VLSI Final Project

Name: 徐竣霆

Student ID: 110060012

The circuit diagram of your design and explaining your

design

Basic gates

```
.subckt INV IN OUT
Mp1 OUT IN VDD VDD p_18 l=0.18u w=0.72u
Mn1 OUT IN GND GND n_18 l=0.18u w=0.36u
.ends

.subckt BUF IN OUT
Xinv1 IN INV INV
Xinv2 INV OUT INV
.ends

.subckt NOR2 IN1 IN2 OUT
Mp1 N1 IN1 VDD VDD p_18 l=0.18u w=0.72u
Mp2 OUT IN2 N1 VDD p_18 l=0.18u w=0.72u
Mn1 OUT IN1 GND GND n_18 l=0.18u w=0.36u
Mn2 OUT IN2 GND GND n_18 l=0.18u w=0.36u
.ends

.subckt NAND2 IN1 IN2 OUT
Mp1 OUT IN1 VDD VDD p_18 l=0.18u w=0.72u
Mp2 OUT IN2 VDD VDD p_18 l=0.18u w=0.72u
Mn1 N1 IN1 GND GND n_18 l=0.18u w=0.36u
Mn2 OUT IN2 N1 GND n_18 l=0.18u w=0.36u
.ends

.subckt AND2 IN1 IN2 OUT
Xnand2 IN1 IN2 N1 NAND2
Xinv N1 OUT INV
.ends

.subckt OR2 IN1 IN2 OUT
Xnor2 IN1 IN2 N1 NOR2
Xinv N1 OUT INV
.ends

.subckt TG SN SP IN OUT * SN: NMOS gate, SP: PMOS gate; SP usually = ~SN
Mp1 OUT SP IN VDD p_18 l=0.18u w=0.72u
Mn1 OUT SN IN GND n_18 l=0.18u w=0.36u
.ends

.subckt XOR2 IN1 IN2 OUT
Xinv1 IN1 IN1_INV INV
Xinv2 IN2 IN2_INV INV
Mp1 N1 IN2_INV VDD VDD p_18 l=0.18u w=0.72u
Mp2 OUT IN1 N1 VDD p_18 l=0.18u w=0.72u
Mp3 N2 IN2 VDD VDD p_18 l=0.18u w=0.72u
Mp4 OUT IN1_INV N2 VDD p_18 l=0.18u w=0.72u
Mn1 N3 IN1 GND GND n_18 l=0.18u w=0.36u
Mn2 OUT IN2 N3 GND n_18 l=0.18u w=0.36u
Mn3 N4 IN1_INV GND GND n_18 l=0.18u w=0.36u
Mn4 OUT IN2_INV N4 GND n_18 l=0.18u w=0.36u
```

```
.ends


************************
*** Bit Multiplier ***
************************
.subckt BIT_MUL IN1 IN2 OUT
Xand IN1 IN2 OUT AND2
.ends

************************
*** 4-bit Multiplier ***
************************
.subckt MUL4 W0 W1 W2 W3 IN O0 O1 O2 O3
Xmul1 IN W0 O0 BIT_MUL
Xmul2 IN W1 O1 BIT_MUL
Xmul3 IN W2 O2 BIT_MUL
Xmul4 IN W3 O3 BIT_MUL
.ends

************************
*** Half-Adder ***
************************
.subckt HA A B SUM CARRY
Xxor A B SUM XOR2
Xand A B CARRY AND2
.ends

************************
*** Full-Adder ***
************************
.subckt FA A B CIN SUM COUT
Xxor1 A B N1 XOR2
Xxor2 N1 CIN SUM XOR2
Xand1 A B N2 AND2
Xand2 N1 CIN N3 AND2
Xor N2 N3 COUT OR2
.ends
```

I switched to Non-pass transitor XOR after I found out the voltage drop and delay will cause the error in bonus computaion. I also choose carry look-ahead adder over ripple carry adder and AND as bit multiplier instead of 2 serial NMOS because of the delay issue. So the Adder and TG here have actually not been used.

