FPGA Variable Block Adder (1C)

-
-

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Variable Block

like the carry select chain, a variable block structure
consists of blocks of ripple carry element
however, instead of precomputing the Cout value
for each possible Cin value, it instead provides a way
for the carry signal to skip over intermediate cells
where appropriate.
Contiguous blocks of the computation are grouped
together to form a unit with a standard ripple carry chain
As part of this block, logic is to the value of the block's
Cin, allowing the carry chain to bypass this block's normal
carry chain on its way to later blocks.

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

Variable Block

The Cin still ripples through the block itself, since
the intermediate carry values must also be computed
If any of the cells in the carry chain are not in propagate mode,
the Cout output is generated normally by the ripple carry chain.
WHile this carry chain does start at the block's Cin signal,
and leads to the block's Cout, this long path is a false path
That is since there is some cell in the block that is not in
propagate mode, it must be in generate or kill mode,
and thus the block's Cout output does not depend on the block's
Cin input

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

Variable Block

the variable block carry structure

mux1 performs an initial two sing stage ripple carry
mux2 ~ mux5 form a 2-bit variable block block
mux5 decides whether the Cin signal should be sent directly to Cout, while mux4 decides whether to invert the Cin signal or not

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

Variable Block

a major difficulty in developing a version of the Variable Block carry chain for inclusion in an FPGA's architecture is the need to support both the propagate and inverse propagate state the cells.

To do this, we compute two values.
First, we check to see if all the cells are in some form of propagate mode (either normal propagate or inverse propagate) by ANDing together the XOR of each stage's C1 and C0 signal

If so, we know that the Cout function will be equal to either Cin or Cin bar.

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

Variable Block

to decide whether to invert the signal or not,
we must determine how many cells are in inverse propagate mode.
if the number is even (including zero), the output is not inverted,
while if the number is odd, the output is inverted.

the inversion check can be done by looking for inverse
signal from each cell.
if this signal is true, the cell is in either generate or
inverse propagate mode, and if it is in generate mode inversion signal
will be ignored anyway (we only consider inverting the Cin signal
if all cells are in some form of propagate mode).

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

Variable Block

note that for both of these tests we can use a tree of gates to compute the result.
Also, since we ignore the inversion signal when we are not bypassing the carry chain we can use C1 as the inverse of C0 for the inversion signal's computation, which avoids the added inverter in the XOR gate

the organization of the blocks in the variable block carry structure bears some similarity to the carry select structure
the early stages of the structure grow in length, with short blocks for the low order bits, building in length further in the chain
in order to equalize the arrival time of the carry from the block with that of the previous block

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

Variable Block

however, unlike the carry select structure, the variable block adder must also worry about the delay from the Cin input through the block's ripple chain

Thus, after the carry chain passes the midpoint of the logic, the blocks begin decreasing in length.

This balances the path delays in the system and improves performance

The division of the overall structure into blocks depends on the details of the logic structure and the length fo the entire computation

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

Variable Block

We use a block length from low order to high order cells of 2, 2, 4, 5, 7, 5, 4, 2, 1 for a normal 32 bit structure
The first and last block in each adder is a simple ripple carry chain, while all other blocks use the variable block structure.

Delay values of the variable block carry chain relative to other carry chains

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