HyperPipelining of High-Speed Interface Logic

Gregg Baeckler
02-16-2016
Define “Hyper Pipeline”?

- [Intel ~2000]
  - A 20 stage (heavy, speed record setting) Pentium 4 pipeline

- [Altera internal, ~2011]
  - The word assigned to the already prevalent network logic practice of adding more registers until the circuit stops going faster

- [Altera, now part of Intel 2015]
  - Pipelining a RTL design with Stratix10 HyperFlex™ Registers

Why? Pushing up the clock speed.
Why the clock needs to go faster

- Because it was there (?)
- A little googling tells you humans do obsess over speed
- Ignoring why, my job is to help, I’ll talk about the what and how.

“THERE IS MORE TO LIFE THAN INCREASING ITS SPEED.”
MAHATMA GANDHI

“I FEEL THE NEED... THE NEED FOR SPEED.
-TOP GUN

"If one day the speed kills me, do not cry because I was smiling"
-Paul Walker (1973-2013)
There aren’t a lot of quote images for “parallelism”

Complicated, Amdahl.

N things at 100 MHz is generally less exciting as one thing at N-hundred MHz.

People who say otherwise can’t do N-hundred (yet)
In (mainstream) FPGAs the clock speed is governed by

The worst case of - How much look up table (LUT) and wire is between some pair of registers in the design

Myriad secondary effects
Delay for single path register -> (N) LUTS -> register

Collected on Altera Arria 10
Fast speed grade device

LUT depth
- 8
- 7
- 6
- 5
- 4
- 3
- 2
- 1
- 0

Manhattan distance between src and dst regs
Interpreting these path delays

Respectable summary formula for Arria 10
- Delay = (188ps * LUT_depth) + (22ps * distance) + 540ps
- Pretty well aligned with silicon model delays

Can I really run no-LUT no-routing R2R at 1.8GHz?
- Maybe. You can timing close a little FIFO at 800MHz
- On the desk, room temperature you can spin it up well past 1GHz before it fails.

Routing is dangerous. Traveling 8.5 points ~ Another LUT delay

These are singleton near-optimal hops, let’s look at repeating them, with zero LUTs
How far can I travel between two N bit registers at 400MHz?
Bus reach sample point

- This is a zoomed in routing view of a missed 512 bit probe, it is only rated for 309 MHz.

- You can see wires traversing all across the rectangle, and some emerging congestion in pink.

- If it were 16 bits it would be meeting 400 MHz comfortably.
What’s going on

- Competition for physical resources
- Minimum of N trials

- Widest busses in somewhat routine use today are 1024-1280 bit
  - This is 300-500G
  - Yeah it hurts ->

- Around 2048 it becomes impractical to move

(This is a 400GE/500G Ilk bridge on Stratix 5)
Key observations

- The LUTs are fundamentally fast. If you aren’t travelling you can go through a bunch of them (e.g. a carry chain).

- The speed of travel is impeded by the number of travelers.

- At typical communication distances (say 15) the routing delay and delay of a couple LUTs are about equal.

- There is an unpleasant noise-floor term, traveling zero distance through zero LUTs costs a half nanosecond.
  - Stratix 10 is going to tackle this aspect.
Back to “hyper pipeline”

- The physical resources in the A10 device provide >1 register per LUT
- Operate comfortably > 500 MHz
- Deviating far from here is inefficient

- Taken together with the previous speed numbers you see a sweet spot where a 1:1 REG:LUT path can traverse about 50 points (1/4 device), operate at 500MHz, handle bus up to 512 bits

- This is more heavily pipelined than “average” FPGAs today, but routine for high speed interface logic
Define “high speed interface logic”

The label makes me uncomfortable

- An AND gate in a state machine, or an ALU, or masking out a packet header at 400G. He’s just an AND gate.
- The ‘type’ of logic is I think a completely human red herring construct

Anyway, “the logic near the edges of the chip, usually running a well defined protocol like Ethernet, Interlaken, PCIE, memory control, usually talking to neighboring chips” is “interface logic”

It tends to be made by obsessive designers for reuse by others. As such it tends to be fast, compact, high bandwidth

Using Ethernet as an example…
If you want to understand 40/100/400 G Ethernet

There is a 450 page IEEE specification which talks about the relevant portions of the standard

The Altera 100G Ethernet with all optional features is about 1000 pages of Verilog

You can explain the digital meat pretty well with a 2 page C program, which talks mostly about byte operations, some 64 and 66 bit.