Adders

```
*************************
*** 4-bit Carry Look-Ahead Adder ***
*************************
.subckt ADDER4 CIN A0 A1 A2 A3 B0 B1 B2 B3 S0 S1 S2 S3 COUT

* Generate (G) and Propagate (P) logic
Xg0 A0 B0 G0 AND2
Xp0 A0 B0 P0 XOR2
Xg1 A1 B1 G1 AND2
Xp1 A1 B1 P1 XOR2
Xg2 A2 B2 G2 AND2
Xp2 A2 B2 P2 XOR2
Xg3 A3 B3 G3 AND2
Xp3 A3 B3 P3 XOR2

* Carry calculation
Xc1 G0 P0 CIN C1 CLA_CARRY
Xc2 G1 P1 C1 C2 CLA_CARRY
Xc3 G2 P2 C2 C3 CLA_CARRY
Xcout G3 P3 C3 COUT CLA_CARRY

* Sum logic
Xsum0 A0 B0 CIN S0 CLA_SUM
Xsum1 A1 B1 C1 S1 CLA_SUM
Xsum2 A2 B2 C2 S2 CLA_SUM
Xsum3 A3 B3 C3 S3 CLA_SUM

.ends


*************************
*** 5-bit Carry Look-Ahead Adder ***
*************************
.subckt ADDER5 CIN A0 A1 A2 A3 A4 B0 B1 B2 B3 B4 S0 S1 S2 S3 S4 COUT

* Generate (G) and Propagate (P) logic
Xg0 A0 B0 G0 AND2
Xp0 A0 B0 P0 XOR2
Xg1 A1 B1 G1 AND2
Xp1 A1 B1 P1 XOR2
Xg2 A2 B2 G2 AND2
Xp2 A2 B2 P2 XOR2
Xg3 A3 B3 G3 AND2
Xp3 A3 B3 P3 XOR2
Xg4 A4 B4 G4 AND2
Xp4 A4 B4 P4 XOR2

* Carry calculation
Xc1 G0 P0 CIN C1 CLA_CARRY
Xc2 G1 P1 C1 C2 CLA_CARRY
Xc3 G2 P2 C2 C3 CLA_CARRY
Xc4 G3 P3 C3 C4 CLA_CARRY
Xcout G4 P4 C4 COUT CLA_CARRY
```

```
* Sum logic
Xsum0 A0 B0 CIN S0 CLA_SUM
Xsum1 A1 B1 C1 S1 CLA_SUM
Xsum2 A2 B2 C2 S2 CLA_SUM
Xsum3 A3 B3 C3 S3 CLA_SUM
Xsum4 A4 B4 C4 S4 CLA_SUM

.ends

************************
*** CLA_CARRY ***
************************
.subckt CLA_CARRY G P CIN COUT
Xand1 P CIN N1 AND2
Xor1 G N1 COUT OR2
.ends

************************
*** CLA_SUM ***
************************
.subckt CLA_SUM A B CIN S
Xxor1 A B N1 XOR2
Xxor2 N1 CIN S XOR2
.ends

************************
*** 6-bit Adder ***
************************
.subckt ADDER6 CIN A0 A1 A2 A3 A4 A5 B0 B1 B2 B3 B4 B5 S0 S1 S2 S3 S4 S5 COUT

* Generate (G) and Propagate (P) logic
Xg0 A0 B0 G0 AND2
Xp0 A0 B0 P0 XOR2
Xg1 A1 B1 G1 AND2
Xp1 A1 B1 P1 XOR2
Xg2 A2 B2 G2 AND2
Xp2 A2 B2 P2 XOR2
Xg3 A3 B3 G3 AND2
Xp3 A3 B3 P3 XOR2
Xg4 A4 B4 G4 AND2
Xp4 A4 B4 P4 XOR2
Xg5 A5 B5 G5 AND2
Xp5 A5 B5 P5 XOR2

* Carry calculation
Xc1 G0 P0 CIN C1 CLA_CARRY
Xc2 G1 P1 C1 C2 CLA_CARRY
Xc3 G2 P2 C2 C3 CLA_CARRY
Xc4 G3 P3 C3 C4 CLA_CARRY
Xc5 G4 P4 C4 C5 CLA_CARRY
Xcout G5 P5 C5 COUT CLA_CARRY

* Sum logic
Xsum0 A0 B0 CIN S0 CLA_SUM
Xsum1 A1 B1 C1 S1 CLA_SUM
Xsum2 A2 B2 C2 S2 CLA_SUM
```

```
Xsum3 A3 B3 C3 S3 CLA_SUM
Xsum4 A4 B4 C4 S4 CLA_SUM
Xsum5 A5 B5 C5 S5 CLA_SUM

.ends

************************
*** 7-bit Adder ***
************************
.subckt ADDER7 CIN A0 A1 A2 A3 A4 A5 A6 B0 B1 B2 B3 B4 B5 B6 S0 S1 S2 S3 S4 S5 S6

* Generate (G) and Propagate (P) logic
Xg0 A0 B0 G0 AND2
Xp0 A0 B0 P0 XOR2
Xg1 A1 B1 G1 AND2
Xp1 A1 B1 P1 XOR2
Xg2 A2 B2 G2 AND2
Xp2 A2 B2 P2 XOR2
Xg3 A3 B3 G3 AND2
Xp3 A3 B3 P3 XOR2
Xg4 A4 B4 G4 AND2
Xp4 A4 B4 P4 XOR2
Xg5 A5 B5 G5 AND2
Xp5 A5 B5 P5 XOR2
Xg6 A6 B6 G6 AND2
Xp6 A6 B6 P6 XOR2

* Carry calculation
Xc1 G0 P0 CIN C1 CLA_CARRY
Xc2 G1 P1 C1 C2 CLA_CARRY
Xc3 G2 P2 C2 C3 CLA_CARRY
Xc4 G3 P3 C3 C4 CLA_CARRY
Xc5 G4 P4 C4 C5 CLA_CARRY
Xc6 G5 P5 C5 C6 CLA_CARRY
Xcout G6 P6 C6 COUT CLA_CARRY

* Sum logic
Xsum0 A0 B0 CIN S0 CLA_SUM
Xsum1 A1 B1 C1 S1 CLA_SUM
Xsum2 A2 B2 C2 S2 CLA_SUM
Xsum3 A3 B3 C3 S3 CLA_SUM
Xsum4 A4 B4 C4 S4 CLA_SUM
Xsum5 A5 B5 C5 S5 CLA_SUM
Xsum6 A6 B6 C6 S6 CLA_SUM

.ends

************************
*** 8-bit Adder ***
************************
.subckt ADDER8 CIN A0 A1 A2 A3 A4 A5 A6 A7 B0 B1 B2 B3 B4 B5 B6 B7 S0 S1 S2 S3 S4

