2005-04-07

Reducing Energy in FPGA Multipliers Through Glitch Reduction

Nathaniel Rollins

Michael J. Wirthlin
wirthlin@ee.byu.edu

Reducing Energy in FPGA Multipliers Through Glitch Reduction

Nathaniel Rollins and Michael J. Wirthlin,
nhr2@ee.byu.edu, and *wirthlin@ee.byu.edu*
Department of Electrical and Computer Engineering
Brigham Young University, Provo, UT. 84602

Abstract

While FPGAs provide flexibility for performing high-performance DSP functions, they consume a significant amount of power. For arithmetic circuits, a large portion of the dynamic power is wasted on unproductive signal glitches. Pipelining can be used to significantly reduce the unproductive power wasted in signal glitches. This paper presents a methodology for estimating the amount of power consumed by glitches and applies this methodology to non-pipelined, pipelined, and digit-serial multipliers. This glitch estimation is used to evaluate these multipliers using four energy metrics: energy per operation, energy delay, energy throughput, and energy density. Understanding the energy cost of arithmetic operators can be used to aid the designer or synthesis tool in the creation of energy efficient datapath circuits.

1 Introduction

The flexibility, reprogrammability, and high performance of field programmable gate arrays (FPGAs) make them an attractive architecture choice for many digital signal processing (DSP) applications. FPGA designs however, consume more power than application-specific integrated circuits (ASICs) [1] which makes them less attractive for wireless and handheld DSP applications. Consequently, energy consumption is becoming a critical design parameter for high-performance signal processing applications. Designers must increasingly consider the impact of power on FPGA signal processing systems.

The programmable nature of FPGA interconnect results in an interconnect structure with significantly larger loading than custom circuits. The signal buffers, pass transistors and other programmable switching structures significantly increase the capacitive load of signal nets over dedicated metal wires. This loading burden increases both the delay of interconnect as well as the power. Several studies suggest that the primary contribution in global system timing is the global interconnect [2]. It has also been shown that most of the power dissipated by an FPGA design occurs in the interconnect[3].

Because of the relatively large capacitive loading of programmable interconnect, the switching activity of individual signal wires will have a significant contribution to the dynamic power of the circuit. Much of the dynamic switching power can be wasted in unproductive circuit "glitches". Signal glitching refers to the transitory switching activity within a circuit as logic values propagate through multiple levels of combinational logic. While glitching is not unique to FPGAs, the relatively high capacitive loading of programmable interconnect places a much higher power cost to signal glitching for FPGAs. Previous studies have shown that power dissipation caused by glitching can makeup a significant amount of total dissipated power[4, 5].

An important technique for reducing FPGA power consumption is to reduce the amount of signal glitching within the circuit. Pipelining is one technique for reducing signal glitches. Previous studies have shown that pipelining can be used to reduce power by 90% [6]. A pipelined design has less logic between registers and therefore is less prone to glitching. Pipelining an FPGA design can come at little or no cost since flip-flops are included in every FPGA logic block and often go unused. Digit serial techniques, a form of pipelining, can also be used to

reduce signal glitching in arithmetic circuits [7].

This paper will describe a methodology for estimating the power associated with glitches within FPGA datapath circuits. This paper will quantify the dynamic glitch power, energy per operation, energy delay, energy throughput, and energy density of each multiplier used in the study. This methodology will be applied to a wide variety of multiplier circuits including non-pipelined multipliers, pipelined multipliers (with various pipelining levels), and digit-serial multipliers. The methodology presented and results from the study can be used by designers and high-level synthesis tools to properly select arithmetic operators based on energy consumption.

This paper will begin by describing the importance of dynamic transient power in FPGA circuits. A methodology will be described for carefully estimating the static, dynamic, and glitch dynamic power of datapath circuits. This methodology will be applied to a combinational multiplier to identify the unproductive glitching power component. Next, the energy measures used to evaluate power will be introduced. These will include energy per operation, energy delay, and energy density. The paper will apply this methodology to a number of non-pipelined multiplier circuits and highlight the significant contribution of glitches in circuit power. The paper will continue by applying this methodology on pipelined circuits and digit-serial circuits. These circuit types will be compared by evaluating the "operation energy" of each implementation style.

2 Dynamic Transient Power

Dynamic power makes up a large portion of the total amount of power consumed by an FPGA design. FPGA interconnect is largely responsible for dynamic power consumption. It has been shown that the interconnect of an FPGA accounts for the majority of the area on an FPGA chip and also accounts for the majority of the power dissipated by FPGA designs[8]. The amount of power consumed by the interconnect and clock tree can account for up to 86% of total dissipated power[8]. Unnecessary and unproductive use of FPGA interconnect will therefore be very costly in terms of power.

