Digital Systems Design and Laboratory
Spring 2019
Lab 2

Tzu-Hsu Yu
e841018@gmail.com
2019/05/20

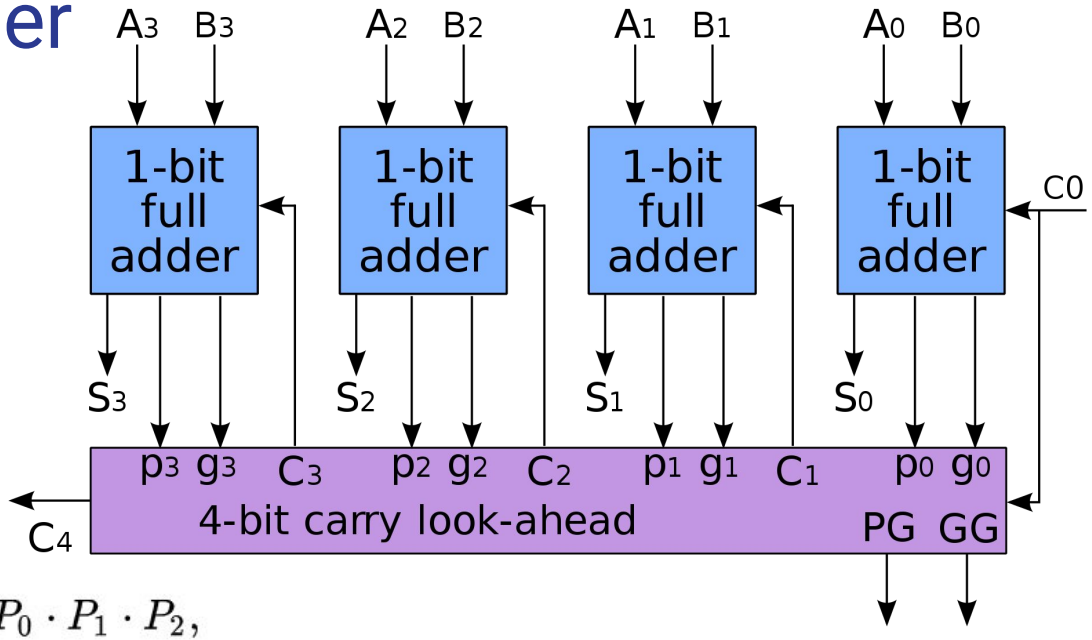
Sample code:
https://drive.google.com/open?id=1x5j8CSZk8CGCsSLyX54GMCv041W_bHFm

Agenda

- More on Lab1
- Multiplier
- Pipeline
- Fast multiplier
- Assignment
- Appendix

More on Lab1

Carry-lookahead adder



$$G_i = A_i \cdot B_i.$$

Pi can be either:

$$P_i = A_i \oplus B_i,$$

$$P_i = A_i + B_i.$$

$$C_1 = G_0 + P_0 \cdot C_0,$$

$$C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1,$$

$$C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2,$$

$$C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3.$$

PG and GG don't have to be implemented in the assignment.

https://en.wikipedia.org/wiki/Carry-lookahead_adder

Source code

Some students did optimizations!

5 versions included in `adders.v` in the sample code:

- Cascaded: 23 ticks

$$C_1 = G_0 + P_0 \cdot C_0,$$
$$C_2 = G_1 + P_1 \cdot C_1,$$
$$C_3 = G_2 + P_2 \cdot C_2,$$
$$C_4 = G_3 + P_3 \cdot C_3.$$