* Generate (G) and Propagate (P) logic
Xg0 A0 B0 G0 AND2
Xp0 A0 B0 P0 XOR2
Xg1 A1 B1 G1 AND2
```

```
Xp1 A1 B1 P1 XOR2
Xg2 A2 B2 G2 AND2
Xp2 A2 B2 P2 XOR2
Xg3 A3 B3 G3 AND2
Xp3 A3 B3 P3 XOR2
Xg4 A4 B4 G4 AND2
Xp4 A4 B4 P4 XOR2
Xg5 A5 B5 G5 AND2
Xp5 A5 B5 P5 XOR2
Xg6 A6 B6 G6 AND2
Xp6 A6 B6 P6 XOR2
Xg7 A7 B7 G7 AND2
Xp7 A7 B7 P7 XOR2

* Carry calculation
Xc1 G0 P0 CIN C1 CLA_CARRY
Xc2 G1 P1 C1 C2 CLA_CARRY
Xc3 G2 P2 C2 C3 CLA_CARRY
Xc4 G3 P3 C3 C4 CLA_CARRY
Xc5 G4 P4 C4 C5 CLA_CARRY
Xc6 G5 P5 C5 C6 CLA_CARRY
Xc7 G6 P6 C6 C7 CLA_CARRY
Xcout G7 P7 C7 COUT CLA_CARRY

* Sum logic
Xsum0 A0 B0 CIN S0 CLA_SUM
Xsum1 A1 B1 C1 S1 CLA_SUM
Xsum2 A2 B2 C2 S2 CLA_SUM
Xsum3 A3 B3 C3 S3 CLA_SUM
Xsum4 A4 B4 C4 S4 CLA_SUM
Xsum5 A5 B5 C5 S5 CLA_SUM
Xsum6 A6 B6 C6 S6 CLA_SUM
Xsum7 A7 B7 C7 S7 CLA_SUM

.ends

*************************
*** 9-bit Carry Look-Ahead Adder ***
*************************
.subckt ADDER9 CIN A0 A1 A2 A3 A4 A5 A6 A7 A8 B0 B1 B2 B3 B4 B5 B6 B7 B8 S0 S1 S2

* Generate (G) and Propagate (P) logic
Xg0 A0 B0 G0 AND2
Xp0 A0 B0 P0 XOR2
Xg1 A1 B1 G1 AND2
Xp1 A1 B1 P1 XOR2
Xg2 A2 B2 G2 AND2
Xp2 A2 B2 P2 XOR2
Xg3 A3 B3 G3 AND2
Xp3 A3 B3 P3 XOR2
Xg4 A4 B4 G4 AND2
Xp4 A4 B4 P4 XOR2
Xg5 A5 B5 G5 AND2
Xp5 A5 B5 P5 XOR2
Xg6 A6 B6 G6 AND2
Xp6 A6 B6 P6 XOR2
```

```
Xg7 A7 B7 G7 AND2
Xp7 A7 B7 P7 XOR2
Xg8 A8 B8 G8 AND2
Xp8 A8 B8 P8 XOR2

* Carry calculation
Xc1 G0 P0 CIN C1 CLA_CARRY
Xc2 G1 P1 C1 C2 CLA_CARRY
Xc3 G2 P2 C2 C3 CLA_CARRY
Xc4 G3 P3 C3 C4 CLA_CARRY
Xc5 G4 P4 C4 C5 CLA_CARRY
Xc6 G5 P5 C5 C6 CLA_CARRY
Xc7 G6 P6 C6 C7 CLA_CARRY
Xc8 G7 P7 C7 C8 CLA_CARRY
Xcout G8 P8 C8 COUT CLA_CARRY

* Sum logic
Xsum0 A0 B0 CIN S0 CLA_SUM
Xsum1 A1 B1 C1 S1 CLA_SUM
Xsum2 A2 B2 C2 S2 CLA_SUM
Xsum3 A3 B3 C3 S3 CLA_SUM
Xsum4 A4 B4 C4 S4 CLA_SUM
Xsum5 A5 B5 C5 S5 CLA_SUM
Xsum6 A6 B6 C6 S6 CLA_SUM
Xsum7 A7 B7 C7 S7 CLA_SUM
Xsum8 A8 B8 C8 S8 CLA_SUM

.ends

*************************
*** 12-bit Carry Look-Ahead Adder ***
*************************
.subckt ADDER12 CIN A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 B0 B1 B2 B3 B4 B5 B6 B7

* Generate (G) and Propagate (P) logic
Xg0 A0 B0 G0 AND2
Xp0 A0 B0 P0 XOR2
Xg1 A1 B1 G1 AND2
Xp1 A1 B1 P1 XOR2
Xg2 A2 B2 G2 AND2
Xp2 A2 B2 P2 XOR2
Xg3 A3 B3 G3 AND2
Xp3 A3 B3 P3 XOR2
Xg4 A4 B4 G4 AND2
Xp4 A4 B4 P4 XOR2
Xg5 A5 B5 G5 AND2
Xp5 A5 B5 P5 XOR2
Xg6 A6 B6 G6 AND2
Xp6 A6 B6 P6 XOR2
Xg7 A7 B7 G7 AND2
Xp7 A7 B7 P7 XOR2
Xg8 A8 B8 G8 AND2
Xp8 A8 B8 P8 XOR2
Xg9 A9 B9 G9 AND2
Xp9 A9 B9 P9 XOR2
Xg10 A10 B10 G10 AND2
```

```
Xp10 A10 B10 P10 XOR2
Xg11 A11 B11 G11 AND2
Xp11 A11 B11 P11 XOR2

* Carry calculation
Xc1 G0 P0 CIN C1 CLA_CARRY
Xc2 G1 P1 C1 C2 CLA_CARRY
Xc3 G2 P2 C2 C3 CLA_CARRY
Xc4 G3 P3 C3 C4 CLA_CARRY
Xc5 G4 P4 C4 C5 CLA_CARRY
Xc6 G5 P5 C5 C6 CLA_CARRY
Xc7 G6 P6 C6 C7 CLA_CARRY
Xc8 G7 P7 C7 C8 CLA_CARRY
Xc9 G8 P8 C8 C9 CLA_CARRY
Xc10 G9 P9 C9 C10 CLA_CARRY
Xc11 G10 P10 C10 C11 CLA_CARRY
Xcout G11 P11 C11 COUT CLA_CARRY

* Sum logic
Xsum0 A0 B0 CIN S0 CLA_SUM
Xsum1 A1 B1 C1 S1 CLA_SUM
Xsum2 A2 B2 C2 S2 CLA_SUM
Xsum3 A3 B3 C3 S3 CLA_SUM
Xsum4 A4 B4 C4 S4 CLA_SUM
Xsum5 A5 B5 C5 S5 CLA_SUM
Xsum6 A6 B6 C6 S6 CLA_SUM
Xsum7 A7 B7 C7 S7 CLA_SUM
Xsum8 A8 B8 C8 S8 CLA_SUM
Xsum9 A9 B9 C9 S9 CLA_SUM
Xsum10 A10 B10 C10 S10 CLA_SUM
Xsum11 A11 B11 C11 S11 CLA_SUM

.ends
```

