Team NAND's Digital Signal Processor

Kyle Powers, Dar-Eaum Nam,
Izak Baranowski, Jacob
Gravely, Min Je Chang
ECE 3663 – Spring 2012
University of Virginia
<kep4md, dan6ra, jb4er, jdg5ra,
mc4jx>@virginia.edu

ABSTRACT

In this paper, a digital processor signal (DSP) design by Team NAND is described. The innovations of the design are discussed, including its multiplexer topology, adder topology, adder-subtractor combination, and multiplier topology. Various metrics are discussed regarding the comparison between a Kogge-Stone parallel prefix adder, the results of the overall DSP, the metrics of adder-subtractor, and the findings of the Dadda multiplier. Extensions are provided for future Team NAND-PICo collaboration for DSP 2.0 and a conclusion is reached to effectively provide why Team NAND has the best DSP design.

1. INTRODUCTION

On February 2nd, 2012, Team NAND learned of the opportunity to win a large contract from Portable Instruments Company (PICo) to develop an embedded Digital Signal Processor (DSP) design in the FreePDK 45nm technology. We have fulfilled this request with the DSP described in this paper, utilizing a variety of high performance logic and area/power-saving components. Design aspects regarding delay, area, energy, and the PICo metric are addressed, while other innovations and extensions are investigated.

2. DESIGN DESCRIPTION

The basic structure of the PICo-requested DSP is represented below in Figure 1:

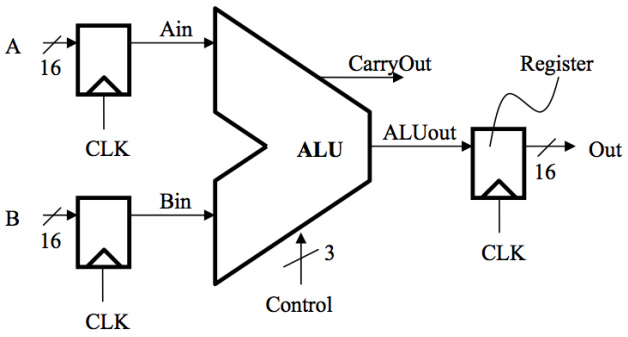

Figure 1. The DSP's basic structure as desired by PICo

Two 16-bit inputs A<0:15> and B<0:15> each feed a 16-bit register. The 16-bit outputs of each register serve as 16-bit inputs to the Arithmetic Logic Unit (ALU). Through the use of a 3-bit control line, the ALU outputs a 16-bit signal that feeds into a third 16-bit register. From there, the 16-bit signal is successfully read out from the register. A 1-bit CarryOut signal is also provided from the ALU. The ALU performs add (ADD), subtract (SUB), AND, OR, SHIFT, PASS A, and multiply (MULT) operations and by the implementation of an 8x1 multiplexer (MUX), is governed by the following table of 3-bit control line combinations:

Table 1. 8x1 MUX control line combinations for ALU output

ALU Function	3-bit Control	Description
NOP	000	No change at Out
ADD	001	Out = A + B
SUB	010	Out = A – B
SHIFT	011	Out = A << B
MULT	100	Out = A<0:7> * B <0:7>
AND	101	Out = A & B
OR	110	Out = A | B
PASS A	111	Out = A

Figure 2 below shows the top-level view of our design, illustrating the composition of the ALU:

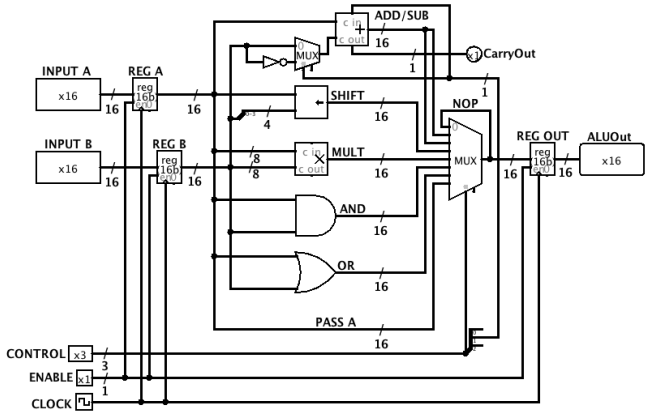

Figure 2. A top-level view of Team NAND's DSP

Of particular interest are the multiplexers, which were created using pass-gate transistor logic, and our combined adder-subtractor, which utilizes a high performance circuit topology. These, along with our multiplier, will be discussed at length in the following section.

