Prime Num Generator - Maker Faire 2014

Experimenting with math in hardware

Stanley Ng, Altera

Synopsis

The Prime Number Generator ("PNG") counts from 1 to some number (273 million, on a Cyclone V C5 device) and determines if each number is a prime number. It does so very quickly: <6 seconds to 273 million. It is able to do this because of the massive parallelism possible on the FPGA as compared with a software program: there are 1900 dividers operating simultaneously on the device. This can be thousands of times faster than programs on a PC even though the FPGA runs at 1/60 the frequency of a PC's CPU.

Introduction

The concept of a prime number is easy to grasp--a prime number programming assignment is commonly assigned in basic courses in basic programming languages, and yet prime numbers are important in the real world—the difficulty in determining whether a number is prime, is the basis of a lot of computer security. So I decided to explore prime number generators in FPGA educational kits. I've come up with several implementations of varying speeds and capacities; the one I'm demonstrating here is the Sieve of Erastothenes.

How do you know if a number is Prime?

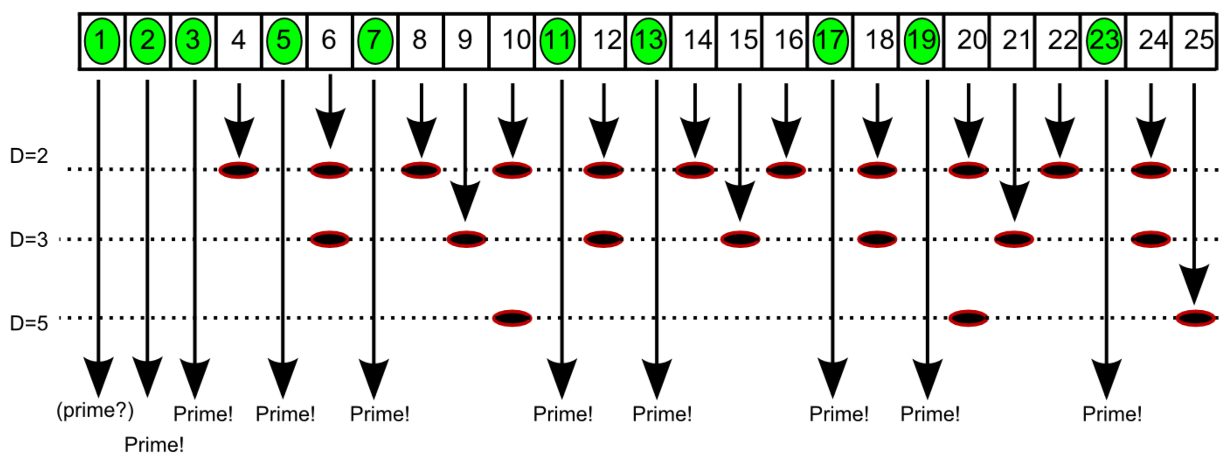
The most obvious way is a "Trial Division" algorithm: try all the numbers that can possibly divide that number, and if none of these numbers divide, then the number is prime. You can tweak this basic technique to be faster, by trying only prime numbers and by ending your search up at the square root of the number in question: for example, to see if 9,543 is a prime, you only have to divide by primes up to 100 (because sqrt(9543) is less than 100)).

However, the fastest prime number searches are "sieves". These start with the set of all positive numbers up to some high limit, and then number are eliminated by deleting every Nth number—for example, delete every 2nd number because they are divisible by 2; delete every 3rd number because they are divisible by 3, and so forth. When you're done, all un-crossed numbers are prime.

This method is called "The Sieve of Erastothenes", after the ancient Greek mathematician who came up with it. Think of a multiple sieves, each with different sized holes. Each sieve hole-size is a different prime number. What isn't caught by the first sieve element will be caught by another sieve element.

Here is an illustration of the concept. “D” is a denominator for each of the numbers. For each number in the top line, we see if our sieve elements of 2, 3, 5 catch any numbers divisible by 2,3,5. Those that aren’t caught will fall through, to be marked “Prime”.