I have implemented carry look-ahead adders for many different bit scenarios, most of them are for bonus.

Adder trees

```
*************************
*** 4 of 4 bit Adder Tree ***
*************************
.subckt ADDER_TREE4 CIN I10 I11 I12 I13 I20 I21 I22 I23 I30 I31 I32 I33 I40 I41 I
Xadder41 CIN I10 I11 I12 I13 I20 I21 I22 I23 TP10 TP11 TP12 TP13 TP14 ADDER4
Xadder42 CIN I30 I31 I32 I33 I40 I41 I42 I43 TP20 TP21 TP22 TP23 TP24 ADDER4
Xadder5 CIN TP10 TP11 TP12 TP13 TP14 TP20 TP21 TP22 TP23 TP24 S0 S1 S2 S3 S4 S5 A
.ends

*************************
*** 32 of 4 bit Adder Tree ***
.subckt ADDER_TREE32 CIN I10 I11 I12 I13 I20 I21 I22 I23 I30 I31 I32 I33 I40 I41
Xadder41 CIN I10 I11 I12 I13 I20 I21 I22 I23 TP10_1 TP11_1 TP12_1 TP13_1 TP14_1 A
Xadder42 CIN I30 I31 I32 I33 I40 I41 I42 I43 TP20_1 TP21_1 TP22_1 TP23_1 TP24_1 A
Xadder43 CIN I50 I51 I52 I53 I60 I61 I62 I63 TP30_1 TP31_1 TP32_1 TP33_1 TP34_1 A
Xadder44 CIN I70 I71 I72 I73 I80 I81 I82 I83 TP40_1 TP41_1 TP42_1 TP43_1 TP44_1 A
Xadder45 CIN I90 I91 I92 I93 I100 I101 I102 I103 TP50_1 TP51_1 TP52_1 TP53_1 TP5
Xadder46 CIN I110 I111 I112 I113 I120 I121 I122 I123 TP60_1 TP61_1 TP62_1 TP63_1
Xadder47 CIN I130 I131 I132 I133 I140 I141 I142 I143 TP70_1 TP71_1 TP72_1 TP73_1
Xadder48 CIN I150 I151 I152 I153 I160 I161 I162 I163 TP80_1 TP81_1 TP82_1 TP83_1
Xadder49 CIN I170 I171 I172 I173 I180 I181 I182 I183 TP90_1 TP91_1 TP92_1 TP93_1
Xadder410 CIN I190 I191 I192 I193 I200 I201 I202 I203 TP100_1 TP101_1 TP102_1 TP1
Xadder411 CIN I210 I211 I212 I213 I220 I221 I222 I223 TP110_1 TP111_1 TP112_1 TP1
Xadder412 CIN I230 I231 I232 I233 I240 I241 I242 I243 TP120_1 TP121_1 TP122_1 TP1
Xadder413 CIN I250 I251 I252 I253 I260 I261 I262 I263 TP130_1 TP131_1 TP132_1 TP1
Xadder414 CIN I270 I271 I272 I273 I280 I281 I282 I283 TP140_1 TP141_1 TP142_1 TP1
Xadder415 CIN I290 I291 I292 I293 I300 I301 I302 I303 TP150_1 TP151_1 TP152_1 TP1
Xadder416 CIN I310 I311 I312 I313 I320 I321 I322 I323 TP160_1 TP161_1 TP162_1 TP1
Xadder51 CIN TP10_1 TP11_1 TP12_1 TP13_1 TP14_1 TP20_1 TP21_1 TP22_1 TP23_1 TP24_
Xadder52 CIN TP30_1 TP31_1 TP32_1 TP33_1 TP34_1 TP40_1 TP41_1 TP42_1 TP43_1 TP44_
Xadder53 CIN TP50_1 TP51_1 TP52_1 TP53_1 TP54_1 TP60_1 TP61_1 TP62_1 TP63_1 TP64_
Xadder54 CIN TP70_1 TP71_1 TP72_1 TP73_1 TP74_1 TP80_1 TP81_1 TP82_1 TP83_1 TP84_
Xadder55 CIN TP90_1 TP91_1 TP92_1 TP93_1 TP94_1 TP100_1 TP101_1 TP102_1 TP103_1 T
Xadder56 CIN TP110_1 TP111_1 TP112_1 TP113_1 TP114_1 TP120_1 TP121_1 TP122_1 TP12
Xadder57 CIN TP130_1 TP131_1 TP132_1 TP133_1 TP134_1 TP140_1 TP141_1 TP142_1 TP14
Xadder58 CIN TP150_1 TP151_1 TP152_1 TP153_1 TP154_1 TP160_1 TP161_1 TP162_1 TP16
Xadder61 CIN TP10_2 TP11_2 TP12_2 TP13_2 TP14_2 TP15_2 TP20_2 TP21_2 TP22_2 TP23_
Xadder62 CIN TP30_2 TP31_2 TP32_2 TP33_2 TP34_2 TP35_2 TP40_2 TP41_2 TP42_2 TP43_
Xadder63 CIN TP50_2 TP51_2 TP52_2 TP53_2 TP54_2 TP55_2 TP60_2 TP61_2 TP62_2 TP63_
Xadder64 CIN TP70_2 TP71_2 TP72_2 TP73_2 TP74_2 TP75_2 TP80_2 TP81_2 TP82_2 TP83_
Xadder71 CIN TP10_3 TP11_3 TP12_3 TP13_3 TP14_3 TP15_3 TP16_3 TP20_3 TP21_3 TP22_
Xadder72 CIN TP30_3 TP31_3 TP32_3 TP33_3 TP34_3 TP35_3 TP36_3 TP40_3 TP41_3 TP42_
Xadder81 CIN TP10_4 TP11_4 TP12_4 TP13_4 TP14_4 TP15_4 TP16_4 TP17_4 TP20_4 TP21_
.ends
```

The adder trees for both final and bonus part, resulting 6 bits and 9 bits respectively.