Unproductive interconnect activity is usually caused by glitching. Glitching refers to spurious signal transitions on interconnect lines caused by unequal logic or interconnect delays. For example consider the signal activity of an N-bit ripple carry adder. When new inputs arrive at the adder, all N-bit sums are computed simultaneously but the carry bits must ripple from the least significant bit up to the most significant bit. The most significant bit of the adder could switch N times due to this rippling. Only the final transition can be called a productive transition and so any other transitions are called glitches. The carry-out of the 32nd bit of a 32-bit carry chain will have on average 2x more useless transitions (glitches) than useful transitions per cycle and the sum output will have on average 1.5x more useless transitions per cycle[9]. Fast carry chains in FPGAs may significantly reduce the glitching caused by such an example, however FPGA designs are laden with glitches caused unequal line lengths leading to combinational logic block (CLB) logic.

Estimating the glitches that occur in FPGA designs is important in order to understand how much power is consumed by glitching. A static simulator cannot estimate dynamic signal activity and thus is not sufficient for accurate glitching power analysis of an FPGA design. A previous study demonstrated how static simulations of FPGA circuits can under estimate the circuit signal activity. In that study, the dynamic power estimation was 24% less than the actual power consumption [4]. In designs with larger amounts of glitching (such as a multiplier) the accuracy of such a static power model will be even worse.

An accurate power estimation should take signal glitching into account. To produce an effective power estimation, a back-annotated timing model is required in order to quantify net delays. A timing simulation which uses this back annotated timing model will then be able to simulate glitches so that the amount of glitching can be determined. A power analysis based on this timing simulation will produce a more accurate power consumption estimation.

ModelSim together with Xilinx's ISE tools can be used effectively to model dynamic transient signal activity and produce an accurate power consumption estimation. The Xilinx tools can be used to generate a back-annotated timing file which can be used by ModelSim to effectively model glitching through a timing simulation. The amount of glitching reported by ModelSim depends on the granularity chosen for the simulations. This study finds that a resolution of 100ps is sufficient. ModelSim simulations can be cap-

tured and recorded in a file which reports the switching activity of every net in a design. The switching activity of each net can be analyzed and tabulated into two categories: unproductive glitching transitions and productive signal transitions. Power estimation tools such as XPower[10] can then use this report to estimate the power consumption of the design.

The overall power consumption of a single circuit component can be broken into three categories: normalized static power, dynamic glitching power and the remaining dynamic power.

Static Power The static power of an individual circuit module is obtained by dividing the total static power of the device by the relative size of the circuit (i.e. # Circuit LUTs / Total LUTs). For circuits such as multipliers with large signal activity, this component is relatively small.

Dynamic Glitching Power The glitching power is obtained by counting the temporary signal glitches in the timing simulation. The percentage of signal glitches to total glitches is used to divide the dynamic power between glitching power and useful dynamic power.

Useful Dynamic Power The useful dynamic power is obtained by tabulating the "useful" transitions within the module. If the final value of a signal is different from the beginning of a clock cycle to the end, then a useful transition is the *last* transition that occurs, and all others are glitches. Otherwise, all transitions during the clock cycle were glitches.

This study will show that the dynamic glitching power and the remaining dynamic power make up the majority of the total power for arithmetic circuits such as multipliers. Reducing one or both of these parts is the key to reducing overall power consumption.

3 Multiplier Dynamic Glitch Power

This study uses multiplier designs to demonstrate the effects of glitching on total power consumption and on operation energy. A multiplier is a good design to demonstrate this due to it's large amount of net delays and varied net lengths which lead to a large number of glitches[8, 11]. The multiplier in Figure 1 shows that pipelining can be easily implemented by adding registers between multiplier stages. Additionally, a digit-serial multiplier based on the representation in Figure 1 can be easily created[7, 12]. A pipelined multiplier and a digit-serial multiplier are used in later sections to show how glitches can be reduced in order to lower the operation energy.

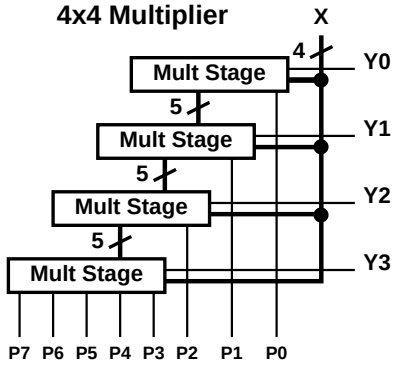

Figure 1: 4x4 multiplier.

