UNIVERSITY OF CALIFORNIA, LOS ANGELES UCLA

BERKELEY · DAVIS · IRVINE · LOS ANGELES · RIVERSIDE · SAN DIEGO · SAN FRANCISCO SANTA BARBARA · SANTA CRUZ

CS M151B / EE M116C

MIDTERM EXAM

All work and answers should be written directly on these pages, use the backs of pages if needed.

This is an open book, open notes quiz – but you cannot share books or notes.

We will follow the departmental guidelines on reporting incidents of academic.
Keep your eyes on your own exam!

NAME: _____

ID: _____

Do not write anything in the area below on this page:

Problem 1: _____ (20)

Problem 2: _____ (5)

Problem 3: _____ (15)

Problem 4: _____ (30)

Problem 5: _____ (30)

Total: _____ (out of 100)

♠

1. ***Performance Anxiety (20 points)***: The Apricot computer features a processor design with a 4 GHz clock. The processor is capable of executing a subset of the MIPS ISA, where Load instructions take 6 cycles to execute, Stores and simple R-types take 5 cycles to execute, and BEQ/BNE instructions take 4 cycles to execute. The Apricot computer executes an application with 30 billion instructions. The application has the following instruction mix.

Instruction	% of Instructions
Load	30%
Store	10%
Simple R-type (i.e. add, and, slt)	45%
BEQ/BNE	15%

a. What is the CPI and ET for this application running on this processor? *You must show your work.*

$CPI = .3 \times 6 + (.1 + .45) \times 5 + .15 \times 4 = 1.8 + 2.75 + .6 =$

CPI: 5.15 (5)

$ET = 30 \times 10^9 \times \frac{5.15}{CPI} \times .25 \times 10^{-9}$

ET: 38.6 s (5)

We are going to add a new instruction type to the processor – the BLT (branch less than). This particular BLT instruction will replace the specific case where an slt instruction is followed by a bne instruction. So

```
slt $1, $8, $9
bne $1, $0, label
```

would become:

```
blt $8, $9, label
```

This has two benefits – first, there is the replacement of two instructions with a single instruction. Second, the register used to communicate the slt result to the bne instruction is no longer required (e.g. the $1 in the example above) – this is a reduction in register file pressure. To evaluate this benefit, assume that a third of all branches are bne instructions that can be converted into a blt instruction. Furthermore, assume that the reduction in register file pressure means that we can reduce the number of loads by 10% and the number of stores by 10% (e.g. less register spilling).

Latency of BLT unspecified on exam - go with their assumption

b. What is the CPI and ET for this application running on this processor **after** the addition of the BLT instruction? *You must show your work.*

1.78 + 2.69 + .66

Load 27
Store 9
BEQ/BNE 10
BLT 5
R 40
91

$CPI = \frac{27}{91} \times 6 + \frac{49}{91} \times 5 + \frac{10+5}{91} \times 4 =$ CPI: 5.13 (5)

$ET = (91)(30 \times 10^9) \times \frac{5.13}{CPI} \times .25 \times 10^{-9}$ ET: 35 s (5)

2. ***Single Cycle Conundrum (5 points)***: Consider the single cycle datapath we covered in class. Suppose that we run an application with the following instruction mix:

Instruction	% of Instructions
Load	35%
Store	5%
Simple R-type (i.e. add, and, slt)	50%
BEQ/BNE	10%

What is the CPI for this application running on this processor?

