, the greater number of frames implies the more bits (and subsequently more time) needed to complete reconfiguration.

A note of clarification: $1\ \text{Frame} = 40\ \text{words} \times \frac{4\ \text{bytes}}{1\ \text{word}} \times \frac{8\ \text{bits}}{1\ \text{byte}} = 1280\ \text{bits}$

As mentioned above, area is measured in slices or CLBs. For the case of the Virtex-4™ FPGA boards, each frame spans a height of 16 CLBs (or 32 slices). As a rule of thumb, 1 CLB width-wise spans approximately 22-23 frames. If one bit must be reprogrammed in a frame, then the entire frame must be reprogrammed. Thus, it is advantageous to make area constraints in terms of the frame height (or every 16 CLBs). For example, Module A spans 2x8 CLBs and Module B spans 1x16 CLBs. Both modules have the same area (16 CLBs) but Module B needs half as many frames (22-23) as Module A (44-46).

To minimize area and frames, all of the area constraints are made in the following manner: the height of the area is determined first by the frame height and then the width is minimized afterwards.

Timing

All modules need a certain amount of time to perform a given duty. Specifically for the MACs, the dominating module is the multiplier, as it takes substantially more time than the adder. The optimal implementation of MAC implies that the amount of the time the complete module uses should be as small as possible. When using Xilinx, a tight timing constraint must be specified. Xilinx only checks if it meets the given time constraint and will not optimize further if the current timing constraint is met. For the MACs, the timings are constrained at the nanoseconds [ns] scale.

8-Bit Multiplier-Adder

In regards to bit width and size, the 8-bit MAC is the smallest component of the three MAC modules. The three different constraints mentioned above are explained in further detail below. The discussion is based on the data compiled in *Table 1*.

8-Bit Multiplier-Adder						
Area?	**Time (Actual) [ns]**	**x Length [slices]**	**y Length [slices]**	**Total Area [slices]**	**Frames Needed [frames]**	**Bit File [bytes]**
no	5.5 (5.486)	10	8	80	85	15,946
yes	5.5 (5.486)	2	30	60	36	7,454

Table 1: 8-bit MAC data

Area and Frames

According to the synthesis and implementation report as given by Xilinx®, an 8-bit multiplier-adder needs a constant 50 slices of real estate on a Virtex-4™ board. *Table 1* shows 2 trials – one without an specified area constraint and one with. The following graphs show the difference between an unconstrained module and a constrained module when implemented on Xilinx®. As noted, even though both implementations were given the same timing constraint, there is a significant difference between the unconstrained and the constrained modules in terms of area and frames needed. Note that the unconstrained module is shown as the 0th Frame Height to show comparison.

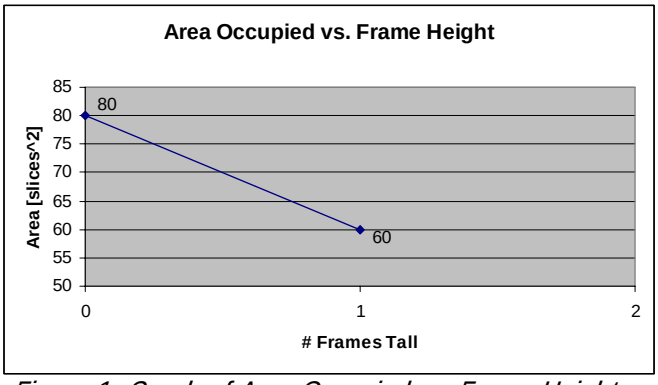

Figure 1: Graph of Area Occupied vs. Frame Height for 8-Bit MAC

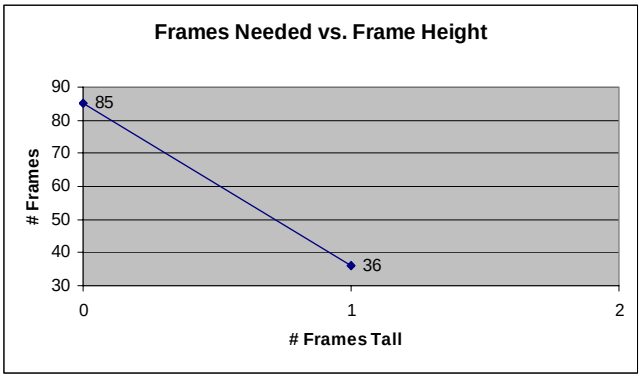

Figure 2: Graph of Frames Needed vs. Frame Height for 8-Bit MAC

From *Figure 1*, the unconstrained module needs approximately 80 slices to fit on the board. However, with an area constraint of 60 slices (1x15 CLBs), the area needed is lessened by a significant amount. The unconstrained 80 slices used relative to only 50 slices of logic needed implies that the placement of the logic slices is sparse and that wires spread across more board space. However, when constrained to only 60 slices, the module is more densely packed and uses less room.