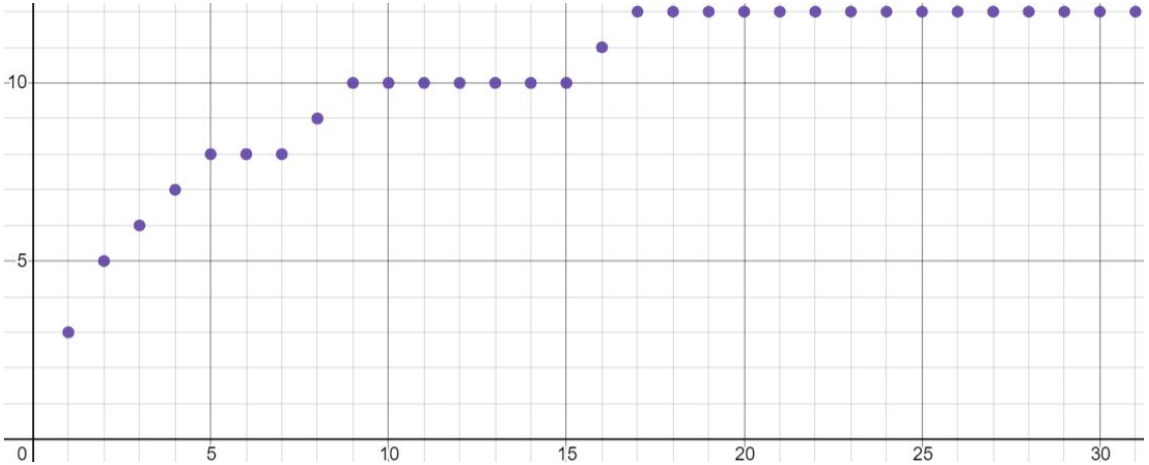
$$C_1 = G_0 + P_0 \cdot C_0,$$
$$C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1,$$
$$C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2,$$
$$C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3.$$

- Implement P_i with XOR: 22 ticks $P_i = A_i \oplus B_i,$
- Implement P_i with OR: 20 ticks $P_i = A_i + B_i.$
- Optimize with AND3 and OR3: 17 ticks
- Optimize with NAND and NOR: 13 ticks

Some derivation

Integer-valued sublinear function:

https://www.desmos.com/calculator/qt9ruazutq



Will be used as the delays of adders in Lab 2

Multiplier

Definitions

Unsigned multiplier: Multiplicand × Multiplier = Product
(signed multipliers will not be covered in Lab 2)

$$
\begin{array}{rrrrrrr}
 & & & & A_3 & A_2 & A_1 & A_0 \\
 & & & \times & B_3 & B_2 & B_1 & B_0 \\
\hline
 & & & & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 \\
 & & & A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 & \\
 & & A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 & & \\
+ & A_3B_3 & A_2B_3 & A_1B_3 & A_0B_3 & & & \\
\hline
\end{array}
$$

AB_i called a "partial product"

Multiplying N-bit number by M-bit number gives (N+M)-bit result

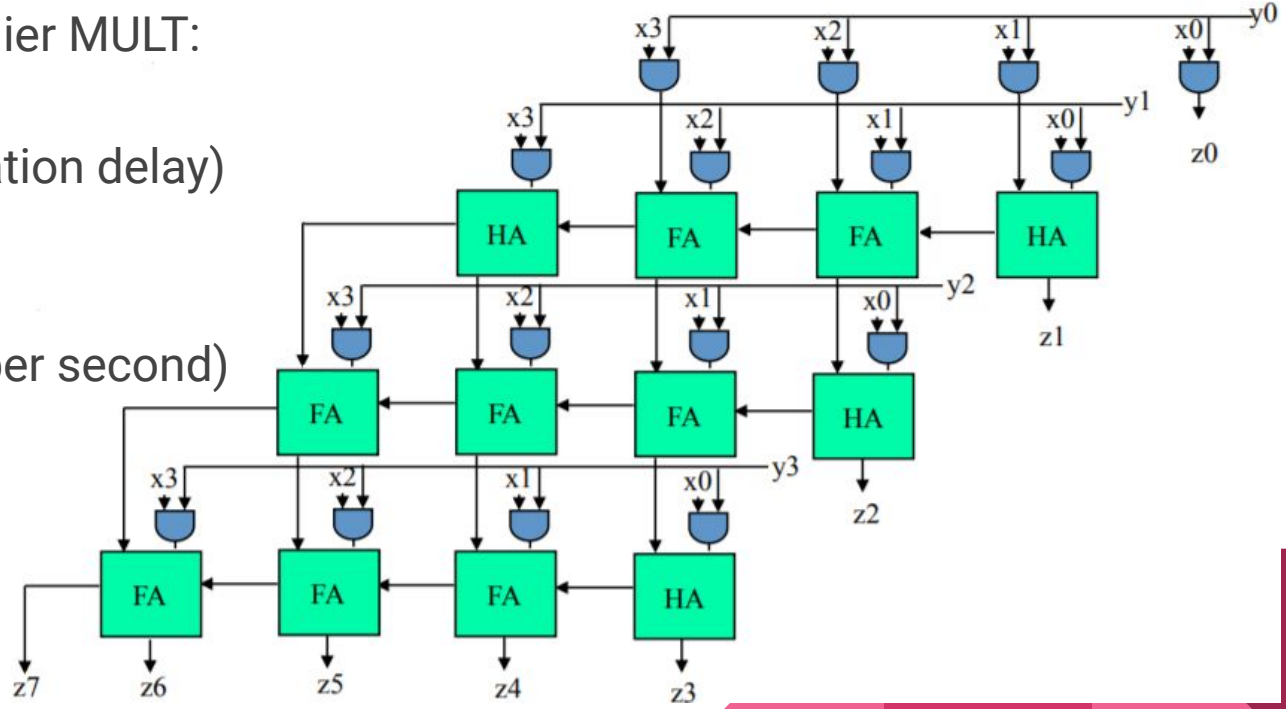
http://courses.csail.mit.edu/6.111/f2008/handouts/L09.pdf