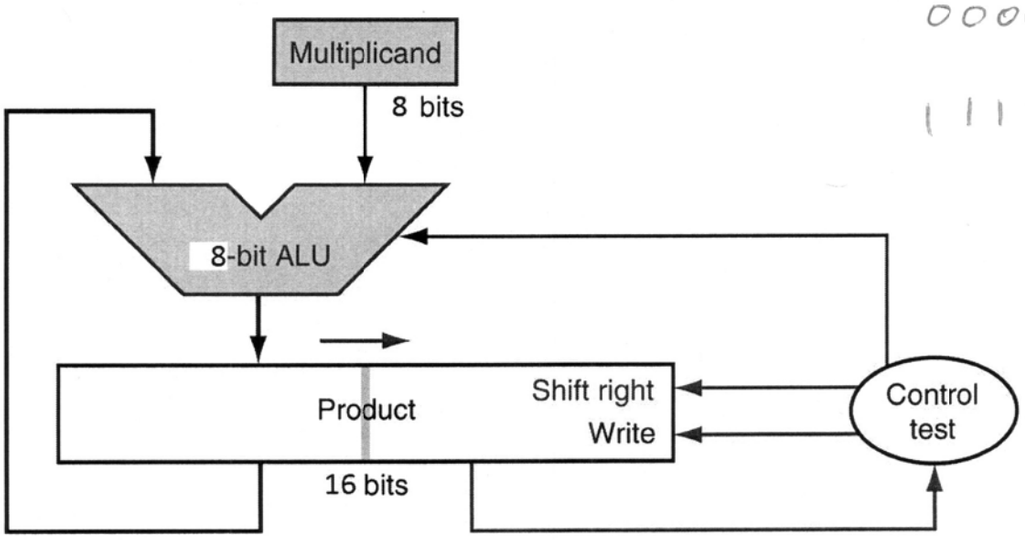
CPI: 1.0 (5)

no partial credit

3. ***Your Problems are Multiplying (15 points)***: Assume the optimized multiplier covered in class, but one that is designed for 8-bit numbers instead of 32-bit numbers:

00000011

11111101



Assume that this multiplier is implemented using Booth's Algorithm, as discussed in class – so the control test follows Booth's Algorithm in performing multiplication. Assume that the multiplicand register initially holds the value -3 and the product register initially holds the value 4. What value will be contained in the product register after the second iteration of Booth's Algorithm is complete (i.e. assume the control has just written the product register at the end of the second iteration)? *Show your work.*

1

PRODUCT BITTR MULTIPLICAND

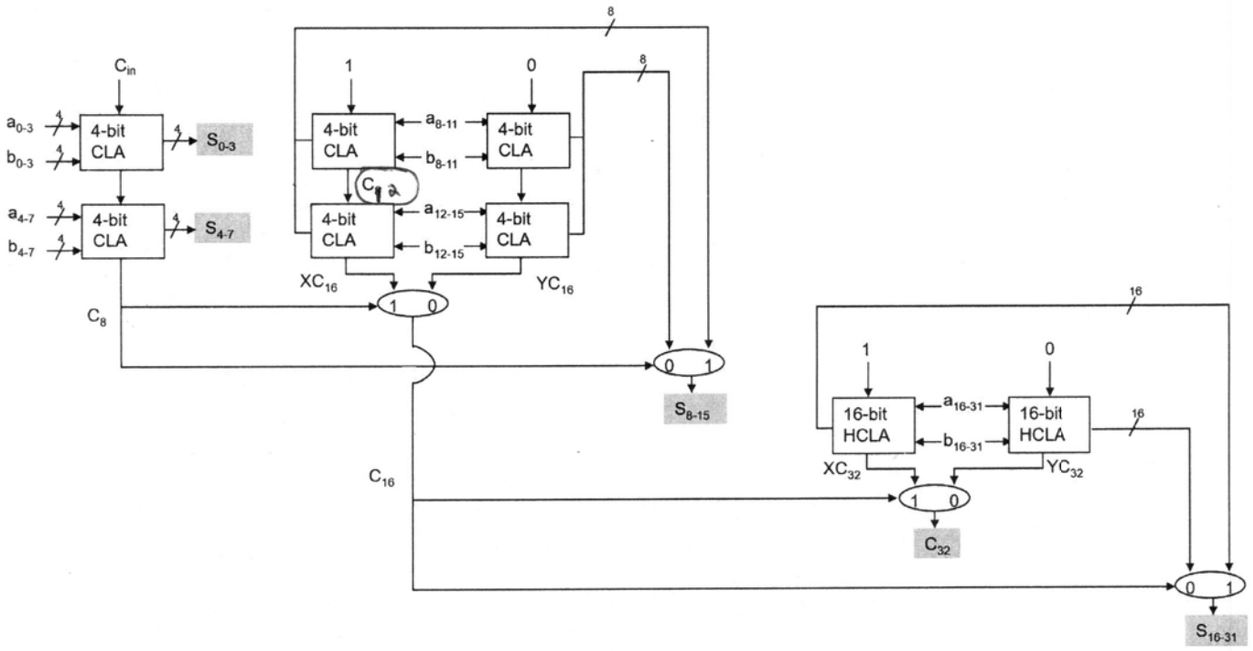
	MULTIPLICAND	PRODUCT	BITTR
Iter.	1111 1101	0000 0000 0000 0100	0
1		0000 0000 0000 0010	0
2		0000 0000 0000 0001	0