From *Figure 2*, the unconstrained module, as it spans many more columns than the constrained module, needs many more frames during reconfiguration than the constrained module. Thus, it is advantageous to minimize the number of columns needed. By doing this with the constrained module, the number of frames needed during reconfiguration is drastically cut down.

A note of observation: The number of frames reported for the constrained module (1x15 CLBs) is 36 frames. This may seem like a discrepancy with the previously stated information that 1 CLB spans approximately 22-23 frames. This is due to the routing of wires when Xilinx® implements the adder. Within the 1x15 area are all the logic slices, but some of the wires cannot fit within this space. Thus, the wires extend out into a neighboring CLB, adding on more frames.

Timing

In general, the 8-bit MAC can operate at approximately 5.5 nanoseconds. As noted from *Table 1*, the actual running time for each synthesis is consistently close to 5.5 ns.

Bit File

The data given in *Table 1* specifies the size of the .bit file needed for reconfiguration. This data is gathered from the program bitgen which is included in the Xilinx® software package. As mentioned earlier, the number of frames directly correlates to the number of bits needed. However, the frames converted into bits are not the only bits needed for partial reconfiguration. This does not include the overhead needed, namely the addresses of the memory locations of where the bits need to go. Thus, it is also important to consider the amount of overhead needed based on the number of frames. The overhead needed for reconfiguration will be discussed further in detail at a later section.

16-Bit Multiplier-Adder

The next level of the three modules, the 16-Bit MAC needs more room than the 8-bit model. As before, the three constraints are discussed in detail with regards to the 16-bit multiplier-adder. The discussion is based on the data gathered in *Table 2*.

16-Bit Multiplier-Adder						
Area?	**Time (Actual) [ns]**	**x Length [slices]**	**y Length [slices]**	**Total Area [slices]**	**Frames Needed [frames]**	**Bit File [bytes]**
no	8 (7.991)	20	12	240	228	41,551
yes	8 (7.996)	8	26	208	104	18,459
yes	8 (7.981)	4	52	208	124	22,511
yes	8 (8.545)	2	102	204	157	32,339

Table 2: 16-Bit MAC data

Area and Frames

The area needed for the 16-bit multiplier-adder, as given by the synthesis report, is a constant 169 slices. *Table 2* shows syntheses of the same module implemented under different area constraints. The first column specifies the unconstrained model and the following rows go by frame height. The graphs below show the relationships between the frame height and area and number of frames needed. Similarly to *Table 1*, the unconstrained model is denoted as the 0^{th} frame height as comparison.

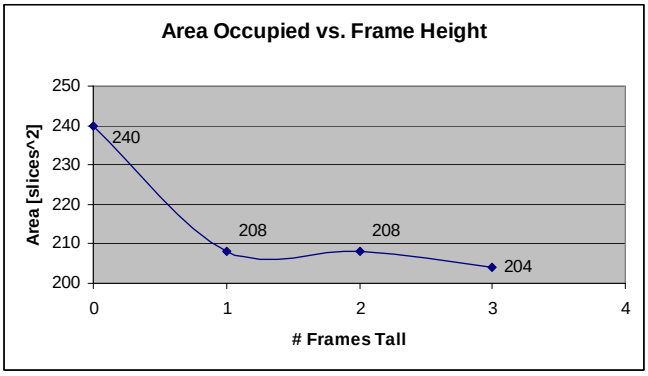

Figure 3: Graph of Area Occupied vs. Frame Height for 16-Bit MAC

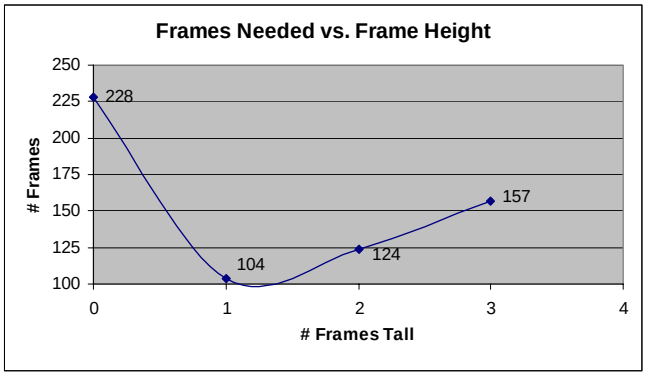

Figure 4: Graph of Frames Needed vs. Frame Height for 16-Bit MAC

From *Figure 3*, the constrained models (frame heights of 1, 2, 3) all have approximately the same area. However, when looking at the unconstrained model, there is significantly more area used. As the number of logic slices needed is only 169, using 240 implies the routing is not optimally done. Notice that the constrained models can achieve consistent area regardless of frame height. This then means that for this particular module, real estate is can be conserved regardless of the shape of the constraint. In accordance with *Table 2*, the timing constraints for each trial are consistent.