3. INNOVATIONS

3.1 Multiplexers

For our MUXes, we chose to utilize pass-gate logic, as opposed to the standard MUX topology that requires eight 4-input AND gates and an 8-input OR gate. The pass gate topology offers many improvements, most notably a greatly reduced cost in area. The pass-gate topology uses only twenty-four transistors (including

the inverters used to invert the select line signals and to buffer the output) versus the ninety-eight transistors required in the gate-based topology. Another benefit is the reduced fanout to the select line inputs results in less power consumption. There are tradeoffs however, in order to compensate for the faults of the pass-gate topology, a buffer was created with two inverters that prevents the signal from weakening significantly. Figure 3 illustrates our MUX.

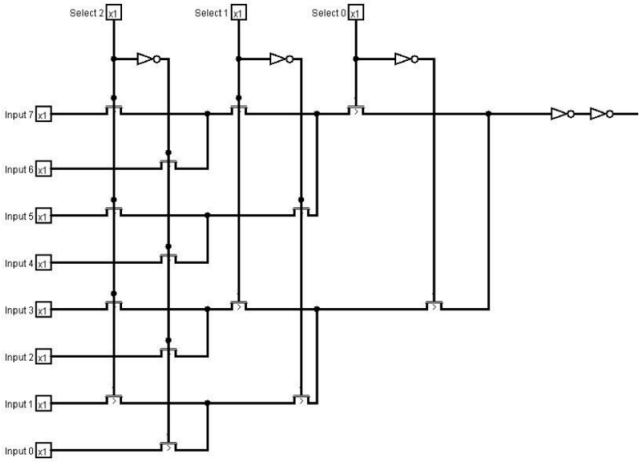

Figure 3. Team NAND's 8x1 MUX using pass-gate logic

3.2 Adder

For our adder topology, we chose to implement a high performance parallel prefix adder (PPA). In a PPA, a prefix operation is constructed that permits the computation of intermediate carries. In conjunction with generate and propagate signals, the prefix operator allows PPAs to obtain an advantageous latency of $O(log_2 N)$ instead of $O(N)$ (like in a Ripple Carry adder), where N is the word length. Figure 4 illustrates a tree graph of an 8-bit Kogge-Stone PPA:

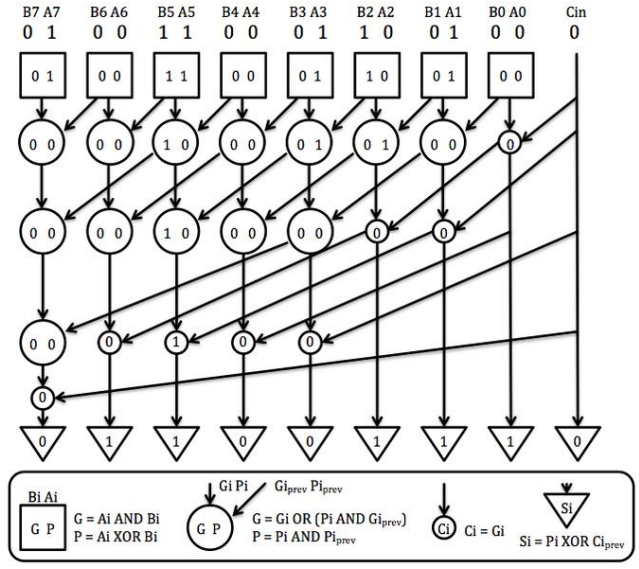

Figure 4. An 8-bit Kogge-Stone PPA tree graph

The attributes associated with a Kogge--Stone are low logic depth, high node count, and minimal fanout. While a high node count implies a larger area, the low logic depth and minimal fanout allow for faster performance. If PICo were to prefer metrics more along the lines of reduced area, a Brent-Kung PPA (Figure 5) could optionally be constructed, which with its minimum number of nodes and maximum logic depth, trades speed for reduced area.

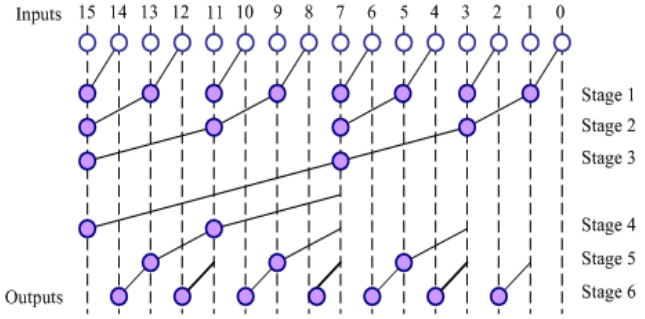

Figure 5. A 16-bit Brent-Kung PPA tree graph[1]

If a compromise between speed and area was sought after, a Han-Carlson (a hybrid Kogge-Stone/Brent-Kung PPA) could be implemented instead (Figure 6).

