EE 486 lecture 15: What's new in Add and Multiply.

M. J. Flynn

Add

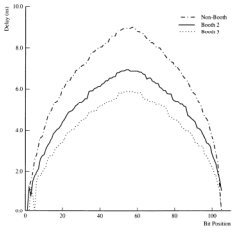
- Integer adders are now mostly available as macro cells. The best of which are parameterizable and carefully optimized.
- In optimizing for various area-time design points, the regularity of the design is not a consideration. The best designs are hybrid with variable length segments which attempt to equalize the delay across all paths.

Integrated Multiply Add

- By making the final CPA of the multiplier a 3 input adder, we can provide (AxB + C) in the same time as the product. This is now generally used in fp arithmetic, as the FMA instruction.
- This final CPA is also an example of the kind of optimization that's possible when the bit position arrival time is known.

CPA optimization

- In multipliers (as in some other situations) if the bit arrival time is known a more area time effective implementation is possible. Ripple at the low end, carry select on the most significant bits.

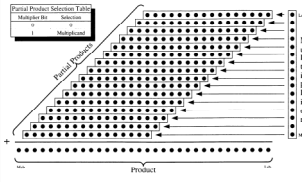


Bit arrival time at CPA, IEEE double precision

Multiply

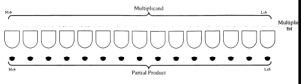
- Redundant Booth
- The effect of wires in determining the optimal implementation.

Non-Booth and pp selection overhead

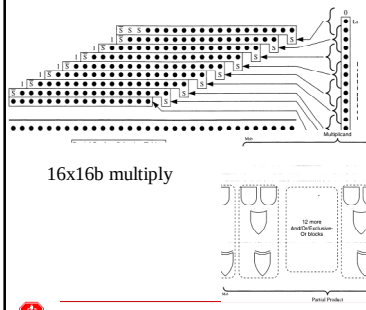


PP selection is a single AND gate of the shifted multiplicand

16x16b multiply



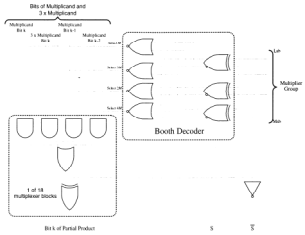
Booth 2 and selection HW



Now selection Requires 5 gate delays

16x16b multiply

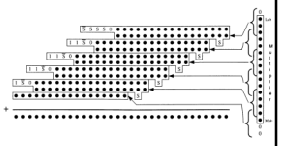
Booth 3 has similar selection overhead and requires the hard pp



Again selection requires about 5 gate delays But now we first need to form the +/-3 multiple

Eliminating the hard multiple

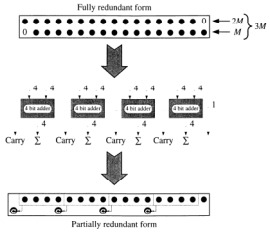
- We can eliminate the hard addition by simply entering both m and 2m instead of 3m.
- The multiple is in fully redundant form.
- But this doesn't reduce the height of the pp tree.



16x16b multiply with Booth 3 fully redundant hard pp

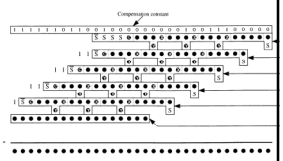
Partially Redundant Booth 3

- Suppose we overlap the delay in selection of the multiple with a partial or "digit" add of the hard multiple.
- This leaves us with a full pp plus a sparse pp for each hard multiple.



Partially redundant Booth 3: "digit" size

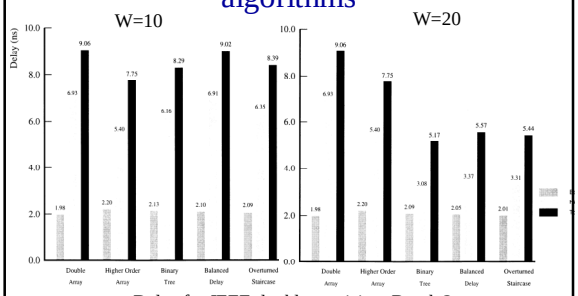
- The sparse pp should not align to increase the over pp height.
- The longer the add, the sparser the pp and the lower the overall pp height.



Partially redundant Booth 3: bias

- Note that the (–3) multiple will have sparse 0's in a pp of all 1's.
- We bias the pp's so that they're all positive.
- Then collect the biases as a single overall compensation constant.

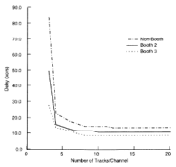
Wires play a big role in some algorithms



Delay for IEEE double precision, Booth 2

But wires play a much bigger role for W < 10

- For W <10 probably Booth with a higher order array is best.
- Once W > 10 or so, the problem for multiplier implementation is which tree algorithm is optimum.

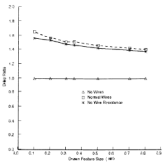


Multiply and feature sizes

- Smaller feature sizes create wire dominated implementations
- Wires, not gates, determine the delay
- Optimality of an implementation depends on the effect of wires, hence feature size

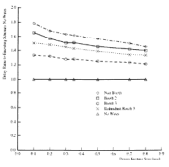
At 0.1 micron wires represent 70% of multiplier delay

- This could be much larger if we used faster dynamic logic.
- Results are for IEEE double precision with W =20.



The effect of wires on particular encodings

- The less the encoding of the multiplier, the greater the effect of wires on delay.
- Each algorithm is plotted relative to itself (w. no wires)



Overall multiplier conclusions

- Booth 2 seemed the best encoding for speed, Booth 3 for area.
- "Compiled" Wallace layout using (3,2)'s proved better than a structured binary tree.