From *Figure 4*, the unconstrained model uses the most number of frames. Regarding the data from *Table 2*, it is evident that the number of columns spanned is much greater than any of the constrained models. Thus, when unconstrained, Xilinx® does not give an optimal implementation with regards to both area and the number of frames. Furthermore, after the initial unconstrained model, the graph shows a positive relationship between frame height and the number of frames needed for reconfiguration. This graph demonstrates that even though the areas remain approximately the same (see the corresponding points on *Figure 3*), the number of frames needed do not. From the graph, the minimum occurs when the constraint shape has a frame height of 1.

Timing

The 16-bit multiplier-adder runs typically at the 8 ns level. The time constraints imposed on the 4 different trials remain constant throughout. Notice that in the last row, the timing constraint given was 8 ns, but the actual time taken is approximately half a nanosecond greater. This is due to the shape of the area constraint. At a frame height of 3, the module width is very thin and narrow – the width of 1 CLB. For smaller bit-widths, this does not pose a problem but for larger modules, it sometimes is not feasible to meet the timing constraint based on the area constraint. Thus, as a first priority, Xilinx® first tries to meet the area constraint and then looks at the timing constraint.

Bit File

As in *Table 1*, *Table 2* also gives the corresponding .bit file sizes to each module synthesis. This data is used at a later section where the .bit files are discussed more in detail.

32-Bit Multiplier-Adder

The largest and most complex of the three MACs, the 32-bit multiplier-adder uses up the most resources than any of the previous models. The different constraints are discussed in detail in this section with regards to the 32-bit MAC. All discussion relates to the data presented in *Table 3*.

32-Bit Multiplier-Adder						
Area?	**Time (Actual) [ns]**	**x Length [slices]**	**y Length [slices]**	**Total Area [slices]**	**Frames Needed [frames]**	**Bit File [bytes]**
no	12 (11.996)	36	22	792	430	74,824
yes	12 (11.984)	32	26	832	386	67,392
yes	12 (11.907)	12	62	744	337	60,068
yes	12 (11.984)	8	92	736	326	56,732
yes	12 (11.991)	6	124	744	368	67,224

Table 3: 32-Bit MAC Data

Area and Frames

The logic slices needed by the 32-bit multiplier-adder comes to a total of 605 slices. Thus, the routing for this module will take more room than the previously discussed ones. In this case, due to the size of the module, it is possible to span up to 4 frame heights. It appears that it is possible to constrain the area into even more narrow strips but due to the limitation of the board used, the maximum frame height achievable is 4. As before, the unconstrained trial is denoted as the 0^{th} frame height in the following graphs.