Why so long?
Endless detail

Parallelism, cell optimizations
The teacher did reply. Why does the lamb love Mary? The eager children cry. Why does the lamb love Mary patiently about, Till Mary did all it lingered near, And waited. The teacher turned it out, But still see a lamb at school. And so the children laugh and play. The lamb was sure to go. He fol b, His fleece was white as snow, The lamb was sure to go. He followed her. Was against the rule, It made the teachers turn it out, But still see a lamb at school. And so the children laugh and play. Mary had a little lamb, His fleece was white as snow, The lamb was sure to go. He followed her to school one day, Why did Mary love the lamb, you know so? The eager children cry. Why does the lamb love Mary patiently about, Till Mary did all it lingered near, And waited. The teacher turned it out, But still see a lamb at school. And so the children laugh and play. The lamb was sure to go. He followed her to school one day, Why did Mary love the lamb, you know so? The eager children cry.

Concept – 8 bits at 12.5 GHz
Status Quo – 256 bits at 390.625 MHz
Why that hurts

Ethernet is not one of the “Embarrassingly Parallel” problems, the circuit growth is typically violent.

Expansion currently done by humans (unfortunately) outperforming the HLS CAD tools.

The topology of the circuitry needs to follow the topology of the data, which doesn’t honor the break we introduced between “lam” and “b”:

- It is not two dimensional =>
- Imagine place and route tool’s problem view of this tube on saran wrap.
Placement in plastic

w. The teacher did reply. Why, Mary loves the lamb, you know, so? The eager children cry. Why does the lamb love Mary patiently about, Till Mary did all it lingered near, And waited for the teacher turned it out, But still see a lamb at school. And so the children laugh and play To ch was against the rule, It made Mary turn against the rule. The lamb was sure to go. He fol lowed her to school one day, Whi b, His fleece was white as snow, Mary had a little lam

Pretty good

In trouble

thrilled
FAQ

Does the place and route see that tube pattern?
- Often.
- Examine the directional routing in the chip planner, which is turbulent in areas where it should conceptually be one-way.

What happens when tubes intersect (e.g. a switch)?
- Topological havoc.

What can the designer do to help keep speed up?
- Add more registers.
There are a variety of situations that force bits physically apart. E.g. coupling to a RAM or SERDES pin.

There are a variety of computations that pull bits together in competing patterns. E.g. bit 0 relates to other bits of a word, the previous cycle, and bit 0 of other words.

If you scrutinize high speed designs you’ll encounter extra pipeline stages and manual register duplication aimed at releasing this sort of tension.
There is a wide variety of FPGA circuitry in the world running heavily pipelined (1 or 2 LUTs deep) logic around the 256-512 bit bus, 300-400MHz range
- Or 100-200 gigabit per second flows

And of course plenty of boutique logic above, legacy below, and variations

The speed drifts up slowly as the devices improve
- But perpetually less than the appetite

There is always friction
#1 request

“I am having trouble closing timing, please slow down by a factor of 2 and increase the bus width by a factor of 2”

Recipe for disappointment

Area and latency > 2x  (area ~ cost, compile time)
- Poor inherent parallelism
- Unpleasant surprises, like encountering multiple packets per cycle

Flat power if lucky

A lot of the new found cycle time evaporates into routing, double area implies double the required travel distance

We help customers who ask for this, many revert back
#2 request

“I want to go faster, but inserting registers isn’t helping anymore.”

Why?
- A bus register takes up physical space which pushes things apart, which tends to make them slower
- It takes time to get in and out of a register recall the zero cost point to point path at a half nanosecond, not 0 nanoseconds

** Stratix 10 HyperFlex registers **

512 bit register
Why adding registers fizzes out at some point

- Fast designs today have more routing than logic delay
- It’s annoying to get off the wire, find a home for a register, mux in and out again.