For a non-pipelined multiplier the amount of glitching increases super-linearly as the size of the multiplier grows. To show this behaviour power estimates are performed on 4x4, 8x8, 16x16, and 32x32 multipliers. Random inputs are presented at the inputs for all simulations. The pie charts in Figure 2 show the breakdown of the total power into its constituent parts of static power, dynamic glitch power and remaining dynamic power. For all four multiplier sizes the amount of static power used by the design makes up less than 1% of the total power. Note how the percentage of dynamic glitch power increases as the bitwidth of the multiplier grows. In the case of the 4x4 multiplier glitching accounts for about 12% of the total power. As the multiplier grows to a 32x32 multiplier, the total power is dominated by glitching which accounts for 76% of the total power.

4 Reducing Glitch Power Through Pipelining

Pipelining a design is an intuitive way to reduce glitching. A pipelined circuit has less glitching due to the reduced amount of logic between registers. With

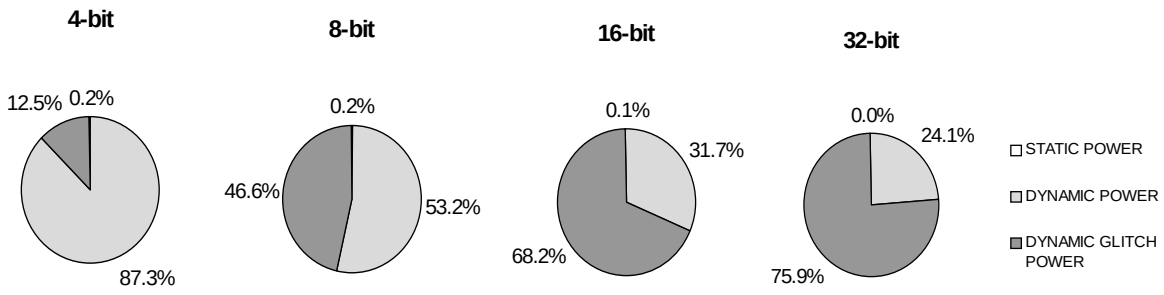

Figure 2: Breakdown of power constituents for a multiplier of various bitwidths.

less logic between registers, the amount of interconnect between registers is also reduced. Pipelining also causes long routing interconnect to be broken up by registers resulting in a smaller range of logic and interconnect delays. Consequently less glitching occurs. In many cases pipelining can be implemented with little additional cost since often many of the flip flops within the design's CLBs go unused. When pipelining can be used to reduce glitching, the amount of power consumed by a design is also reduced.

Implementing pipelining on the multiplier design previously presented shows how fewer glitches reduces overall power consumption and operation energy. Pipeline stages are strategically inserted in the multipliers of different bitwidths (4x4, 8x8, 16x16, and 32x32). For each multiplier, pipelining is gradually introduced until the multiplier is completely pipelined.

Figure 3 shows how glitching is reduced as pipeline stages are inserted. The graph reports the number of glitches as a percentage of the total signal transitions for each multiplier. The glitching percentage drops with the amount of pipelining introduced. The almost linear quality of the graph indicates that the advantages gained from pipelining pay off right up until the multiplier is fully pipelined.

As the amount of glitching goes down the amount of power consumed due to glitching is also reduced. Figure 4 shows how the dynamic glitching power lowers as the amount of pipelining increases. This logarithmic graph agrees with the intuition that as the amount of glitching goes down (see Figure 3) the amount of power consumption due to glitching also goes down. The logarithmic quality of the graph indicates that as pipelining begins to be applied to the multiplier there is a large initial pay-off in reduction of power due to glitching. After a certain point there is less power savings to be had by increasing the amount of pipelining.

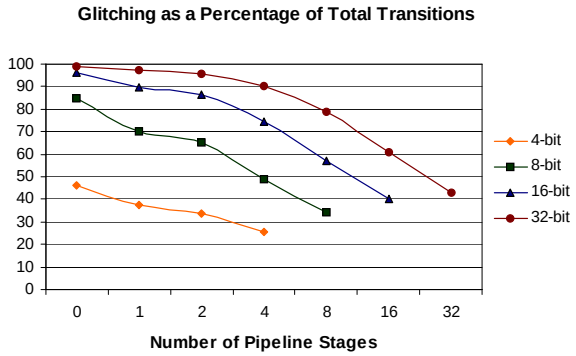

Figure 3: The amount of glitching as a percentage of total design transitions for multipliers of various widths and varying amounts of pipelining.

5 Reducing Glitch Power Through Digit-Serial Computation

Pipelining the stages of a multiplier has proven to be an effective way of reducing glitching (see Figure 3). The amount of pipelining available in the multiplier shown in Figure 1 is limited by number of multiplier stages. In other words an NxN multiplier can have a maximum of N pipeline stages. The

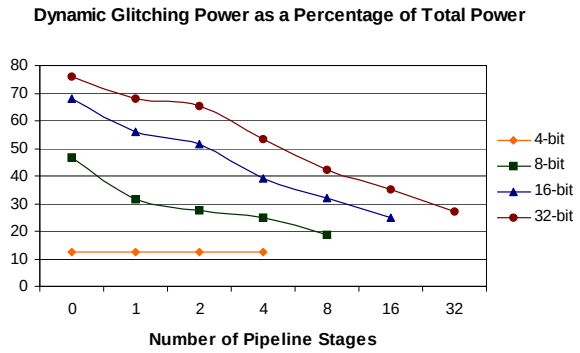

Figure 4: The amount of dynamic glitching power as a percentage of total power for multipliers of various widths and a varying amount of pipelining.