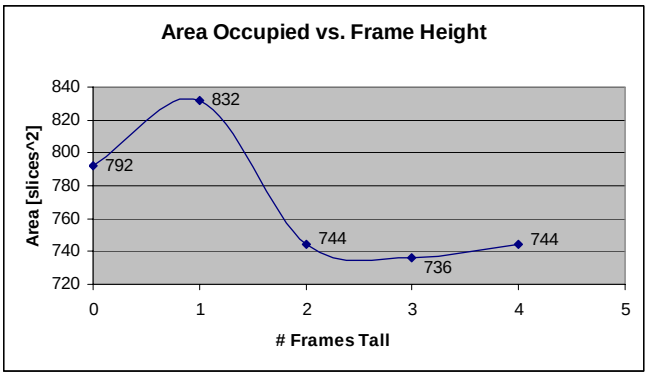

Figure 5: Graph of Area Occupied vs. Frame Height for 32-Bit MAC

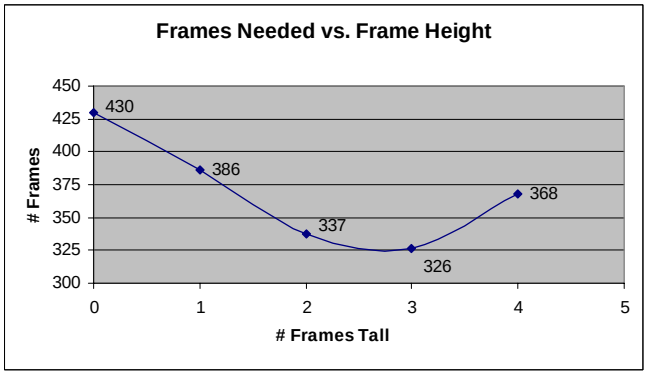

Figure 6: Graph of Frames Needed vs. Frame Height for 32-Bit MAC

From *Figure 5*, the unconstrained model does not show the same trend as the previous two models. That is, for this module, when unconstrained, the area size that Xilinx® has chosen is not very much different than the constrained models. Notice that unconstrained model is actually better than when the module is constrained to be approximately 1 frame tall. A noted observation about Xilinx®: when unconstrained, the module is placed at the bottom left hand corner and branches out from there. Thus, due to the amount of room needed for this particular module, Xilinx® produced a decent area size. For the rest of the frame heights starting at 2, the area remains approximately equal. This shows that the 32-bit MAC is easier to implement at these specific frame heights.

From *Figure 6*, the unconstrained model still uses the most number of frames, even though it takes up less area than a constrained model having a frame height of 1. As the frame height increases, the graph shows a parabolic behavior – a minimum appears where the constrained shape takes on a frame height of 3. It is also worthwhile, however, to note the actual number of frames needed. For the last few entries, the number of frames has a small amount of variation. Between the last three points, there is only a difference of 42 frames, around 10% of the total number of frames. This then implies that even though the best choice is when the height equals 3, a reasonable implementation can occur with any frame height greater than 1.

Timing
The 32-bit multiplier-adder can run consistently at around 12 ns. Note that for these specific frame heights, each one did not have problems passing the given time constraint. However, it is very likely that for area constraints that are taller and narrower to not meet timing constraints, similar to what occurred for the 16-bit multiplier-adder.

Bit File
As in *Table 1* and *Table 2*, *Table 3* also gives the corresponding .bit file sizes to each module synthesis for the 32-bit MACs. This data is used at a later section where the .bit files and the necessary overhead are discussed more in detail.

Timing Effects on Area and Frames

The designs of the implementations of MACs are dependant upon the three constraints mentioned earlier: timing, area, and the number of frames. As discussed, the number of frames for a given area can vary greatly and that the area should be minimized with respect to frame height. What may not have been made clear is how the timing constraint affects the implementation of the module and thereby the total area and frames needed. The timing constraint is used primarily for routing purposes – the lower the constraints, the more rigorous Xilinx® must work in order to meet the constraint. Given different timing constraints, Xilinx® will generate different routing schemes, thereby producing different area and number of frames needed. However, the area and the number of frames given for different timing constraints are insignificant in comparison to the "base numbers". To change the timing constraint is similar to moving a module to a different area of the board – the difference is very little. Thus the timing constraint does not affect the other two parameters.

Bit Files and Overhead

As mentioned in a few of the previous sections, the .bit files contain how many bits it take to configure a given module plus the addresses of where the bits need to go. The addresses are considered the overhead required in the process of partial reconfiguration and must be taken into account. The overhead can be determined from the number of frames needed and the .bit file size per module. The graph below show the relationship between the number of frames and the number of bits (measured in bytes) in the .bit files. All data used is gathered from all of the modules mentioned above in addition to other experimental data from other modules.

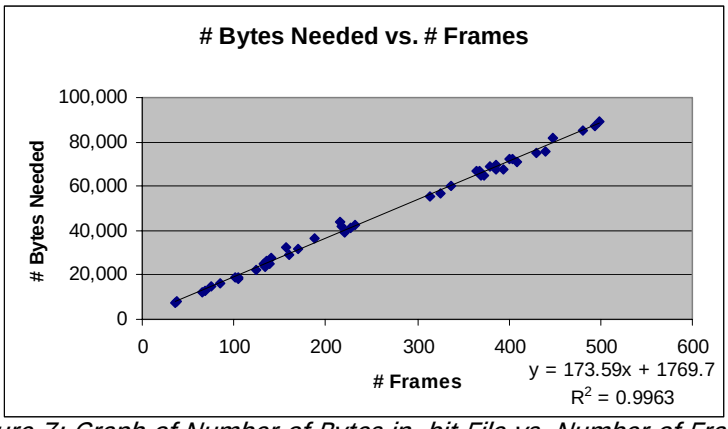

Figure 7: Graph of Number of Bytes in .bit File vs. Number of Frames

From *Figure 7*, the relationship is linear with a very strong correlation coefficient. The given linear regression equation describe that for every frame (denoted by x), it needs approximately 174 bytes plus a constant value. As mentioned earlier, 1 frame = 160 bytes. Subtracting this from 174, there is approximately 14 bytes of overhead per any given frame.