The Sieve of Erastothenes



Example: primes up through 50 can be found with sieve values (2, 3, 5, 7)

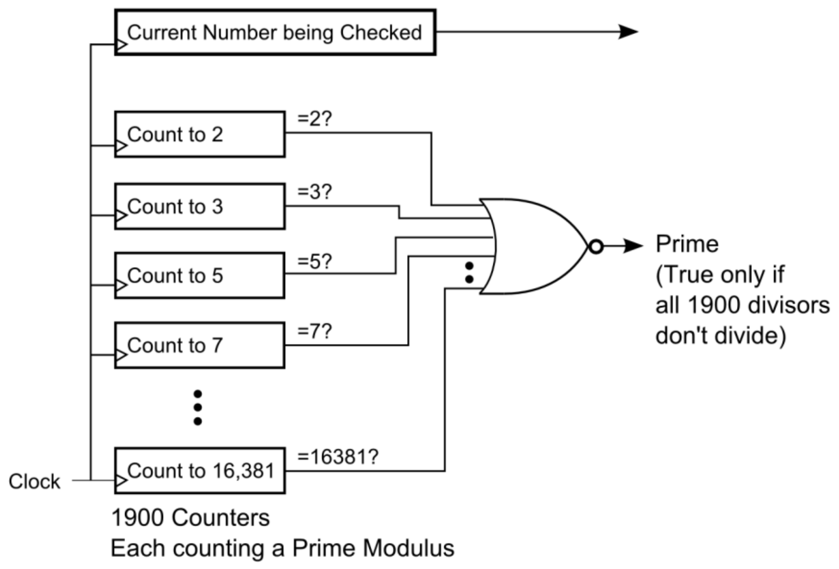
Sieves are fast because no divisions (which are computationally expensive) involved—only counting. We use this fact in our FPGA implementation: instead of dividing, we count by prime numbers. We create a count-by-2 counter, a count-by-3 counter, a count-by-5, each telling us when their terminal count is reached, indicating the current number is divisible by that factor.

We can implement around 1900 counters in our FPGA. That handles divisions up to the 1900th prime number, around 16381. That will let us check all the numbers up to 273 million for primality.

Implementation

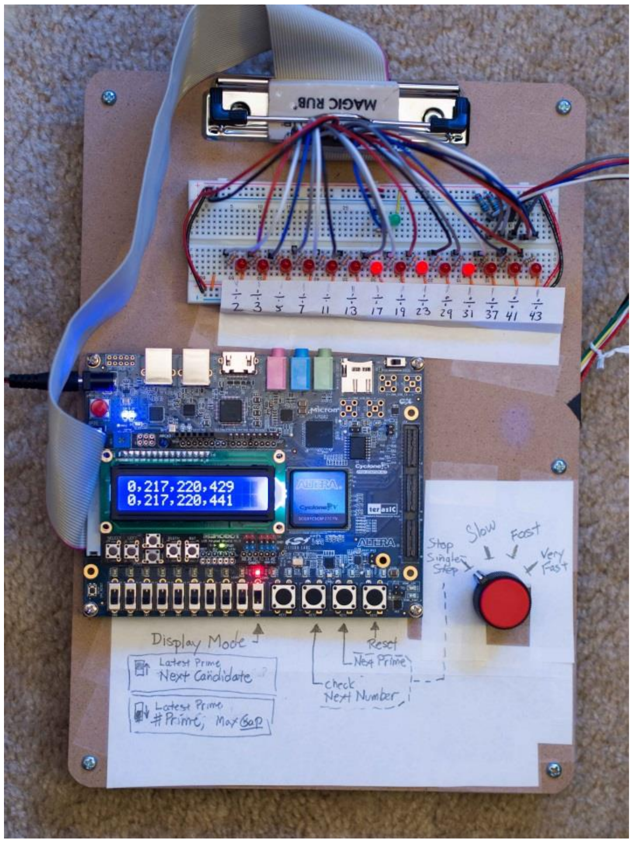
The C5G FPGA development board is used. It has a Cyclone C5 with 29,000 ALM's. This is used to ~98% capacity. The size of the sieve is scalable to use as much of the device as can be tolerated; this turns out to be a little over 1900 dividers. Only ALM's are used—no DSP and no memory blocks are used.