4. ***Putting the CLA into UCLA (30 points)***: Assume for the rest of this problem that all logic gates have the following delays:

Fan In	Delay
2	4T
3	6T
4	9T
5	13T
6	17T
7	22T
8	28T

So a 2-input AND gate would have delay 4T and a 4-input OR gate would have delay 9T. For simplicity, assume that mux's have delay 12T regardless of fan-in.

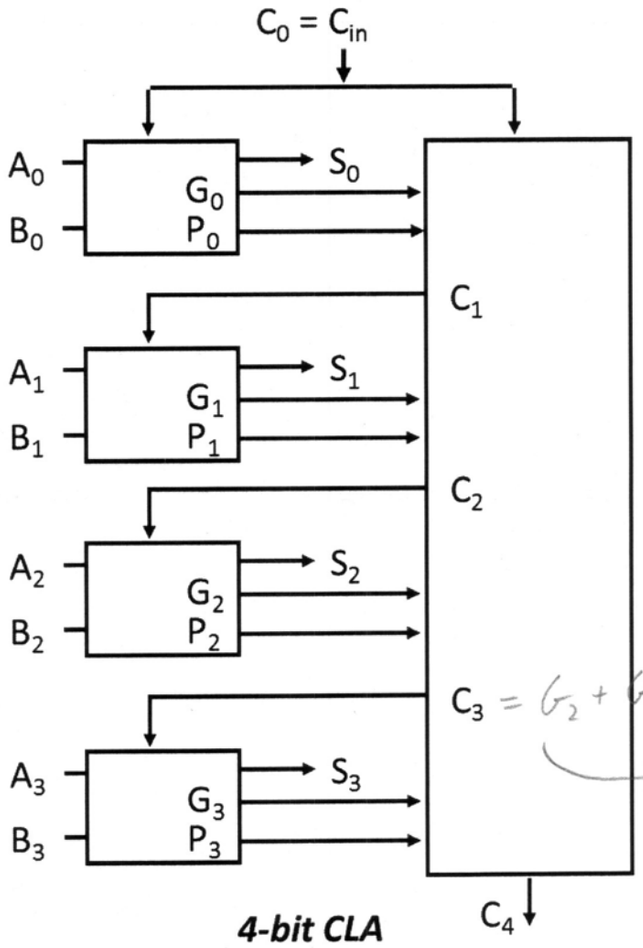
We will create a 32-bit adder out of some building blocks we've covered in class. We will use the 4-bit carry lookahead (4-bit CLA) that we covered in class as a basic building block of this design. And we will use it (as we did in class) to make a 16-bit hierarchical CLA (16-bit HCLA). We will connect 4-bit CLAs and 16-bit HCLAs together in a carry select fashion. The design will look as follows:



The figure shows the outputs of the 32-bit adder, including the Carry Out (C_{32}) and Sum bits S_0 through S_{31}. The individual CLA and HCLA blocks take input from a carry in and the A and B operands.