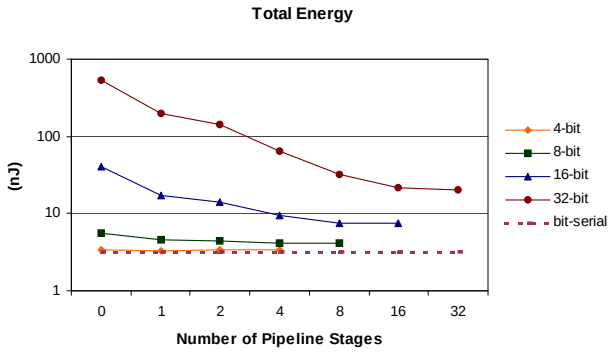

Figure 5: Total energy consumption (in nJ/cycle) of a multiplier of different widths and various amounts of pipelining compared to the total power consumption of a bit-serial multiplier with operands of different widths.

graph in Figure 3 suggests that glitching could be further reduced if additional pipelining was available. Additional pipelining is available in a digit-serial multiplier where pipelining is applied at a smaller granularity[12]. A bit-serial multiplier is pipelined at the bit level. It can reduce the amount of glitching to less than 1% of total signal transitions (for operands of any width). When compared to the percentages shown in Figure 4 for the pipelined multiplier, the less than 1% glitching achieved by the bit-serial multiplier is very impressive. With almost zero glitches, the amount of power consumed by glitching in a bit-serial multiplier approaches zero. This means that at least 98% of the consumed power is due to useful dynamic power.

Figure 5 compares the overall energy consumption of the pipelined multipliers with that of the bit-serial multiplier. The total amount of energy consumed by the bit-serial multiplier is almost independent of the bitwidth of its operands. For this reason, the total energy consumption of the bit-serial multiplier for any width is shown as a single line. The graph shows that a bit-serial multiplier with 32-bit operands consumes the same amount of total power as a fully pipelined 4x4 multiplier or 6x less overall power than a fully pipelined 32x32 multiplier.

With such an extreme amount of pipelining the minimum clock period of the bit-serial multiplier is reduced allowing for a faster clock rate. Unfortunately however, the pipelining in a bit-serial unit not only increases the latency but also the throughput of the design. Whereas the throughput of an NxN multiplier based on the design of Figure 1 is one product per cycle, the throughput of a bit-serial multiplier is one product per N*2 cycles. New operands are introduced to a bit-serial multiplier every N*2 cycles. Since the throughput of a design directly affects operation energy, even though a bit-serial multiplier consumes less overall power it may have a larger operation energy than a pipelined multiplier.

6 Operation Energy

Most studies quantifying the power consumption of FPGA circuits report on the average power of the overall circuit. For high-throughput arithmetic circuits, however, the primary concern is the amount of *energy* required to perform a specific arithmetic operation. Understanding the energy consumption of individual datapath operators will allow the designer or synthesis tool to choose operators that meet a specific throughput constraint while minimizing energy consumption.

This study will use the glitch estimation technique presented in the previous section to estimate the amount of energy consumed by a number of multiplier operators. Four energy parameters will be used: energy per operation, energy delay, energy throughput, and energy density.

6.1 Energy per Operation

Energy per operation quantifies the amount of energy required to complete a single operation of a specific circuit operator. This measure, reported as nJ, is more useful than an average power measure when comparing arithmetic circuits with different implementation approaches. Specifically, this measure allows us to compare the energy efficiency of multi-cycle operators with single-cycle operators.

Circuit energy can be computed by integrating the circuit power consumption over time ($E = \int Pdt$). Assuming constant power consumption, the energy can be estimated by multiplying the average power of the circuit operator by the amount of time to complete a single operation. For single-cycle operators, the energy per operation is simply the average power of the circuit times the clock period, or $E_{cycle} = P \cdot t_{clk}$.

The pipelined multiplier is an example of a circuit with a single-cycle operation. The energy per operation for the pipelined multipliers is shown in Figure 6. Energy per operation is affected by glitching power. Just as the graph of power consumption in Figure 4 drops logarithmically, Figure 6 shows that energy per operation also drops logarithmically. The graph also shows that as the width of the multiplier increases, energy per operation grows exponentially.

