2D Group 26

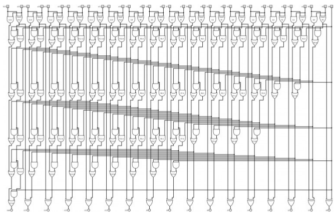
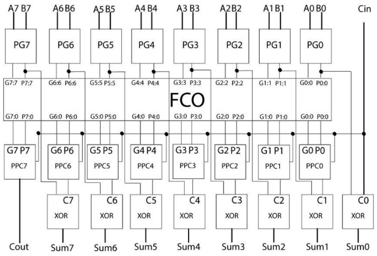
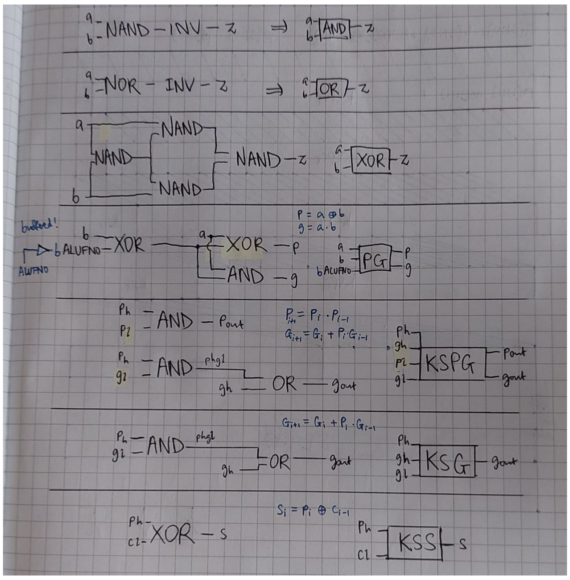
- Anshu Arun Ghate
- Cassie Chong Kenci
- Christy Lau Jin Yun (50.002 POC)
- Li Si Hui Brenda
- Tan Lay Lin
- Win Tun Kyaw

50.002 - Optimising an Adder

Part 1

Structure

Our goal is to optimise a 32 bit adder with delay being under 3ns.
Inspired by Kogge-Stone adder, our adder architecture comprises of gates arranged in a tree-like structure to allow parallel computation of addition, be using propagated and generated signals to calculate carry instead of relying on the previous outpit bit's carry out as done in a ripple carry adder.

16bit with all gates	8bit with abstracted components	Components
	 A7 B7 A6 B6 A5 B5 A4 B4 A3 B3 A2 B2 A1 B1 A0 B0 Cin PG7 PG6 PG5 PG4 PG3 PG2 PG1 PG0 FCO G7 P7 PPC7 G6 P6 PPC6 G5 P5 PPC5 G4 P4 PPC4 G3 P3 PPC3 G2 P2 PPC2 G1 P1 PPC1 G0 P0 PPC0 C7 C6 C5 C4 C3 C2 C1 C0 XOR Cout Sum7 Sum6 Sum5 Sum4 Sum3 Sum2 Sum1 Sum0	 a b NAND – INV – z ⇒ a b AND z a b NOR – INV – z ⇒ a b OR z a b NAND NAND NAND NAND – z a b XOR z buffered! ALUFN0 b ALUFN0 XOR a XOR – p AND – g p = a ⊕ b g = a · b a b ALUFN0 PG p g Ph Pl AND – Pout Ph gl AND phgl gh OR gout Pi+1 = Pi · Pi-1 Gi+1 = Gi + Pi · Gi-1 Ph gh Pl gl KSPG Pout gout Gi+1 = Gi + Pi · Gi-1 Ph gl AND phgl gh OR gout Ph gh gl KSG gout Si = Pi ⊕ Ci-1 Ph Cl XOR – S Ph Cl KSS S

32 bit adder consists of 1 `PG` layer, 5 `KSPG` and `KSG` layers, one last layer for one `KSG` for the carry out, and one `sum` layer to compute the sum bits.
Lastly, `z`, `v` and `n` are computed from the results of the adder.

What was done

All `AND`, `XOR` and `OR` gates were remade with inverting logic.
The most loaded signal `ALUFN0` is buffered.
Kogge-Stone architecture was chosen to allow for parallel prefix adding to speed up computation as compared to ripple carry adders.

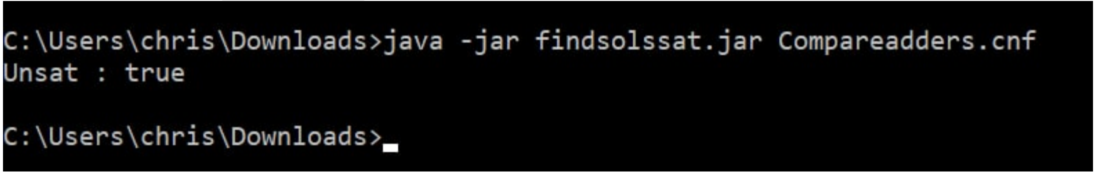
Stats

Fastest timing of adder: 2.814ns
Circuit size: 1337 gates
Minimum observed setup =829.52ps

Part 2

We chose the 8th bit for easier computation.
A BC file is created from the circuit design, which is then converted into a CNF file by using the telegram submission bot.

The CNF file is fed into `findsolssat.jar` and UNSAT result is true.



```
C:\Users\chris\Downloads>java -jar findsolssat.jar Compareadders.cnf
Unsat : true

C:\Users\chris\Downloads>
```