Your task is to find the maximal delay of this design – i.e. determine the delays of $S_{0\text{-}63}$ and C_{64} – the maximal delay of these outputs will be the maximal delay of the entire design. To do this (and to help with possible partial credit) please use the diagrams on the following pages and fill in the tables in every page. Note that the diagrams are taken from the class notes – and are not necessarily labeled to match the 64-bit adder design.

Single 4-bit CLA:



4-bit CLA

Output	Delay (in terms of T)	
G0	4T	*(1 points)*
P0	4T	*(1 points)*
G3	4T	*(1 points)*
P3	4T	*(1 points)*
C3	22T	*(2 points)*
S3	26T	*(1 points)*
C4	30T	*(2 points)*

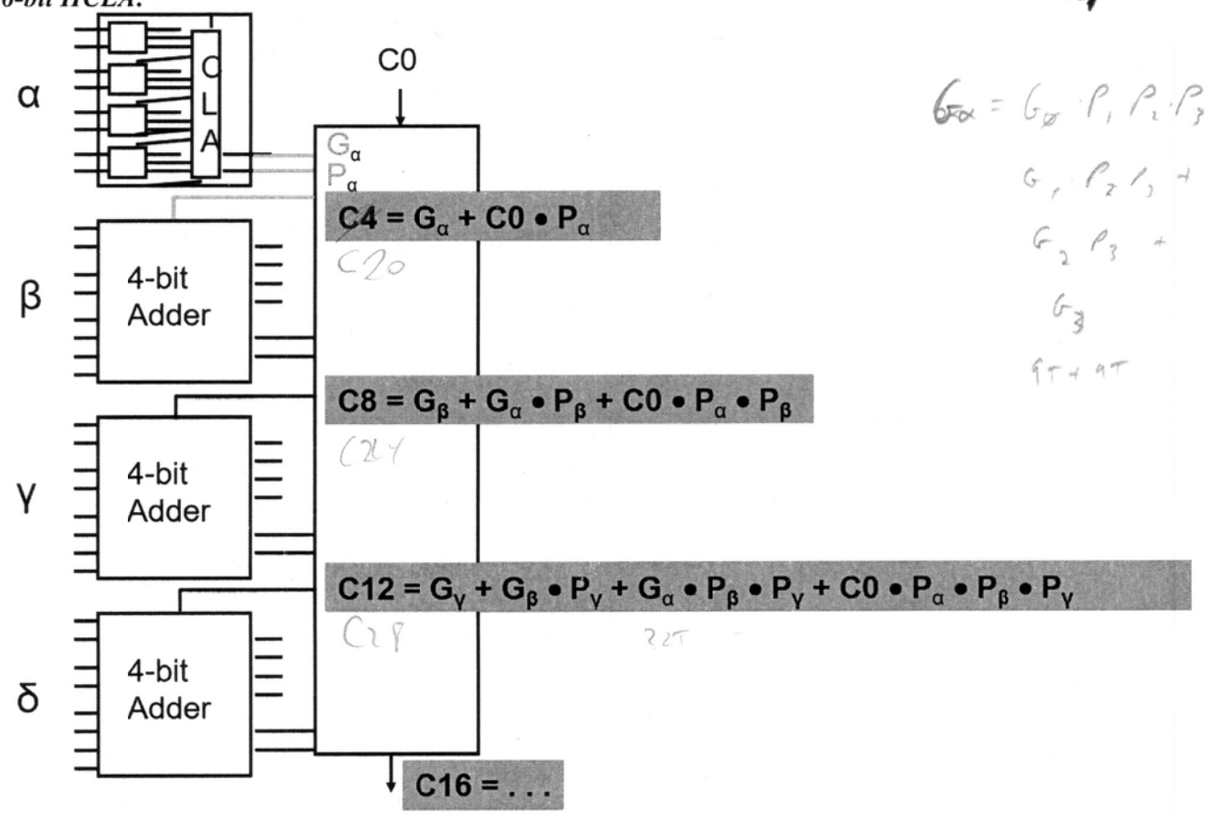
18T + Pø
C3 + 4T or 6T depending on assumptio
26T + Pø