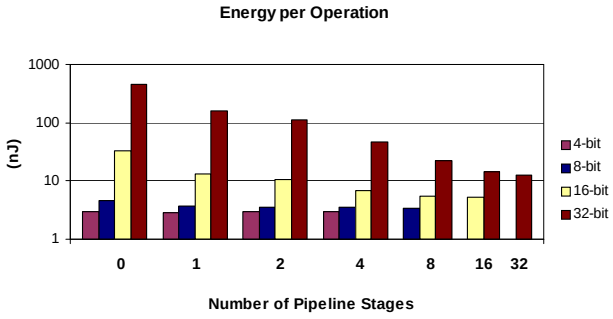

Figure 6: The energy per operation (in nJ) for a multiplier of different widths and various amounts of pipelining.

The energy per operation of multi-cycle circuit operators must take into consideration the number of clock cycles required to perform the operation. For a non-pipelined multi-cycle operator, the energy per operation is $E_{op} = P \cdot t_{clk} \cdot n$ where n is the number of clock cycles required to perform the operation.

Pipelined operators overlap the computation of several discrete operations in a single clock cycle. The energy consumed by a pipelined operator is shared among the various instances of the operator in the pipeline. Thus for pipelined operators, rather than considering the number of cycles required to perform the operation (n), the interval between successive initiations of the pipeline operation is considered (δ). The energy for a single computation is computed as $E_{op} = P \cdot t_{clk} \cdot \delta$. For $\delta = 1$ (which is the case for the pipelined multipliers), the operator is fully pipelined (i.e. a new operation can be initiated every clock cycle) and the energy per operation is $E_{op} = P \cdot t_{clk}$.

The bit-serial multiplier is an example of a circuit with multi-cycle operation. The bit-serial multiplier does not have a $\delta = 1$ like the pipelined multipliers, but has $\delta = N * 2$; where N is the bitwidth of the bit-serial multiplier operands. Thus, despite its much lower overall power consumption (Figure 5) in general the bit-serial multiplier has a large energy per operation.

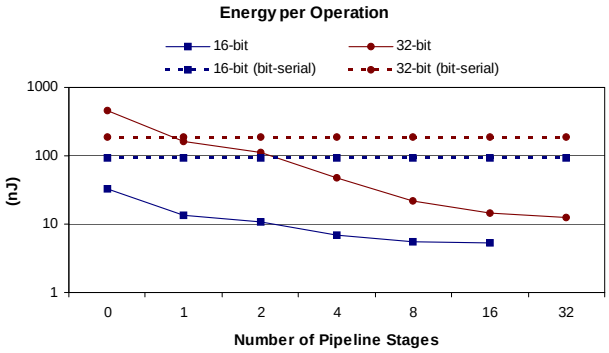

Figure 7: Energy per operation (in nJ) of a multiplier of different widths and various amounts of pipelining compared to the energy delay of a bit-serial multiplier with operands of different widths.

The energy per operation of the bit-serial multiplier is compared to the pipelined multipliers in Figure 7 (16-bit and 32-bit operands only). The x-axis of the graph is irrelevant to the bit-serial plots since the bit-serial multiplier cannot be pipelined any further. For this reason the bit-serial energy per operation plots show up as horizontal lines. The graph shows

that the energy per operation of the bit-serial multiplier is always greater than the pipelined multiplier except for one case of the 32x32 pipelined multiplier. This one case occurs when the 32x32 multiplier is non-pipelined. Having to take δ into consideration causes the large energy per operation for the bit-serial multiplier. For 32-bit operands, the bit-serial multiplier requires $\delta = 64$. Notice the similarity of the graphs in Figure 5 and Figure 7. The value of the bit-serial line in Figure 5 (in nJ) is multiplied by 32 in Figure 5 for the 16-bit operand bit-serial multiplier and by 64 for the 32-bit operand bit-serial multiplier. In general the energy per operation of a bit-serial multiplier is lower than the energy per operation of a pipelined multiplier.

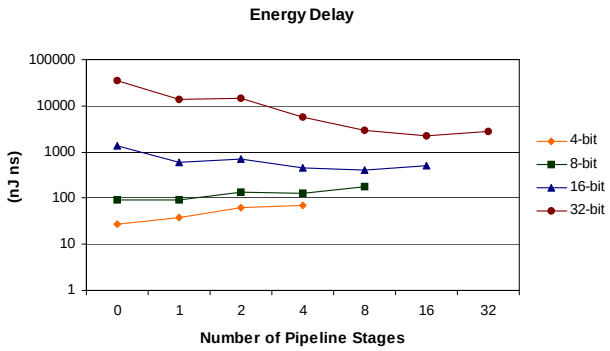

Figure 8: Energy delay (in nJ·ns) for a multiplier of different widths and various amounts of pipelining.