Metrics: latency and throughput

For an n-bit × n-bit multiplier MULT:

Latency (longest propagation delay)
≈ 2n*delay(FA)

Throughput (operations per second)
≈ 1/Latency



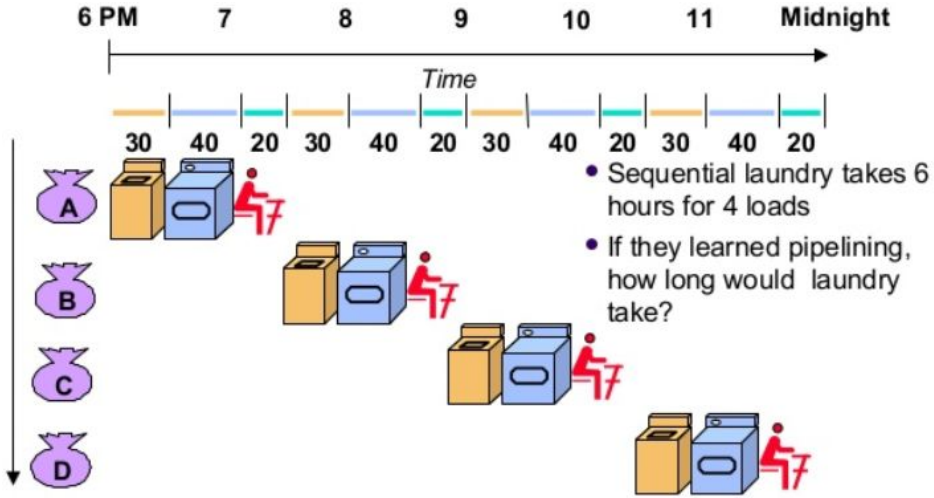
http://courses.csail.mit.edu/6.111/f2008/handouts/L09.pdf