misleading! could be same as C4 with right assumption

Output	Delay (in terms of T)	
G8	4T	*(1 points)*
P8	4T	*(1 points)*
G12	4T	*(1 points)*
P12	4T	*(1 points)*
C8	56T	*(2 points)* 26T + C4
C16	68T	*(2 points)* 12T + C8
S15	64T	*(2 points)* 12T + (4T or 6T) + 18T + C4

OR Could be C_8 and C_{16} and S_{15} from

↳ ***16-bit HCLA:***



$G_{\alpha} = G_0 \cdot P_1 P_2 \cdot P_3 +$
$G_1 P_2 P_3 +$
$G_2 P_3 +$
G_3
9T + 9T

Output	Delay (in terms of T)	
G_δ	22T	*(1 points)*
P_δ	13T	*(1 points)*
C28	47T	*(2 points)*
C31	65T	*(2 points)*
C32	80T	*(2 points)*
S31	81T	*(2 points)*

18T + G_0
9T + P_0
6T + 9T + G_5
9T + 9T + C_{28}
C_{16} + 12T

60T = 12T + 26T + G_5 ←
12T + (4T OR 6T) + C31 ← = 81T

Find the maximum delay **in terms of T** of the 64-bit adder – take the maximum of all output bits – including the sum bits (S_0-S_{31}) and the final carry out (C_{32}).