6.2 Energy Delay

Energy delay is a related measure that combines the energy efficiency and speed of an operator into a single parameter [13]. This measure is frequently used to balance the trade-off between reducing energy and increasing circuit speed. The energy delay of a circuit operator is computed as $E_{delay} = E_{cycle} \cdot \lambda$, where λ is the latency of a single computation of the operator. The latency of the operation is $\lambda = t_{min} \cdot n$ where n is the number of clock cycles required to complete a single operation and t_{min} is the minimum clock time period of the circuit.

The minimum clock time period of a circuit decreases as the amount of pipelining increases. This smaller clock period provides greater throughput for designs with single-cycle operations such as the pipelined multipliers. However, as we increase the amount of pipelining the latency increases (n increases), potentially increasing the energy delay. Thus, as pipelining is increased the energy delay could increase due to an increased latency, but the energy delay could also be reduced due the the reduced amount of glitching.

Figure 8 shows how an increased amount of pipelining could either decrease or increase energy delay. In designs which have a large percentage of glitching (see Figure 2) such as the 32-bit multiplier, increasing the amount of pipelining decreases the energy delay. In designs with less glitching, energy delay rises as pipelining is increased.

The trade-off of larger latency for less glitching is accented with the bit-serial multiplier. The bit-serial multiplier consumes almost no energy due to glitching, but relative to the pipelined multipliers requires a large number of cycles to complete an operation. The graphs in Figure 9 compares the energy density of the bit-serial multiplier with 16-bit and 32-bit operands (where $n = 32$ and $n = 64$ respectively) with the pipelined multipliers (where $n =$ pipeline depth). Due to the large amount of glitching in the 32-bit pipelined multiplier, the bit-serial multiplier has a lower energy delay. As the size of the pipelined multiplier goes down, the energy delay of the pipelined multipliers goes lower than the energy delay of the bit-serial multiplier.

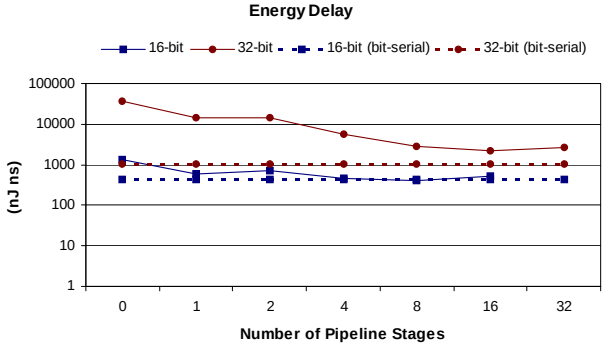

Figure 9: Energy delay (in nJ·ns) of a multiplier of different widths and various amounts of pipelining compared to the energy delay of a bit-serial multiplier with operands of different widths.

6.3 Energy Throughput

The energy throughput parameter is similar to energy delay in that it combines energy efficiency and circuit speed into a single parameter. Energy throughput differs from energy delay in that it does not take into account the number of clock cycles needed to complete a single operation (n), but rather it considers the number of clock cycles between successive operation initiations (δ). In most cases only one or the other is used. Energy throughput is calculated as $E_{thput} = E_{cycle} \cdot t_{min} \cdot \delta$, where t_{min} is the minimum clock time period of the circuit and δ is the interval between successive initiations of the pipeline operation.

Sometimes it is more useful to consider throughput energy rather than energy delay. In some designs a pipelined operation is used successively in such a way that a new operation can be initiated on each clock cycle ($\delta = 1$). In these cases, energy throughput might be a more desirable parameter than energy delay. Energy delay is more useful for designs which require an operation to complete before a new one can be initiated ($\delta = pipedepth$).

Where increasing pipeline depth raises energy delay (due to an increased latency), it lowers energy throughput. Pipelining benefits energy throughput in two ways: it reduces glitches which lowers overall energy consumption, and it reduces the minimum clock period - providing a greater throughput. Figure 10 shows how pipelining greatly benefits energy throughput. Energy throughput (in nJ·ns) is displayed on a logarithmic scale therefore over a range of different pipeline depths it appears linear. The graph shows how critical the first stages of pipelining are. Implementing only a single pipeline stage in the 32x32 multiplier reduces the energy throughput by an order of magnitude.