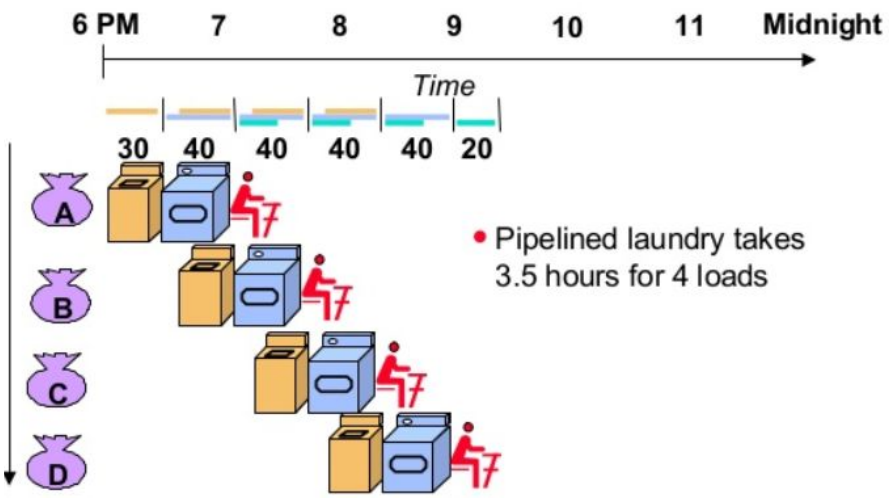
Pipeline

How to increase throughput?

- Cut the circuit into different stages
- Do different tasks in each stage at the same time

Does the latency change? How about the throughput?



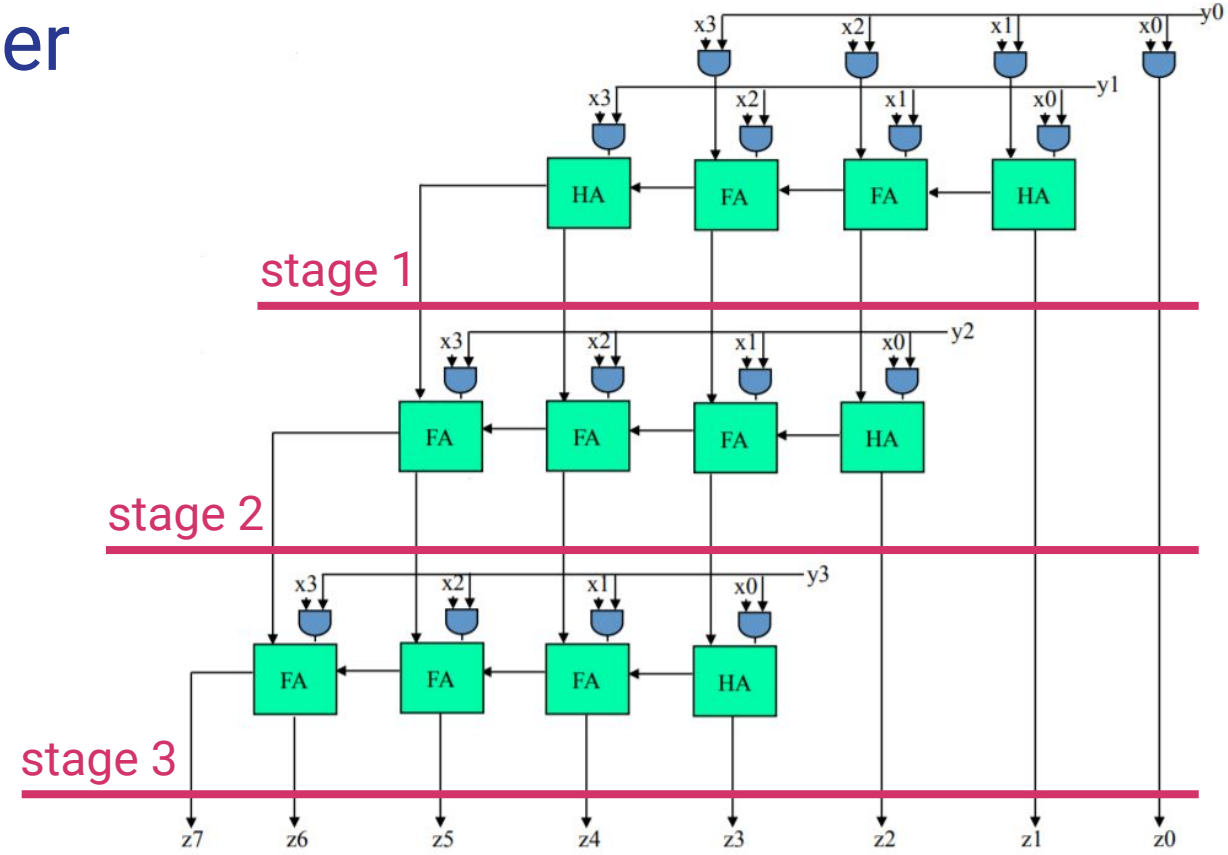


https://www.slideshare.net/siddiqueibrahim37/pipelining-41608675

Pipelined multiplier

Add registers at the end of each stage to keep states

x[3:0] and y[3:0] should also be kept in each stage (not shown in the figure)