Each counter is programmed to count with the modulus being each of the first 1900 primes. Also implemented are a binary-to-BCD converter and finite state machines for the LCD driver and support logic for pushbuttons, rotary switch and slide switches. The "prime found" signal is a large NOR gate, 1900 inputs wide. Since the clock rate is low (50Mhz), no pipeline registers are needed.



The C5G has several buttons and slide switches, and 4, 7-segment LED displays. I've supplemented these with added hardware, all mounted on a clipboard with some nylon standoffs to provide under-the board space:

- An Arduino LCD Shield – this shows off the C5G's ability to play with the hundreds of hardware add-ons available for the Arduino platform. Arduino is very important to the Maker community.
- A Rotary switch to select between the various operating modes of PNG
- 14 discrete LED's to show whether each of the first 14 sieves finds the current number divisible. This for visualizing the sieve concept. There are actually 1900 sieve elements, just the first 14 are brought out to light their own LED. These are mounted on a solderless breadboard to include current limiting resistors. And because exposed, flying jumper wires look cool to Makers.



Here is the PNG. The C5G is mounted on a clipboard, along with a rotary switch and a solderless breadboard with 14 red LED's and a green LED.

The top row of the LCD shows that latest prime found (217,220,429)
The second row shows the current number being checked (217,220,441).

RED LED's shows the divisors of the current number.
GREEN LED is lit when the number is prime.

The knob selects how fast the numbers should be checked

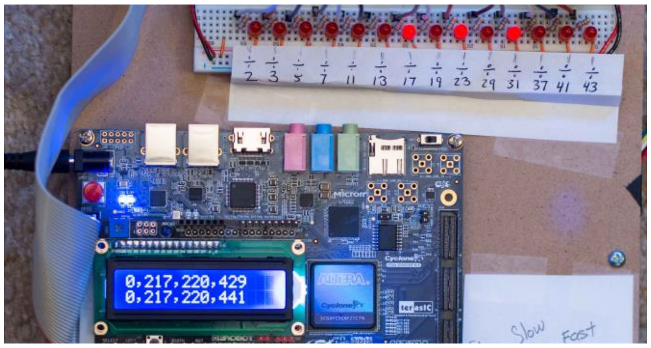
1. **Single-Step**. Numbers do not advance unless you press one of the buttons
    a. Check Next Number – advances the Current Number by one
    b. Select Next Prime – advances the counter until the number is found to be Prime
2. **Slow**. Automatically advance the counter ~1 per second
3. **Fast**. Advance the counter around 500,000 per second
4. **Very Fast**. Advance the counter around 50 million per second(!)
   The highest number is around 273 Million, so this will only last around 5.5 seconds.

There is a second display mode, selected by sliding SW0 (the slider switch) down.

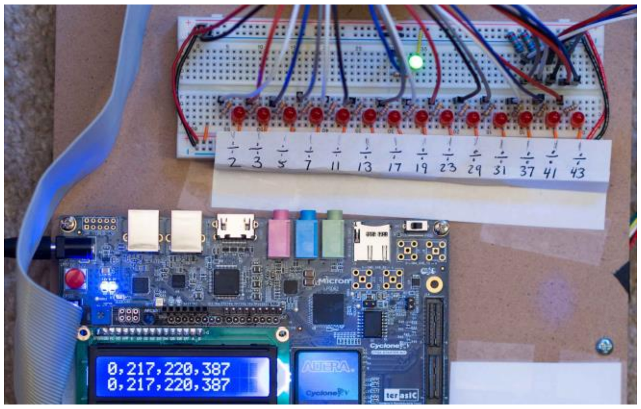
- Top row shows latest prime number
- Bottom row shows # primes found, and Maximal Prime Gap

The Maximal Prime Gap is the biggest gap between two consecutive primes found so far.