Compute Cell

```
************************
*** Compute Cell ***
.subckt COMPUTE I1 I2 I3 I4 W10 W11 W12 W13 W20 W21 W22 W23 W30 W31 W32 W33 W40 W
Xmul41 W10 W11 W12 W13 I1 O10 O11 O12 O13 MUL4
Xmul42 W20 W21 W22 W23 I2 O20 O21 O22 O23 MUL4
Xmul43 W30 W31 W32 W33 I3 O30 O31 O32 O33 MUL4
Xmul44 W40 W41 W42 W43 I4 O40 O41 O42 O43 MUL4
Xadder_tree1 GND O10 O11 O12 O13 O20 O21 O22 O23 O30 O31 O32 O33 O40 O41 O42 O43
.ends

************************
*** Compute Cell 32 * 4 bit***
.subckt COMPUTE I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 I12 I13 I14 I15 I16 I17 I18 I1
Xmul41 W10 W11 W12 W13 I1 O10 O11 O12 O13 MUL4
Xmul42 W20 W21 W22 W23 I2 O20 O21 O22 O23 MUL4
Xmul43 W30 W31 W32 W33 I3 O30 O31 O32 O33 MUL4
Xmul44 W40 W41 W42 W43 I4 O40 O41 O42 O43 MUL4
Xmul45 W50 W51 W52 W53 I5 O50 O51 O52 O53 MUL4
Xmul46 W60 W61 W62 W63 I6 O60 O61 O62 O63 MUL4
Xmul47 W70 W71 W72 W73 I7 O70 O71 O72 O73 MUL4
Xmul48 W80 W81 W82 W83 I8 O80 O81 O82 O83 MUL4
Xmul49 W90 W91 W92 W93 I9 O90 O91 O92 O93 MUL4
Xmul410 W100 W101 W102 W103 I10 O100 O101 O102 O103 MUL4
Xmul411 W110 W111 W112 W113 I11 O110 O111 O112 O113 MUL4
Xmul412 W120 W121 W122 W123 I12 O120 O121 O122 O123 MUL4
Xmul413 W130 W131 W132 W133 I13 O130 O131 O132 O133 MUL4
Xmul414 W140 W141 W142 W143 I14 O140 O141 O142 O143 MUL4
Xmul415 W150 W151 W152 W153 I15 O150 O151 O152 O153 MUL4
Xmul416 W160 W161 W162 W163 I16 O160 O161 O162 O163 MUL4
Xmul417 W170 W171 W172 W173 I17 O170 O171 O172 O173 MUL4
Xmul418 W180 W181 W182 W183 I18 O180 O181 O182 O183 MUL4
Xmul419 W190 W191 W192 W193 I19 O190 O191 O192 O193 MUL4
Xmul420 W200 W201 W202 W203 I20 O200 O201 O202 O203 MUL4
Xmul421 W210 W211 W212 W213 I21 O210 O211 O212 O213 MUL4
Xmul422 W220 W221 W222 W223 I22 O220 O221 O222 O223 MUL4
Xmul423 W230 W231 W232 W233 I23 O230 O231 O232 O233 MUL4
Xmul424 W240 W241 W242 W243 I24 O240 O241 O242 O243 MUL4
Xmul425 W250 W251 W252 W253 I25 O250 O251 O252 O253 MUL4
Xmul426 W260 W261 W262 W263 I26 O260 O261 O262 O263 MUL4
Xmul427 W270 W271 W272 W273 I27 O270 O271 O272 O273 MUL4
Xmul428 W280 W281 W282 W283 I28 O280 O281 O282 O283 MUL4
Xmul429 W290 W291 W292 W293 I29 O290 O291 O292 O293 MUL4
Xmul430 W300 W301 W302 W303 I30 O300 O301 O302 O303 MUL4
Xmul431 W310 W311 W312 W313 I31 O310 O311 O312 O313 MUL4
Xmul432 W320 W321 W322 W323 I32 O320 O321 O322 O323 MUL4
Xadder_1 GND O10 O11 O12 O13 O20 O21 O22 O23 O30 O31 O32 O33 O40 O41 O42 O43 O50
.ends
```

Memories

```
************************
*** D latch ***
.subckt DLATCH D ENB Q Q_INV
Xinv D D_INV INV
Xnand21 D ENB N1 NAND2
Xnand22 D_INV ENB N2 NAND2
Xnand23 N1 Q_INV Q NAND2
Xnand24 N2 Q Q_INV NAND2
.ends

************************
*** D flip-flop ***
************************
.subckt DFF D CLK RST Q Q_INV
Xinv1 CLK CLK_INV INV
Mn1 N1 CLK_INV D GND n_18 l=0.18u w=0.36u
Xnand1 N1 RST N2 NAND2
Xinv2 N2 N3 INV
Mn2 N1 CLK N3 GND n_18 l=0.18u w=0.36u
Mn3 N4 CLK N2 GND n_18 l=0.18u w=0.36u
Xinv3 N4 Q INV
Xnand2 Q RST Q_INV NAND2
Mn4 N4 CLK_INV Q_INV GND n_18 l=0.18u w=0.36u
.ends

************************
*** 10-bit Register ***
************************
.subckt REG10 I0 I1 I2 I3 I4 I5 I6 I7 I8 I9 CLK ENB O0 O1 O2 O3 O4 O5 O6 O7 O8 O9
Xdff1 I0 CLK ENB O0 O0_INV DFF
Xdff2 I1 CLK ENB O1 O1_INV DFF
Xdff3 I2 CLK ENB O2 O2_INV DFF
Xdff4 I3 CLK ENB O3 O3_INV DFF
Xdff5 I4 CLK ENB O4 O4_INV DFF
Xdff6 I5 CLK ENB O5 O5_INV DFF
Xdff7 I6 CLK ENB O6 O6_INV DFF
Xdff8 I7 CLK ENB O7 O7_INV DFF
Xdff9 I8 CLK ENB O8 O8_INV DFF
Xdff10 I9 CLK ENB O9 O9_INV DFF
.ends

************************
*** 13-bit Register ***
************************
.subckt REG13 I0 I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 I12 CLK ENB O0 O1 O2 O3 O4 O5
Xdff1 I0 CLK ENB O0 O0_INV DFF
Xdff2 I1 CLK ENB O1 O1_INV DFF
Xdff3 I2 CLK ENB O2 O2_INV DFF
Xdff4 I3 CLK ENB O3 O3_INV DFF
Xdff5 I4 CLK ENB O4 O4_INV DFF
Xdff6 I5 CLK ENB O5 O5_INV DFF
Xdff7 I6 CLK ENB O6 O6_INV DFF
Xdff8 I7 CLK ENB O7 O7_INV DFF
Xdff9 I8 CLK ENB O8 O8_INV DFF
```

```
Xdff10 I9 CLK ENB O9 O9_INV DFF
Xdff11 I10 CLK ENB O10 O10_INV DFF
Xdff12 I11 CLK ENB O11 O11_INV DFF
Xdff13 I12 CLK ENB O12 O12_INV DFF
.ends
```