http://courses.csail.mit.edu/6.111/f2008/handouts/L09.pdf

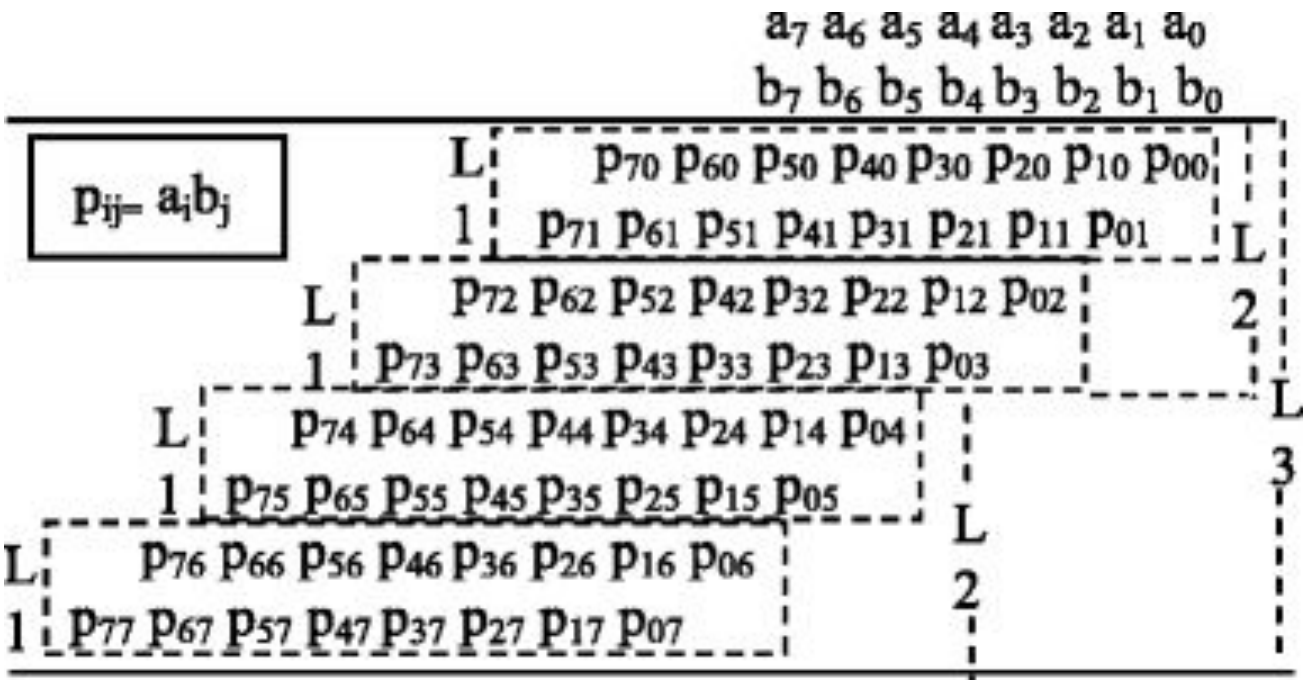
Fast Multiplier

Popular techniques

- Carry-save adder
- Wallace tree
- High-radix
- and many...