Energy throughput for the bit-serial multiplier is the same as energy density since $n = \delta = N * 2$ where N is the size of the bit-serial multiplier operands. Despite its low power consumption (see Figure 5) the bit-serial multiplier tends to have a large energy throughput and energy delay. The bit-serial multiplier benefits from having almost no glitching and a smaller clock period than the pipelined multipliers. However, it suffers from having a large δ value. In almost every case, this large δ value outweighs the advantages of a low minimum clock period and minimized glitching. Figure 11 compares the energy

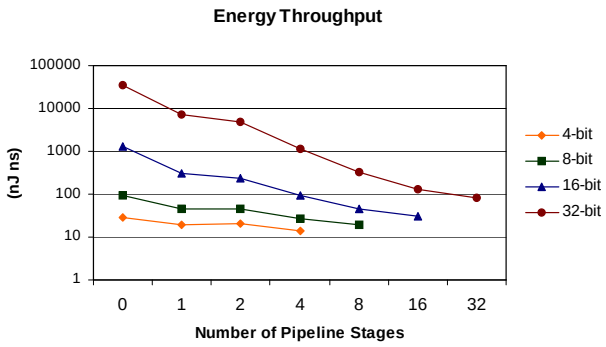

Figure 10: Energy throughput (in nJ·ns) for a multiplier of different widths and various amounts of pipelining.

delay of the bit-serial multiplier with the pipelined multipliers (16x16 and 32x32 only). It shows that in most cases the energy throughput of the bit-serial multiplier is larger than the energy throughput of the pipelined multipliers.

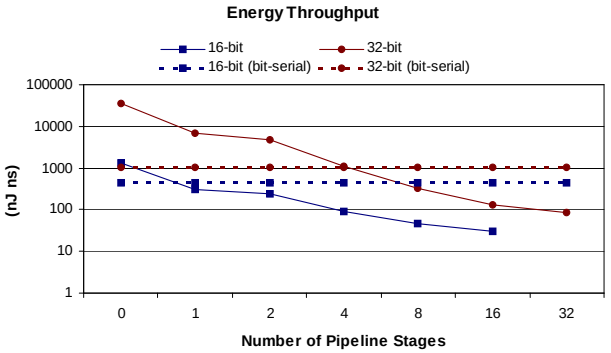

Figure 11: Energy throughput (in nJ·ns) of a multiplier of different widths and various amounts of pipelining compared to the energy delay of a bit-serial multiplier with operands of different widths.

6.4 Energy Density

The final energy parameter used in this study is energy density, $E_{density}$. This parameter normalizes the amount of energy required to perform a single operation to the number of logic resources used by the circuit. This parameter can be calculated as $E_{density} = E_{cycle}/A$, where A is the "area" of the

circuit. It is important to note that the amount of dynamic energy consumed by a circuit module is not necessarily linearly related to the size of the circuit. As described earlier, the primary contributer to dynamic power is signal glitches. Since circuit size affects energy density, large circuits that reduce glitching through pipelining may have a lower energy density than smaller pipelined circuits.

Figure 12 shows the energy density for the pipelined multipliers. From the graphs in this figure it may appear that energy density is not directly related to circuit size, but multiplier size does impact energy density through average net activity rate. The 4x4 multiplier has the smallest area but the largest energy density. The 32-bit multiplier has very high energy density when it is non-pipelined, however as pipelining is introduced the energy density drops until it reports the lowest energy density.

The energy trends shown in Figure 12 are the result of glitching and average net activity rate. As expected, glitching is reduced as pipelining is introduced, lowering the overall energy consumption. The 32-bit multiplier demonstrates this trend. However, the size of the multiplier also affects the energy density. The nets of a smaller multiplier have a higher average activity rate than a larger multiplier, resulting in larger energy densities for smaller multipliers.

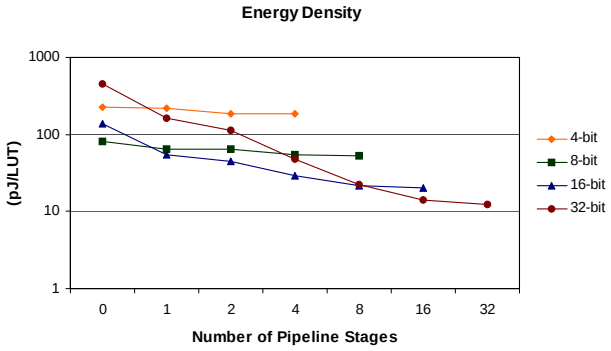

Figure 12: Energy density (in pJ/LUT) for a multiplier of different widths and various amounts of pipelining.

The graph in Figure 13 compares the energy density of the bit-serial multiplier with that of the pipelined multiplier (16x16 and 32x32 only). The graphs in this figure show that despite being a circuit with a multi-cycle operation, the bit-serial energy delay is often lower than the energy delay of the pipelined multipliers. Whereas the area of the pipelined multipliers has no direct correlation with its energy density, the energy density of the bit-serial multiplier is inversely proportional to its area. The bit-serial multiplier with 4-bit operands has the smallest area and the highest energy density. Conversely, the bit-serial multiplier with 32-bit operands has the largest area and the smallest energy density.