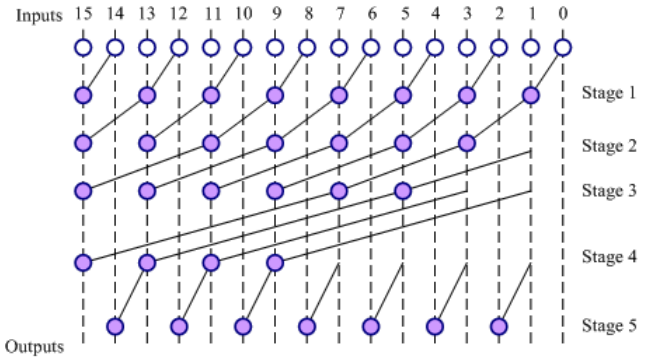

Figure 6. A 16-bit Han-Carlson PPA tree graph[1]

For comparison, Team NAND constructed a standard Ripple Carry adder and put our Kogge-Stone to the test. Our Kogge-Stone was twelve times as fast when exercising the critical path of both adders (.03ns verse .36ns). Presumably such an advantageous adder would consume equally as much current and power, but our results showed only a .05mA increase in current (1.14mA verse 1.09mA) and an approximately 3.5mW increase in power consumption (6.27mW verse 2.55mW).

3.3 Subtractor

For our subtractor topology, we reused the Kogge--Stone parallel prefix adder (PPA). What this implies is that because it is a high performance adder, utilizing it in a subtractor allows for very speedy subtractions. Creating a combined adder-subtractor increases delay slightly (.16ns in our tests), but saves tremendous area overall (435um verse 755um, an area saving of 57.6%) and reduces the power consumption associated with having a separate adder and subtractor operating simultaneously. Figure 7 illustrates how we utilized a 16-bit 2x1 MUX and a 16-bit inverter to create our adder-subtractor:

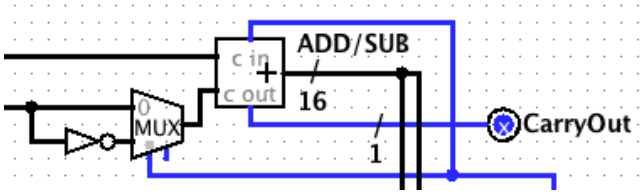

Figure 7. Team NAND's combined adder-subtractor

3.4 Multiplier

PICo requested an arbitrary function to be included with the DSP; thusly, Team NAND unanimously agreed that no ALU is complete without a multiplying function. The common go-to multiplier topology is an array multiplier that ANDs the inputs to create partial products, then uses adders to compute the final product. Unfortunately, due to this multiplying scheme, its functionality is hampered by an $O(log^2 n)$ computation time.

Fortunately, however, a Wallace tree multiplier serves as a much more efficient multiplication scheme. By reducing the number of partial products by two layers of full and half adders, and using a high performance adder to compute the final product, the Wallace tree multiplier achieves an advantageous computation time of $O(log\ n)$. However, Team NAND was not satisfied with this improvement and decided to go for an even more advanced topology called the Dadda multiplier.

The Dadda multiplier is similar to the Wallace multiplier, but is slightly faster for all operand sizes and requires fewer gates. What this translates to for PICo is a DSP with a high, if not the highest, performance multiplier, which keeps area and power in consideration by requiring fewer gates. Figure 8 illustrates the reduction scheme that a Dadda multiplier employs:

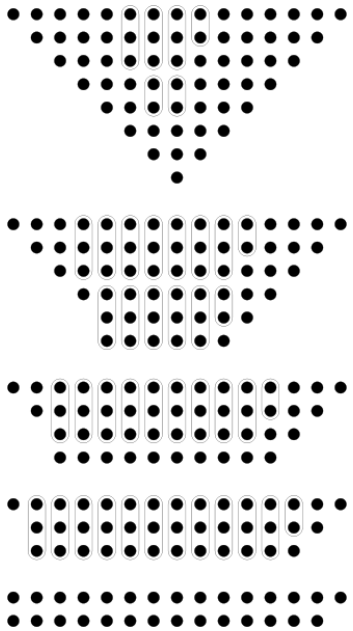

Figure 8. Dadda reduction on an 8x8 multiplier

3.5 Transistor sizing

All transistors were sized to match the resistance of a characteristic inverter. Team NAND knew from the start that the combined adder-subtractor would be the critical path (excluding the multiplier), which lends precedence to our decision to use a high performance adder topology. More than likely, the utilization of a slower adder such as Ripple Carry would require the use of logical effort[2] to size for delay, and while the same could apply to a parallel prefix adder, the inherent advantages of a PPA's structure makes resizing unnecessary, while still providing a hefty speed advantage. Not explicitly sizing for delay saves area and power.

4. RESULTS

This section will outline our results, including a table of our DSP's delay, area, energy, and metric figures, similar figures relating to our adder-subtractor, and lastly, comparable results regarding our multiplier.

4.1 DSP results

The following table outlines the delay, area, energy, and metric values for our DSP:

Table 2. Team NAND's DSP's metric values

Component	Value	Component	Value
Delay, D	196 ps	Energy, E	.963 nJ
Area, A	435 µm	Metric (D^2*A*W)	$8.21 *10^{-23}$ s^2*m*W

Figure 9 illustrates the critical path in our DSP (excluding our multiplier), which involves the subtraction function:

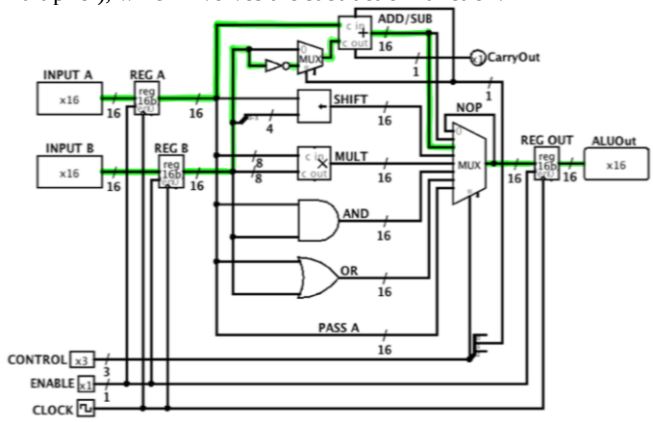

Figure 9. The DSP's critical path highlighted in green

4.2 Adder-subtractor results

The following table outlines the delay, area, energy, and metric for our adder/subtractor:

Table 3. Team NAND's adder-subtractor's metric values

Component	Value	Component	Value
Delay, D	193 ps	Energy, E	.153 nJ
Area, A	321 µm	Metric (D^2*A*W)	$9.48 *10^{-24}$ s^2*m*W

Figure 10 is an extension of Figure 9, illustrating the critical path of the 16-bit Kogge-Stone PPA, located inside the adder-subtractor. The critical path goes through the 16-bit inverter, continues through the 16-bit 2x1 MUX, then into the Kogge-Stone PPA, which comprises all four logical stages as illustrated in green:A

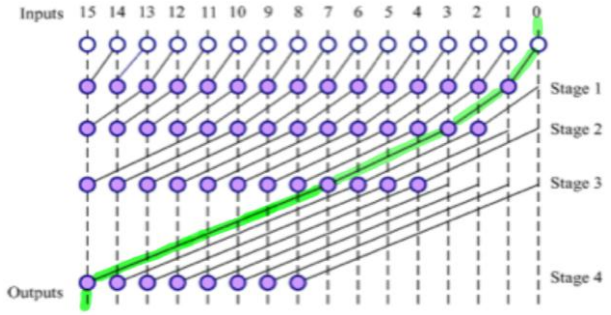

Figure 10. The critical path of a 16-bit Kogge-Stone PPA

4.3 Dadda multiplier results

The following table outlines the delay, area, energy, and metric for our Dadda multiplier:

Table 4. Team NAND's Dadda multiplier's metric values

Component	Value	Component	Value
Delay, D	603 ps	Energy, E	.478 nJ
Area, A	259 μm	Metric (D^2*A*W)	$7.47*10^{-23}$ s^2*m*W

5. EXTENSIONS

While Team NAND believes its DSP is the highest qualified candidate up for PICo's consideration, we have garnered a few additional ideas that could be utilized in a future Team NAND-PICo collaboration. The following section outlines these ideas.