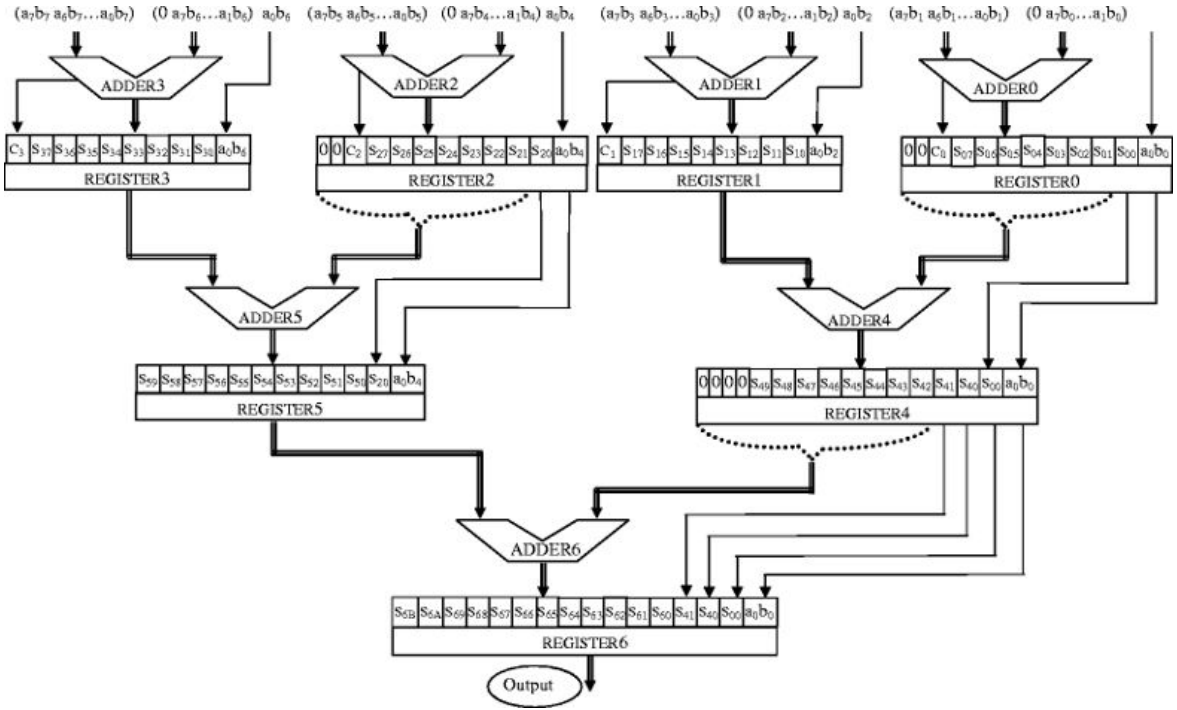
https://www.slideshare.net/siddiqueibrahim37/pipelining-41608675

A hierarchical design



https://link.springer.com/article/10.1007/s11265-012-0657-7

A hierarchical design (cont.)



https://link.springer.com/article/10.1007/s11265-012-0657-7

Assignment

Execute

- Download sample code (link on first page)
- Compile:

  `> iverilog -o lab2.vpp lab2.v`
- Simulate:

  `> vvp lab2.vvp`

  (generates `lab2.vcd`)
- View waveform:

  `> gtkwave lab2.vcd` **`lab2.sav`**

Requirements

- **Replace <index>** in `lab2.v` with proper indexes,
  but **leave the rest of the code unchanged**.
  (1) Show your source code
  (2) Show the waveform with the settings in `lab2.sav`.
- Minimize the clock cycle by **changing the delays** in the module mult_tb.
  (1) What is the minimum clock cycle?
  (2) Show the waveform with the settings in `lab2.sav`.
- Assume the clock cycle is 10 microseconds.
  (1) Calculate throughput.
  (2) Calculate latency.

Appendix

Some details in Verilog:

- Expression bit width
  http://yangchangwoo.com/podongii_X2/html/technote/TOOL/MANUAL/21i_doc/data/fndtn/ver/ver4_4.htm
- Event queue (determines the order of different types of assignment)
  http://www.ece.lsu.edu/v/2015/lsli-event-q.pdf
- Transport delay and inertial delay
  http://www-inst.eecs.berkeley.edu/~cs152/fa06/handouts/CummingsHDLCON1999_BehavioralDelays_Rev1_1.pdf