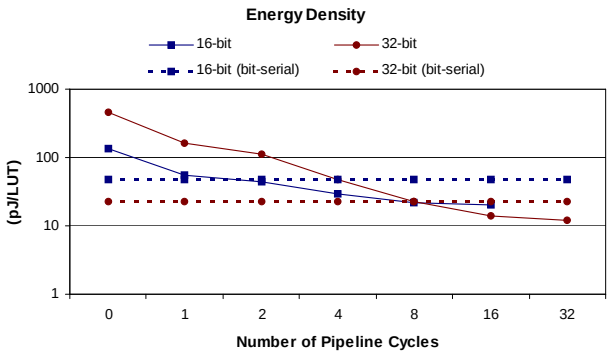

Figure 13: Energy density (in pJ/LUT) of a multiplier of different widths and various amounts of pipelining compared to the energy delay of a bit-serial multiplier with operands of different widths.

7 Conclusion

This paper presents a methodology for estimating the glitches of FPGA circuits and uses this methodology to determine the amount of power wasted in glitching. This paper shows that the majority of dynamic power in non-pipelined multipliers is consumed by glitches. The glitch power can be reduced by pipelining the circuit. Results in the paper show an exponential decrease in glitch power by increasing the pipelining depth.

Several energy related measures were introduced and used to provide better tools for comparing the energy requirements of different implementation approaches. Single cycle and multi-cycle circuit implementations can be compared by estimating the energy per operation rather than the average power consumption. This paper demonstrates that while multi-cycle bit-serial operators have significantly lower average power, the energy per operation is higher than deeply pipelined parallel approaches.

Energy delay, energy throughput and energy density were also used to compare the multiplier implementation approaches. Energy delay is used to balance the trade-off between energy reduction and operator latency. Increasing the pipelining depth exponentially reduces glitch power, but also increases latency. The amount of pipelining which produces the best trade-off depends on the bitwidth of the multiplier. In general, as the bitwidth increases, the amount of pipelining producing the lowest energy delay also increases.

The metrics and estimates generated in this paper will be used as part of a high-level datapath synthesis tool. This tool will use energy estimates of the various multiplier implementation approaches to select the proper multiplication approach. High-power low-pipelined circuits will be selected for datapath circuits with tight latency constraints while low-power, highly pipelined circuits will be selected for latency tolerant datapath circuits.

References

[1] P. Zuchowski et al. A hybrid ASIC and FPGA architecture. In *Proc. ICCAD*, pages 187–194, 2002.

[2] Verghese George, Hui Zhang, and Jan Rabaey. The design of a low energy FPGA. In *International Symposium on Low Power Electronics and Design 1999. Proceedings*, pages 188–193, August 1999.

[3] Juergen Becker, Michael Huebner, and Michael Ullmann. Power estimation and power measurement of Xilinx Virtex FPGAs: Trade-offs and limitations. In *Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI'03)*. IEEE Computer Society Press, 2003.

[4] Nathan Rollins. SLAAC1V power user's manual. Technical report, Department of Electrical and Computer Engineering, Brigham Young University, 2004.

[5] Anand Raghunathan, Sujit Dey, and Niraj K. Jha. Register transfer level power optimization with emphasis on glitch analysis and reduction. *IEEE Transactions on Computer-aided Design of Intergerated Circuits and Systems*, 18(8):1114–1131, August 1999.

[6] Steven J.E. Wilton, Su-Shin Ang, and Wayne Luk. The impact of pipelining on energy per operation in field-programmable gate arrays. In *Field-Programmable Logic and Applications. Proceedings of the 13th International Workshop, FPL 2004*, Lecture Notes in Computer Science, LNCS 3203, pages 719–728. Springer-Verlag, August 2004.

[7] Yun-Nan Chang, Janardhan H. Satyanarayana, and Keshab K. Parhi. Systematic design of high-speed and low-power digit-serial multipliers. 45(12):1585–1596, December 1998.

[8] Eric Kusse and Jan Rabaey. Low-energy embedded FPGA structures. In *International Symposium on Low Power Electronics and Design 1998*, pages 155–160, August 1998.

[9] Jeroen Leijten, Jef van Meerbergen, and Jochen Jess. Analysis and reduction of glitches in synchronous networks. In *European Design and Test Conference, 1995.ED&TC Proceedings*, pages 398–403, March 1995.

[10] Xilinx, Inc. *XPower Manual*.

[11] Suleman Sirri Demirsy, Andrew G. Dempster, and Izzet Kale. Power analysis of multiplier blocks. In *International Symposium on Circuits and Systems*, volume 1, pages I–297–I–300, May 2002.

[12] R. F. Lyon. Two's complement pipeline multipliers. *IEEE Transactions on Communications*, pages 418–425, April 1976.

[13] Ricardo Gonzalez and Mark Horowitz. Energy dissapation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, September 1996.