I chose D-Latch and D-Flip-Flop as my basic memory cells.

Input Controller

```
************************
*** input controller ***
************************
.subckt CON_IN I1 I2 I3 I4 CLK ENB ii1 ii2 ii3 ii4
*** Your code ***
Xinv CLK CLK_INV INV
Xand1 I1 ENB N1 AND2
Xand2 I2 ENB N2 AND2
Xand3 I3 ENB N3 AND2
Xand4 I4 ENB N4 AND2
Xdlatch1_master N1 CLK Y1 Y1_INV DLATCH
Xdlatch1_slave Y1 CLK_INV ii1 ii1_INV DLATCH
Xdlatch2_master N2 CLK Y2 Y2_INV DLATCH
Xdlatch2_slave Y2 CLK_INV ii2 ii2_INV DLATCH
Xdlatch3_master N3 CLK Y3 Y3_INV DLATCH
Xdlatch3_slave Y3 CLK_INV ii3 ii3_INV DLATCH
Xdlatch4_master N4 CLK Y4 Y4_INV DLATCH
Xdlatch4_slave Y4 CLK_INV ii4 ii4_INV DLATCH
.ends
```

Stall the Input for one clock(with enable).

Output Controller

```
*************************
*** output controller ***
*************************
.subckt CON_OUT P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 CLK RST O0 O1 O2 O3 O4 O5 O6 O7 O8
*** Your code ***
Xadder9 GND P0 P1 P2 P3 P4 P5 P6 P7 P8 GND O0 O1 O2 O3 O4 O5 O6 O7 S0 S1 S2 S3 S4
Xreg10 S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 CLK RST O0 O1 O2 O3 O4 O5 O6 O7 O8 O9 REG10
.ends
```

Implement the shift-and-add part.

DCIM

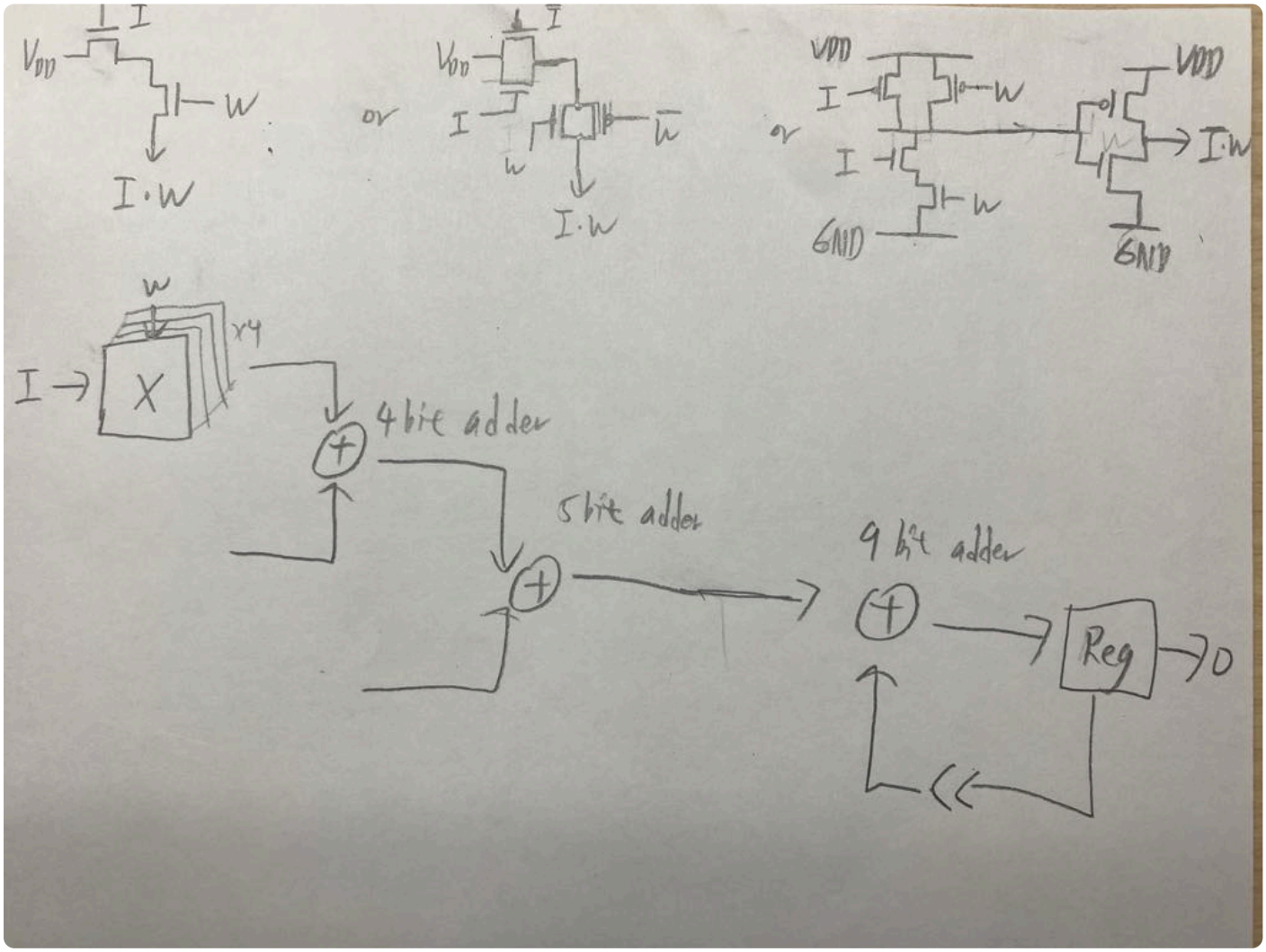
```
******************
*** DCIM block ***
******************
.subckt DCIM I1 I2 I3 I4 IN_VAL CLK RST O10 O11 O12 O13 O14 O15 O16 O17 O18 O19 (

** Your code **

Xcontrol_in I1 I2 I3 I4 CLK IN_VAL ii1 ii2 ii3 ii4 CON_IN
Xcim1 ii1 ii2 ii3 ii4 W110 W111 W112 W113 W120 W121 W122 W123 W130 W131 W132 W133
Xcim2 ii1 ii2 ii3 ii4 W210 W211 W212 W213 W220 W221 W222 W223 W230 W231 W232 W233
Xcontrol_out1 S10 S11 S12 S13 S14 S15 GND GND GND GND CLK RST O10 O11 O12 O13 O14
Xcontrol_out2 S20 S21 S22 S23 S24 S25 GND GND GND GND CLK RST O20 O21 O22 O23 O24
Xdff1 IN_VAL CLK RST IN_VAL_T_1 IN_VAL_T_1_INV DFF
Xdff2 IN_VAL_T_1 CLK RST IN_VAL_T_2 IN_VAL_T_2_INV DFF
Xnor IN_VAL_T_1 IN_VAL_T_2_INV OUT_VAL NOR2
.ends
```