Observation – the registers are TINY, far from datasheet scale
Stratix 10 has ‘plenty’ of Hyper-Registers distributed into the routing network
Routing muxes (all H/V wires) have *optional* registers
- Including LAB, M20K and DSP block inputs, CC, SCLR/CE

Architectural Goals:
- Perfect balance – P&R chooses the right register (of many) to turn on
- Simple Software – Re-timing is a simple push/pull along the path
- No wasted LEs – Designs with high FF:LUT ratios no longer an issue
- No wasted routing – Don’t have to route to find an available FF
What more registers do to a real circuit

Interlaken transmit datapath, 12 words (768 bit). From a Altera 300G hardware demo

Designed for 390 MHz plus 20% margin = 468 MHz on Stratix 5
Currently scoring 561 MHz on Arria 10

Stratix 10 prototype making 842 MHz

Variations assuming changes to register counts, hold avoidance software, etc..

The software (illegally) increased the latency from 13 to 14, and (legally) did some minor register retiming. The latency edit amounts to adding a layer of input registers and fixing up the valid schedule. Nothing extreme. Heavily pipelined logic doesn’t need much rework to shine in the new fabric.
Implications of this result

- For a 10 minute RTL edit to legalize what the prototype software just assumed I would get (real) 842 MHz
- I listed the target at 781 MHz, which is double the current operating rate of 390 \((390.625 \text{ MHz} \times 768 \text{ bit} = 300\text{Gbps})\)

- This thing is now a 600G Interlaken.
- If you actually wanted 300G
  - The datapath slices to 384 bit, area and latency (in ns) less than half
  - Stress taken off the place and route tool from half sized problem, speed will increase.
  - Option to move to a smaller device, option to add another port (speed is very convertible to other benefits)
Good secondary effects

Less parallel = easier to think about = less circuit, less buggy

IP makers in the process of retargeting major cores

Register retiming works dramatically better due to having more and finer grained choices
Challenges

- Harder to shake off small absolute delay surprises

- Unsettling for many FPGA designers to see 10x registers and be asked to not fill them up.

- Chip is similar scale, and capable of sustaining much higher toggle rates. Customer bandwidth appetite ~unlimited. Clock and power distribution more exciting.

- The usual concerns about verification and debug-ability in the presence of retiming
Where we’re heading with soft IP

Science projects at extreme speed, this is a Ethernet MAC targeted at 1.25GHz, described at Ethernet Technology Summit

<table>
<thead>
<tr>
<th>Project</th>
<th>Device</th>
<th>Logic Cells (ALMs)</th>
<th>Registers</th>
<th>Performance (MHz)</th>
<th>Latency (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10G Ethernet MAC</td>
<td>Arria 10</td>
<td>1500</td>
<td>2600</td>
<td>340</td>
<td>59</td>
</tr>
<tr>
<td>Falcon 10G MAC</td>
<td>Stratix 10</td>
<td>338</td>
<td>1231</td>
<td>1358</td>
<td>12.8</td>
</tr>
</tbody>
</table>

Double pumping RAM / DSP / switching variants
Where we’re heading with soft IP

More ports and fatter pipes.

This is a 400G Ethernet CRC assembly, used in Stratix 5 hardware demo on YouTube (390 MHz x 1024 bit)

With minimal labor, it is modeling at 896 MHz

896 MHz * 1024 bit is a stone’s throw from Terabit.
Where we’re heading with soft IP

In the presence of a register surplus, and fluid retiming, loops inevitably become critical.

We’re having a resurgence of interest in Shannon-izing decisions, methods for restructuring accumulations (both manually and automatically) and adapting early decisions to anticipated future retiming.
In summary

Bandwidth is going to keep climbing, pipelines will grow steadily deeper, and architectural changes will take the sting out of the increased register count.

For more information

  - Myriad white papers and app notes.

- The training section is a (somewhat hidden) gem
  - https://www.altera.com/products/fpga/stratix-series/stratix-10/support.html#training

Lots of direct and detailed discussion written by SW / TS / IP staff on the RTL strategies they use to make FPGA designs go faster, with lab examples.
Thank You