Cairo University

Faculty of Engineering

Computer Engineering Department

Faculty of Engineering
Cairo University

VLSI

Mini Project #1

Floating-Point Adder

Team Members Info:

Name	Sec	BN	Email
Sarah Mohamed Hossam	1	30	sarahelzayat@outlook.com
Ahmed Atta Abdallah	1	6	aatta1082001@gmail.com
Abdelrahman Hamdy	1	36	abdelrahmanhamdy49@gmail.com
Abdelrahman Noaman	2	2	abdelrahmannoaman1@gmail.com

Adders Explanation:

- **Floating Point Adder**
  It sums the standard ieee format
  First, I shift the mantissas if there is a difference in the exponents
  Then I get the sum of the mantissas and Do normalization in case of the addition and subtraction.
  I detect the overflow and underflow and round the results to ±INF or to Zero

- **Verilog (‘+’) version of adders**
  Its just a simple adder with a + sign
- **Ripple Carry Adder**
  It is constructed with full adders connected in sequence

- **Carry Save Adder**
  - The carry save adder was originally made to reduce the addition of 3 numbers to 2. In our implementation, we ignored the 3rd input and set the sum to a width of N bits.
  - The 32 bit carry save adder consists of 64 full adders
  - The first 32 full adders are responsible to add each bit of the 2 inputs and ignore the carry. It’s obviously not thrown away but stored in an array of carries to be added in the next operation and the sum of the main bits are also stored in an array so we have 2 arrays to input in the next step.
  - The second 32 full adders add both arrays of sums and carries together with the carries shifted one bit to the left but with a special case for the first and last full adder. The first one receives cin as zero and the last one outputs the last carry as cout.
  - The last step is to output the first sum bit as it is (since we shifted the carries array) and then calculate the overflow by checking the last bits of the inputs and sum.

- **Carry Look-Ahead Adder**
  - The carry look-ahead adder is based on the concept of predicting the carry in each step to eliminate the propagation delay which occurs in parallel adders.
  - We first build a 4 bit carry look-ahead adder and initialize 3 4-bit arrays p (carry propagate), g (carry generate) and c (carries).
  - The equation to know the output carry from the summation of 2 inputs and cin is **Co = A.B + (A xor B).Cin** which can be easily obtained from a truth table.
  - The (A.B) part is called the carry generator because there is no dependency on cin. On the other hand, the (A xor B) part is called the carry propagator.
  - The equation is applied 4 times for the 4 carries we have but we need to make sure the dependency is always on cin only and the previous generators and propagators and this is why it gets longer due to the constant substitution.
  - The last carry is set as cout and the sum is simply the xor of p and c just like in a normal full adder (in1 ^ in2 ^ cin)
  - For the 32 bit carry look-ahead adder we apply the 4 bit addition 8 times to cover up the whole width and the advantage here is that we no longer wait on the carry between each stage but we already have it ready from the above calculations.

- **Carry Increment adder**

  - Carry increment adder is divided into multiple sectors, the first is a regular ripple carry adder, the output sum of this adder is taken as it is.
  - Any other sector consists of a ripple carry adder with a carry in of 0, half adders and the carry out of the previous block as an input.
  - The first half adder adds the carry of the previous block and the first bit of the ripple carry adder's sum, the sum of the half adder is takes as it is, and the carry out signal is taken as an input to the next half adder…etc.

- The final carry out signal of the block is the result of oring the ripple carry adder's carry out and the carry out of the last half adder in the block.

- **Carry Skip adder**

  - Carry skip adder consists of number of ripple carry adders with a special block which plays a role in speeding up the adding process.
  - This block is called skip chain which makes Carry Skip Adder much faster than ripple carry adder especially in large number of bits, this block is repeated for every 4 bits.
  - It aims to provide the propagation speed of the carry bit but how does it make that?
  - First of all we calculate the group propagate signal
    Pi = XOR(Ai, Bi) **(for i is from 0 to N/4-1)**
    P = And(P0, P1, P2, P3, P4, P5, P6, P7) **(in case of 32 bit)**
  - Then this signal is our selector so if it's 1 then the carry in bit will propagate to the next ripple carry adder
  - But if it's 0 then the next ripple carry adders will need to wait for the carry out of the previous ripple adders.
- **Carry Bypass adder**
  - Like the ripple carry adder each full adder waits for the carry of the previous one, Carry Bypass adder improves that in some conditions in which the next full adders won't need to wait for the carry of the previous ones
  - That cases are when the group propagate signal is set to one so the carry in will propagate to the next group of the full adders
  - Pi = XOR(Ai, Bi) **(for i is from 0 to N/4-1)**
    P = And(P0, P1, P2, P3, P4, P5, P6, P7) **(in case of 32 bit)**
  - And this is repeated each 8 bits (for each 8 full adders)

- **Carry Select Adder**

- Carry select adder is divided into sectors, the adder calculates each sector’s sum and carry out for an input carry of 0 & 1 and selects the correct sum and carry out using a multiplexer with the selector as the actual input carry of the sector.
- As it does double the number of calculations for each sector, it consumes a large area and power but it’s very fast.

Why choosing the carry look ahead adder?

There’s always a tradeoff between speed, power, and area.

In terms of speed, it has a moderate time performance (as shown in the excel sheet), not the fastest yet not the slowest.

In terms of power consumption, it has the 2nd best power consumption amongst all adders, the 1st is the Verilog’s + adder which is very slow.

In terms of area, it also has the 2nd best area next to the Verilog’s + adder.

So, we chose the excellent power consumption and area with a moderate time to implement the floating-point adder.

Roles of each team member and estimated time they worked

Name	Role	Estimated time
Sarah Mohamed Hossam	Carry increment adder and carry select adder	2 days
Ahmed Atta Abdallah	Verilog’s + adder, carry ripple adder and floating-point adder	2 days
Abdelrahman Hamdy	Carry lookahead adder and carry save adder + TB	2 days
Abdelrahman Noaman	Carry skip adder and carry bypass adder + TB	2 days