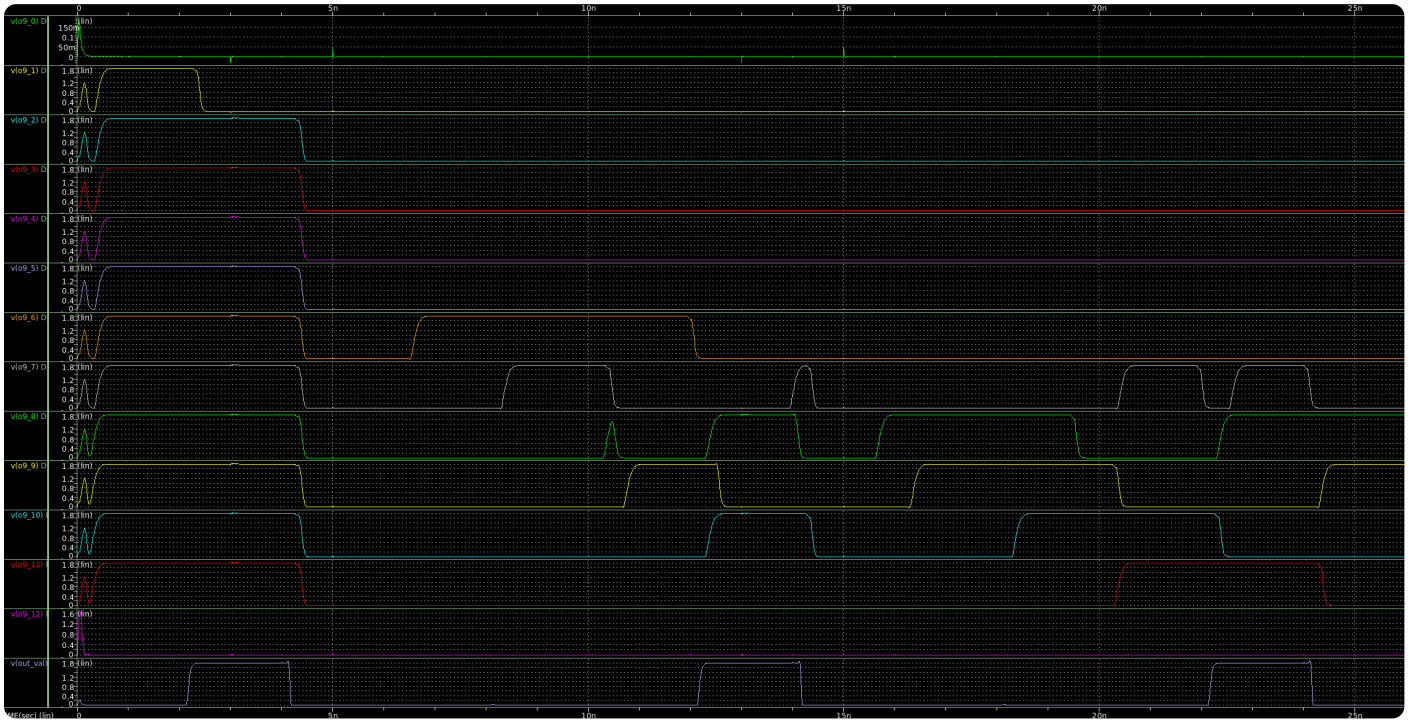
Add two flip-flops to handle the `OUT_VAL`.

The transistor level view of your CIM circuit and adder



The bonus circuit if implemented

I have write more than 3000 line of code for bonus. First, before passing the ii to CIM, I have to add buffers to reduce the fan-out and load, it not, the delay and drop will be significant. And, I have to redesign all adder as carry look-ahead(and add adder from 6-bit to 12 bit). Last, of course, initialize the weights and expand the Input and Ouput(32 * 16 now).



Here I take O9 as example, the weight should be 8, and I1 = I5 = I9 = I13 = I17 = I21 = I25 = I29 = 8, I2 = ... = 2, I3 = ... = 3, I4 = ... = 7. In the waveform, we can see that O9 = 0010100000000 = 1280 for testcase.