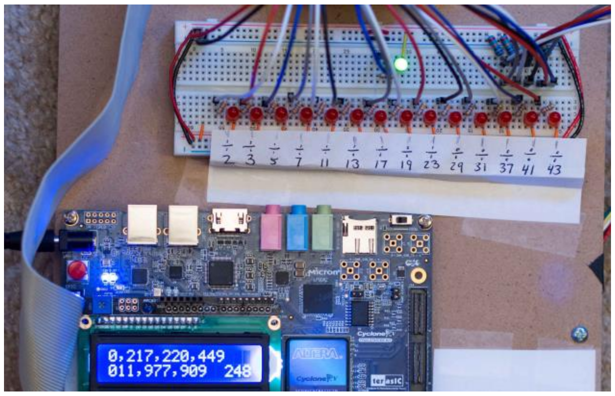
Here are some examples of different number displays.



217,220,441 **is NOT** a prime number. The sieve elements for 17, 23, 31 are lit so they all report that this number is divisible (it's also divisible by 17,921, the 2051st prime, but I don't have that many LED's!)



217,220,387 **IS** a prime number. None of the divisors is lit, up through 43, and none of the other ~1900 dividers have shown it's divisible by any of them either. Although we don't have 1900 LED's we know no other dividers have reported divisibility because the green LED is lit.



217,220,449 is a prime, and it's the 11,977,909th prime number. The maximal prime gap is 248.

This display is gotten by flipping the "Display Mode" switch to the "down" position.

The Demonstration

The Starting Configuration

1. Display Mode Switch – Down (to show latest prime, and currently tested number)
2. Rotary Switch – Fast, until you reach some number < 270,000,000 then…
3. Switch to "SLOW" mode

The PNG will be counting some large numbers, showing primes and factors for the first 14 primes (up to 43) for non-primes. The green light will turn on when PNG finds a prime.

Showing the Sieve in Action

1. Switch Rotary knob to "Stop/SingleStep"
2. The LCD Displays
   a. Latest prime
   b. Current number being checked.
3. The LED's displays
   a. RED LED's: the prime factors up to 43
   b. GREEN LED: whether or not the number is prime.
4. Push the "Check Next Number" button to advance the Current Number by one
5. Push the "Next Prime" button to advance until the next prime is found.

The Sieve is simple: each LED counts the number of times labeled and then turns on. Example: press the "next number" button until the "7" LED lights. Press 7 more times, and you'll see it light again.

One way to view this operation: all counters have a chance to veto the decision to call the number a prime. Any counter can assert divisibility and the green light stays off. If *all* counters don't have an objection to calling the number prime, then the green light turns on.

How Fast is it?

After showing the single-step, return to the "slow" setting to show PNG counting up by around 1 per second.

Then show that operation at "fast" which is 1/1000th full speed, then full speed.

The entire 273 million numbers are exhausted in ~5.5 seconds. I've looked for software-based prime number search programs. Software runtimes on a 3+ Ghz PC vary: ~20 minutes to get to 100,000 primes in one fast implementation; 110 seconds to get to 100 Million in another, on a scripting language. These equate to an FPGA performance advantage of between 50x and 50,000x even with a clock rate of 50 Mhz—1/60th the clock rate of a PC.

Suggestions for Further Work

1. Remove limitation of 273 Million as the maximum number. This might be done by using sieves with several passes
2. There is a bit of “cheat” here: all sieves are pre-loaded with prime numbers. A bootstrap approach (loading counters as primes are discovered instead of seeding the counters) seems more elegant. I considered doing this but it would have been much more complicated to bootstrap the seeding of counters, and it would have been less area-efficient because 16-bit counters would need to be pre-allocated for all counters even if you only need a 1- or 2-bit counter (divide by 2, divide by 4).
3. Another way to remove the limit is to make a hybrid prime counter: use a sieve to get to some threshold, perhaps 220 million, then use a trial divide function to find further primes. Trial divide is slow, but quite small—around 400 ALM’s
4. Include a soft processor for control. An inordinate amount of effort was devoted to writing LCD driver complete with binary-to-BCD converter. I wanted to assert that there is NO PROCESSOR involved in this project, a fact which startled may demo audiences. But it’s not easy to work at the bit banging level!

Conclusion

FPGA’s can be especially fast for certain well defined operations that lend themselves to parallel execution. In this case a sieve algorithm allows the use of as many counters as can fit into an FPGA to accelerate operation, resulting in a performance advantage of 50 to 50,000 times over PC programs doing this function.