FPGA Carry Lookahead Adder (1D)

-
-

Carry Lookahead and Brent-Kung

there are two inputs to the fast carry logic C1, C0
the value of C1, is programmed by the LUT's so that
it contains the value that Cout should have if Cin is false.

We can combine the information from two stages together
to determine what the Cout of one stage will be given
the Cin of the previous stage.

$$C1_{i,i-1} = (C1_{i-1} * C1_i) + (\overline{C1_{i-1}} * C0_i)$$
$$C0_{i,i-1} = (C0_{i-1} * C1_i) + (\overline{C0_{i-1}} * C0_i)$$

Where C1_x,y is the value of Cout, assuming that Cin_y =1
This allows us to have the length of the carry chain
Since once these new values are computed a single mux
Can compute Cout, given Cin_i-1
In fact, similar rules can be used recursively,
Halving the length of the carry chain with each application

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Lookahead and Brent-Kung

$$C1_{i,k} = \left(C1_{j-1} * C1_{i,j}\right) + \left(\overline{C1_{j-1,k}} * C0_{i,j}\right)$$
$$C0_{i,k} = \left(C0_{j-1,k} * C1_{i}\right) + \left(\overline{C0_{j-1,k}} * C0_{i,j}\right)$$

Assuming i > j > k

The digital logic computing both of these functions will be
Called a concatenation boxes, where each level in the hierarchy
Halves the length of the carry chain, until we have computed C1_i,0
And C0_i,0 for each cell I

A single level of muxes at the bottom of the Brent-Kung carry chain
Can then use these values to compute the Cout for each cell
Given a Cin

The Brent-Kung carry chain

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Lookahead and Brent-Kung

The 3-level, 16-bit Brent-Kung structure
The details of the concatenation block
Note that once the Cin has been computed for a given stage,
A mux is used in place of a concatenation block

The Brent-Kung adder is a specific case of
the more general Carry Lookahead adder.
In a Carry Lookahead adder a single level of concatenation combines together
The carry information from multiple sources

A typical Carry Lookahead adder will combine 4 cells together
In one level (computing C1_{i,i-3} and C0_{i,i-3}, combine
Four of these new values together in the next level, and so on

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Lookahead and Brent-Kung

However, while a combining factor of 4 is considered for a standard
Adder, in FPGAs combining more than two values in a level
Is not advantageous

The problem is that although the logic to concatenate N values
Together grows lineary for a normal adder, it grows exponentially
For a reconfigurable carry chain.

For example, to concatenate three values together for a normal adder
We have the equation

C_{x,z} = G_x +(P_x times C_{x-1,z}

Where P_x and G_x are the propagate and generate values of the
Current cell

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Lookahead and Brent-Kung

However, to concatenate three values together for a reconfigurable
Carry chain together we have the equation

C1_{w,z} = (C1_{x-1,z} times C1_{w,x}) `+` (overline {C1_{x-1,z} times C0_{w,x})

= (C1_{y-1,z} times C1_{x-1,y}) `+` (overline {C1_{y-1,z} times C0_{x-1,y}) times C1_{w,x}

+ (C1_{y-1,z} times overline {C1_{x-1,y}}) `+` (overline {C1_{y-1,z} times overline {C0_{x-1,y}}) times C0_{w,x}

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Lookahead and Brent-Kung

An alternative way to see why combining 4 cells together
In one level is bad for FPGAs is to consider how this combining
Would be implemented

Figure shows a concatenaton box can be built using only three 2-cell concatenation boxes
This second method of creating a 4-cell concatenation box is really the equivalent of a 2-Level
Carry Lookahead adder using 2-cell concatenation boxes

Using the simple delay model discussed earlier, the delay for the 4-cell
Concatenation in Figure is 6 units since the signal must ravel through 3 muxes

The delay for the 4-cell concatenation box equivalent found in Figure, however,
Is only 4 units since the signal must ravel through only 2 muxes

Thus a 4-cell concatenation box is never used since it can always be implemented
With a smaller delay using 2-cell concatenation boxes
Therefore, the Brent-Kung structure is the best approach

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Lookahead and Brent-Kung

Another option in Carry Lookahead adders is the possibility of using less
Levels of concatenation than a Brent-Kung structure
Specifically, a Brent-Kung structure for a 32-bit adder would require
4 levels of concatenation

While this allows C_in0 to quickly reach Cout_31, there is a significant amount
Of delay in the logic that computes the individual C1_i,0 and C0_i,0 values

We can instead use fewer levels than the complete hierarchy of the Brent-Kung
Adder and simply ripple together the top-level carry computation of small
Carry lookahead adders

Specifically, if we talk about an N-level Carry Lookahead adder, that means
That we only apply N levels of 2-input concatenation units
A 2-Level, 16 bit Carry Lookahead carry chain is shown in Figure

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Chain Performance

In order to compare the carry chains developed,we computed the performance
Of the carry chains of different lengths
The delay is computed form the output of the 2-LUTs in one cell to the final output (F)
In another using the simple delay model

This simple model calculates a delay based on the number of gates
that must be traversed by a signal
Precise circuit timings are discussed
Figures show the delays of a carry chain starting at Cell X and ending at Cell Y
For the Basic Ripple and Brent-Kung carry chains, respectively.
These figures show how the delay patterns are different for each carry chain

One important issue to consider is what delay we should use to compare carry chain performance

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Chain Performance

While the carry chain structure is dependent on the length of the carry
Computation supported by the FPGA (such as the Variable Block segmentation),
The user may decide to use any contiguous subsequence of the carry
chain’s length for their mapping

To deal with this, it is assumed that the FPGAs are built to support up to a
32-bit carry chain, and record the maximum carry chain delay for any length L
Carry computation within this structure
That is, since we do not know where the user will begin their carry computation
Within the FPGA architecture, we measure the worst case delay for a length L
Carry computation starting at any point in the FPGA
Note that this delay is the critical path within the L-bit computation,
Which means carries starting and ending anywhere within this computation
Are considered

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Chain Performance

Figure shows the maximum carry delays for each of the carry structures,
As well as the basic ripple carry chain found in current FPGAs
These delays are based on the simple delay model that was discussed earlier.
More precise delay timing from VLSI layouts of the carry chains will be discussed later

As can be seen, the best carry chain structure for short distances is different from
The best chain for longer computations, with the basic ripple carry structure
Providing the best delay for length 2 carry computations,
While the Brent-Kung structure provides the best delay for computations
Of four bits or more

In fact, the ripple carry structure is more than twice as fast as the Brent-Kung
Structure for 2-bit carry computations, yet is approximately eight times slower
For 32-bit computations

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Carry Chain Performance

However, short carries are often not that critical, since they can be
Supported by the FPGA’s normal routing structure and will tend not to
Dominate the performance of the overall system.

Therefore, it is believed that the Brent-Kung structure is the preferred structure
For FPGA carry computations, and that it is capable of providing significant
Performance improvement over current FPGA carry chains

Other carry lookahead adder designs were considered which do not use
As many levels of concatenation boxes as a full Brent-Kung adder.
However, as can be seen in figure, the other carry structures provide only modest
Improvements over the Brent-Kung structure for short distances, and perform
Significantly worse than the Brent-Kung structure for longer carry chains

https://en.wikipedia.org/wiki/Carry-lookahead_adder