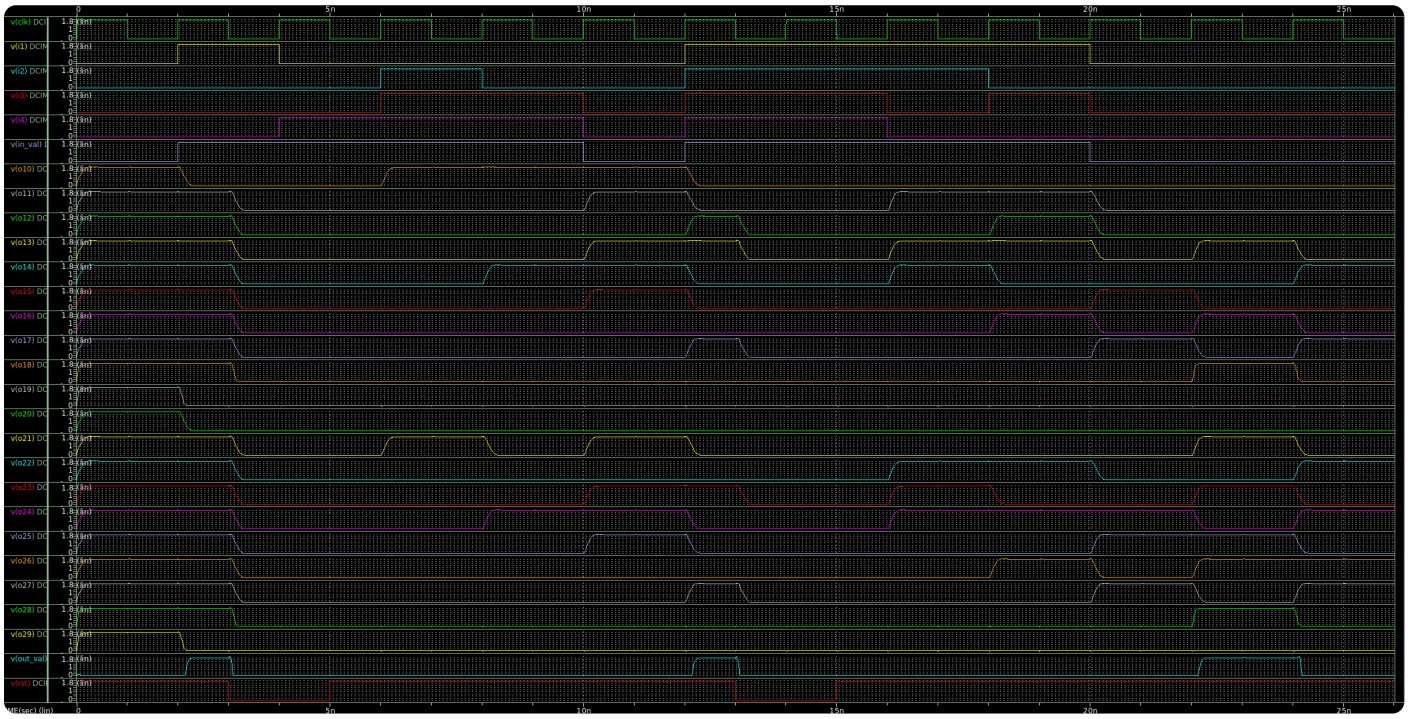
```
******  Circuit Statistics  ******
# nodes       =  426188 # elements   =  167386
# resistors   =       0 # capacitors =       0 # inductors   =       0
# mutual_inds =       0 # vccs       =       0 # vcvs        =       0
# cccs        =       0 # ccvs       =       0 # volt_srcs   =      36
# curr_srcs   =       0 # diodes     =       0 # bjts        =       0
# jfets       =       0 # mosfets    =  167350 # U elements  =       0
# T elements  =       0 # W elements =       0 # B elements  =       0
# S elements  =       0 # P elements =       0 # va device   =       0
# vector_srcs =       0 # N elements =       0
```

```
******
** dcim bonus **

****** transient analysis tnom=  25.000 temp=  30.000 ******
td=   2.4512n  targ=   6.3557n   trig=   3.9045n
pwr=  67.9603m  from=   0.              to=  26.0000n

    ***** job concluded
******
** dcim bonus **
```

Waveform of your simulation



The delay and the power of the circuit

```
****** transient analysis tnom=  25.000 temp=  30.000 ******
td= 751.1333p  targ=   4.0600n   trig=   3.3089n
pwr=   1.6453m  from=   0.              to=  26.0000n
```

The total number of transistors (NMOS and PMOS) you use in this program

```
******  Circuit Statistics  ******
# nodes       =    6654 # elements   =    2618
# resistors   =       0 # capacitors =       0 # inductors   =       0
# mutual_inds =       0 # vccs       =       0 # vcvs        =       0
# cccs        =       0 # ccvs       =       0 # volt_srcs   =       8
# curr_srcs   =       0 # diodes     =       0 # bjts        =       0
# jfets       =       0 # mosfets    =    2610 # U elements  =       0
# T elements  =       0 # W elements =       0 # B elements  =       0
# S elements  =       0 # P elements =       0 # va device   =       0
# vector_srcs =       0 # N elements =       0
```

The hardness of this assignment and how you overcome it

I think the hardest part is comprehensing the whole structure, understanding the requirements. After fully realized it, the remaining part is just implementation, which can be easily followed up if you doing it step by step. The other hardest part(or the most troublesome) is drawing the diagram. I didin't really draw the diagram and transistors before my design, the hspice code just come naturally. So I have to draw it with pens after I finished my Project...

Plus, while designing bonus, it takes so much time for .lis and .tr0 to generate(the hspice -i... command), I don't know whether it's because the circuit is too big or there were too many people using the VLSI server at the same time.

And the delay problem is driving me crazy, I have to re-design lots of my basic gates and circuits, and even implement carry look-ahead adders(lots of adders) to avoid delay corrupting my output.

Any suggestions about this programming assignment

I think it's fun, since I only need to write spice code. However, this project is quite like a logic design project HaHa.