Maximal Delay: 81T *(1 points)*

Correctly takes max of all delays

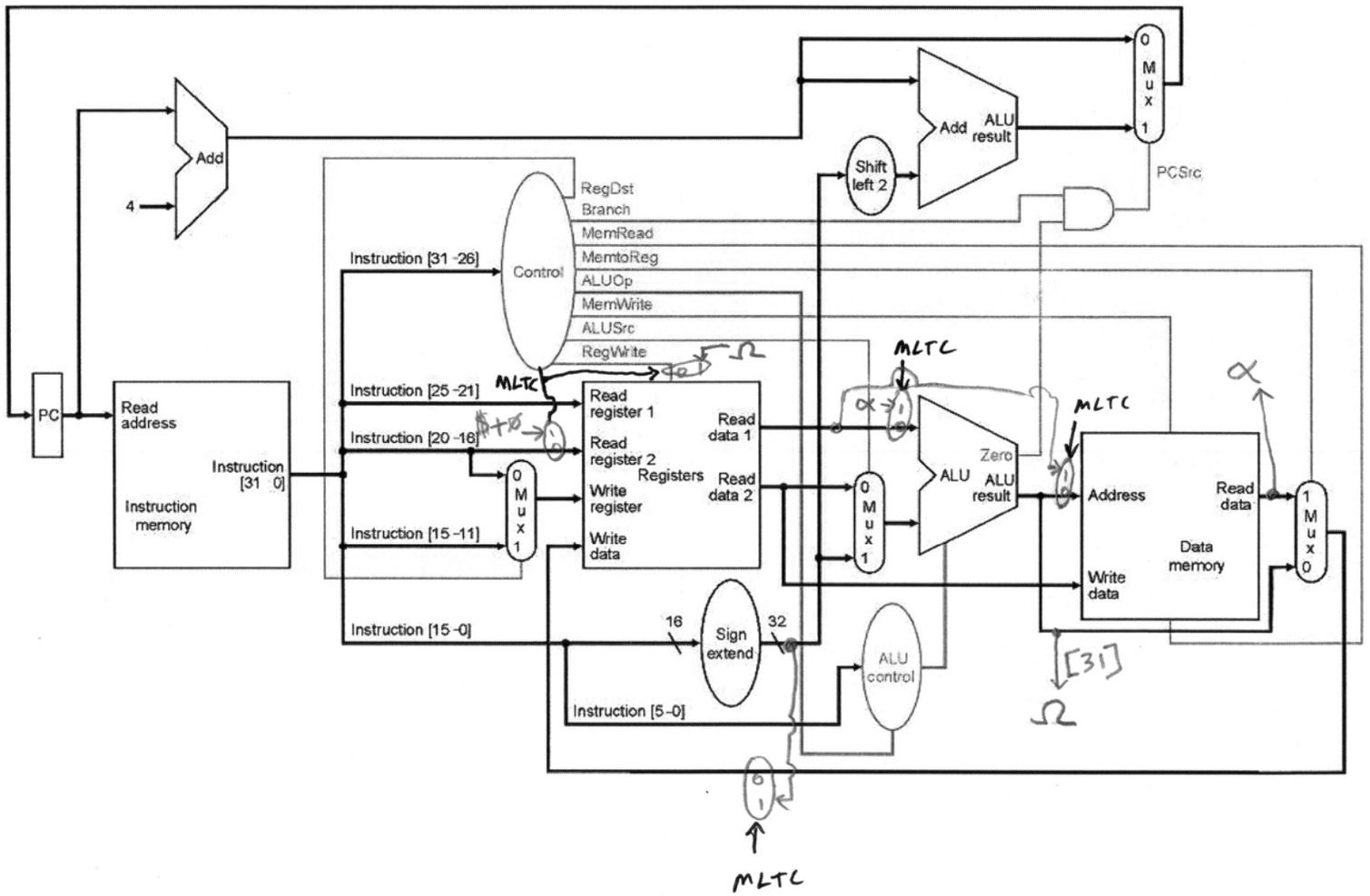
5. ***MLT (30 points)***: Consider the single-cycle processor implementation from class. Your task will be to augment this datapath and control with a new instruction: the *mlt* instruction. This instruction will be an I-type instruction, and will have the following effect:

```
if (M[R[rs]] < R[$t0])
    R[rt] = SE(I);
```

Register $t0 is implicitly used by this instruction – it does not need to be encoded in the I-type fields, it is always used in the < comparison above. The register specified by the rt field is only written if the memory contents at the address specified by the register contents of the rs field register is less than the register contents of register $t0.

Implement your solution on the following two pages. All other instructions must still work correctly after your modifications. You should not add any new ALUs, register file ports, or ports to memory.

5 Muxes → 5 pth each



Main Controller

Input or Output	Signal Name	R-format	lw	sw	Beq	MLT
Inputs	Op5	0	1	1	0	UNIQUE
	Op4	0	0	0	0	
	Op3	0	0	1	0	
	Op2	0	0	0	1	
	Op1	0	1	1	0	
	Op0	0	1	1	0	
Outputs	RegDst	1	0	X	X	Ø
	ALUSrc	0	1	1	0	Ø
	MemtoReg	0	1	X	X	X
	RegWrite	1	1	0	0	X
	MemRead	0	1	0	0	1
	MemWrite	0	0	1	0	Ø
	Branch	0	0	0	1	Ø
	ALUOp1	1	0	0	0	Ø
	ALUOp0	0	0	0	1	1
	MLTC	0	0	0	0	1

5 pts

ALU Controller

Opcode	**ALUOp**	**instruction**	**function**	**ALU Action**	**ALUCtrl**
Lw	00	load word	XXXXXX	add	010
Sw	00	store word	XXXXXX	add	010
Beq	01	branch equal	XXXXXX	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	AND	000
R-type	10	OR	100101	OR	001
R-type	10	SLT	101010	SLT	111