5.1 Encoding/decoding the inputs

Reduced power consumption, area, and delay could potentially be achieved by encoding the inputs prior to the ALU. Doing this would lead to smaller logic gates and arithmetic functions, which in turn would consume less power, comprise less area, and reduce delay. After the inputs have been performed on by the ALU, they could then be decoded, thusly achieving a fairly advantageous DSP.

5.2 DeMUXing VDD

Utilizing a reverse multiplexor to DeMUX VDD to each ALU function could save power, as in a traditional ALU, all the functions are still "on" and consistently performing its operation. If one were to only turn on one function at a time, power consumption could be greatly reduced.

5.3 Multi-fault-tolerant, reconfigurable adder

While the 2-D regularity found in memory circuits lends precedence to common fault-tolerant techniques such as parity or error correction code (ECC), various fault-tolerant schemes for 1-D regularity and irregular random logic have yet to become standard. Triple modular redundancy (TMR) schemes are costly in overhead, but on-going research has brought novel fault-tolerance to arithmetic functions due to their 1-D regularity.

One current method being explored is by utilizing the inherent redundancy found in PPAs. By constructing a full-tree adder (Figure 11) out of a Kogge-Stone and the reverse tree found in a Brent-Kung, the full-tree comprises enough nodes to effectively produce different varieties of a Han-Carlson and Brent-Kung PPA. Then by utilizing a multiplexor at each node (illustrated by a black dot), reconfigurability can be achieved. Determining the combination of nodes that give functioning adders can be difficult, but *A New Taxonomy for Reconfigurable Parallel Prefix Adders*[2] reduces that difficulty by organizing adders into a design space.

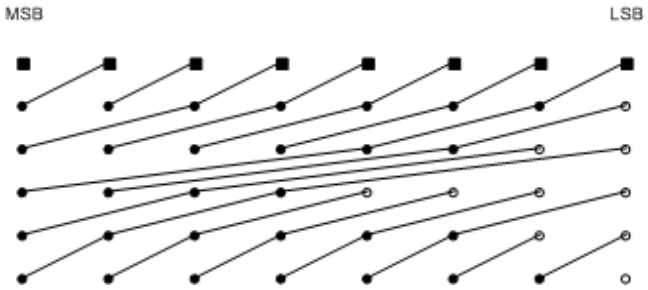

Figure 11. An 8-bit full-tree graph used for reconfigurability

So what does this mean for PICo? A multi-fault-tolerant, reconfigurable adder provides a robust DSP experience, with an indescribable level of trust with regards to its functionality. Such a solution is extremely novel and very cutting edge. Luckily for PICo, a member of Team NAND is involved with such research, and such exclusivity agreements could be easily reached.

6. CONCLUSIONS

In this paper, Team NAND presented to PICo a design for a digital signal processor. While the DSP had certain requirements, Team NAND took it upon themselves to implement high performance topologies, space-saving circuits, and a highly advanced multiplier technology. The pass-gate logic multiplexor utilized, buffered to compensate for swing, serves effective functionality while vastly reducing area, delay, and power consumption when compared to our competitors' common multiplexer design. The advanced Kogge-Stone parallel prefix adder provides $O(log_2N)$ latency, producing a critical path delay twelve times less than the standard Ripple Carry topology used by our competitors (even if they used mirror adders). The combined adder-subtractor allows for greatly reduced area (a savings of 57.6% over having a separate adder and subtractor), while reducing overall power consumption, yet still providing exceedingly fast performance. Our Dadda tree multiplier not only surpasses the performance of common array multipliers; but also, is even faster and utilizes less gates than our competitors' Wallace tree multipliers. Lastly, our extensions mentioned provide a hopeful future for a Team NAND-PICo collaboration, developing a DSP 2.0 with reduced power consumption, area, and delay through the encoding/decoding of inputs, vast reduction in power consumption by the use of a demultiplexer on the main voltage supply, and robust multi-fault-tolerance, reconfigurability via the use of a full-tree parallel prefix adder with MUXed nodes. Thusly, we conclude, that by the implementation of our innovations and future ideas, Team NAND is the best choice for PICo.

7. REFERENCES

[1] Tree graphs from *Hardware algorithms for arithmetic modules*, http://www.aoki.ecei.tohoku.ac.jp/arith/mg/algorithm.html

[2] B Sutherland, B. Sproull, and D. Harris, *Logical Effort*, Morgan Kaufmann, 1999.

[3] S. Bailey, M. Stan, *A New Taxonomy for Reconfigurable Parallel Prefix Adders*, University of Virginia, Charlottesville, VA, 2011