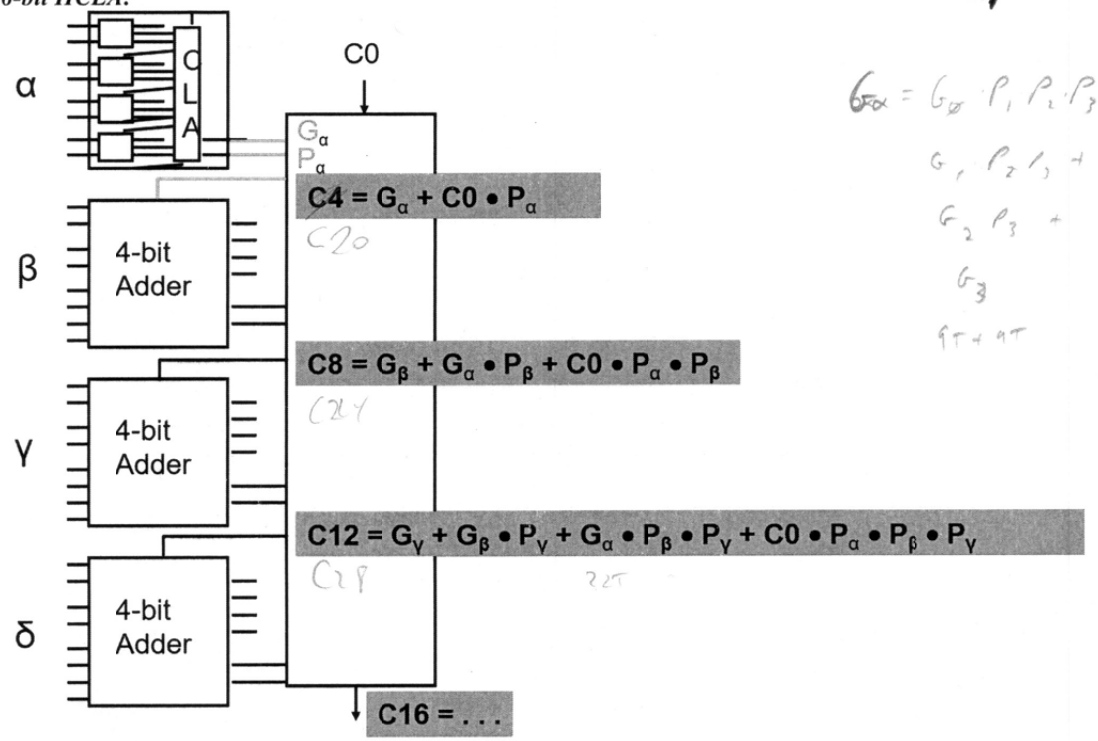
misleading! could be same as C4 with right assumption

Output	Delay (in terms of T)	
G8	4T	*(1 points)*
P8	4T	*(1 points)*
G12	4T	*(1 points)*
P12	4T	*(1 points)*
C8	56T	*(2 points)* 26T + C4
C16	68T	*(2 points)* 12T + C8
S15	~~64T~~	*(2 points)* 12T + (4T or 6T) + 18T + C4

OR Could be C_8 and C_{16} and S_{15} from

52T(before mux/68Taftermux)

16-bit HCLA:



$G_{\delta\alpha} = G_\delta P_1 P_2 P_3 +$
$G_1 P_2 P_3 +$
$G_2 P_3 +$
G_3
9T + 9T

Wrong Calculation

Output	Delay (in terms of T)	
G_δ	22T	*(1 points)*
P_δ	13T	*(1 points)*
C28	~~47T~~ 37T	*(2 points)*
C31	~~65T~~ 55T	*(2 points)*
C32	80T	*(2 points)*
S31	~~81T~~ 80T	*(2 points)*

60T = 12T + 26T + G_δ
12T + (4T OR 6T) + C31
= 81T

18T + G_δ
9T + P_δ
6T + 9T + G_δ
9T + 9T + C_{28}
C16 + 12T