# IC Design HW#4

B07901165 曾煒哲

Simulation:

Minimum cycle time: 6.0ns

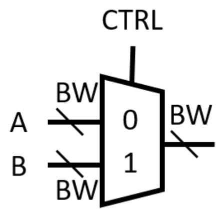
Design strategy: pipeline

Circuit diagram:

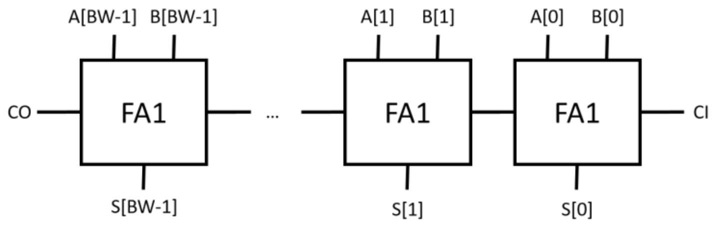
Gate level circuit diagram:

Building blocks:

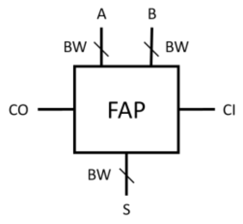
**MUXP(BW)**: basically BW MUX21s sharing the same CTRL.



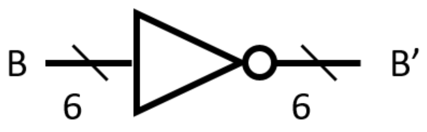
**FAP(BW)**: BW-bit full-adder

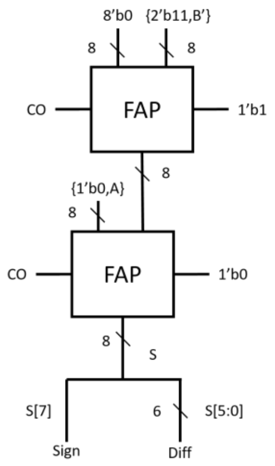


, which is denoted as:

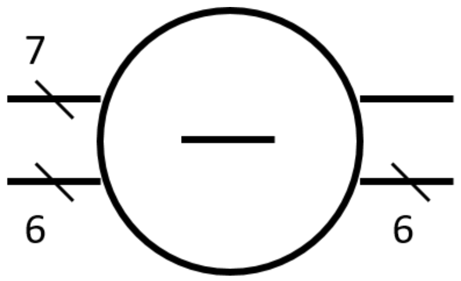


**SUB76**: subtract a 6-bit unsigned number B from a 7-bit unsigned number, then produce a sign bit and a 6-bit unsigned difference.

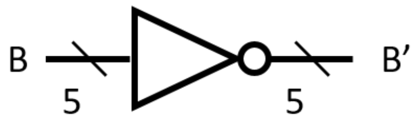


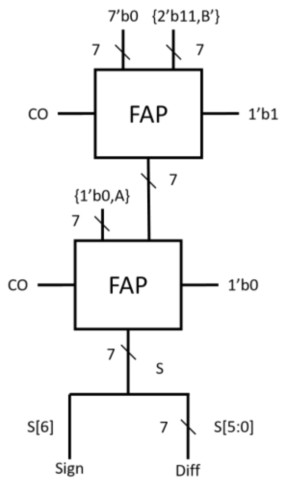


notation:

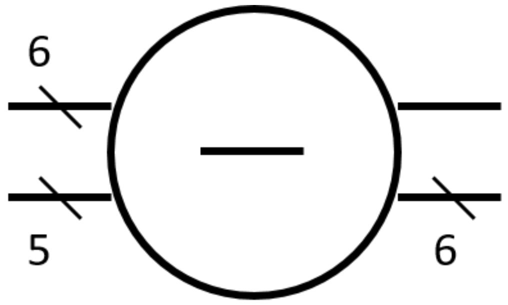


**SUB65**: subtract a 5-bit unsigned number B from a 6-bit unsigned number, then produce a sign bit and a 6-bit unsigned difference.

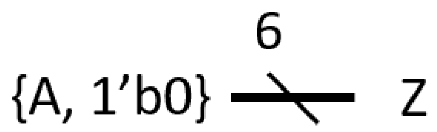




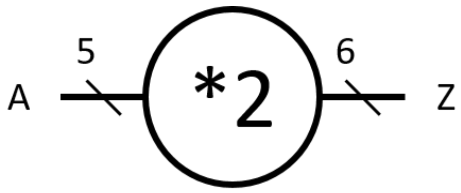
notation:



**Shifter**: multiply a 5-bit number by 2, then produce a 6-bit product.



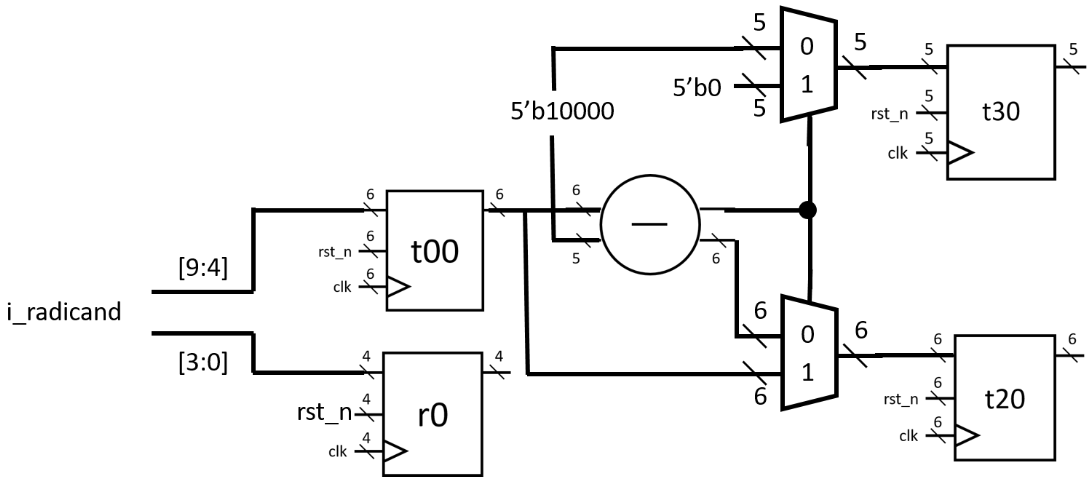
Notation:



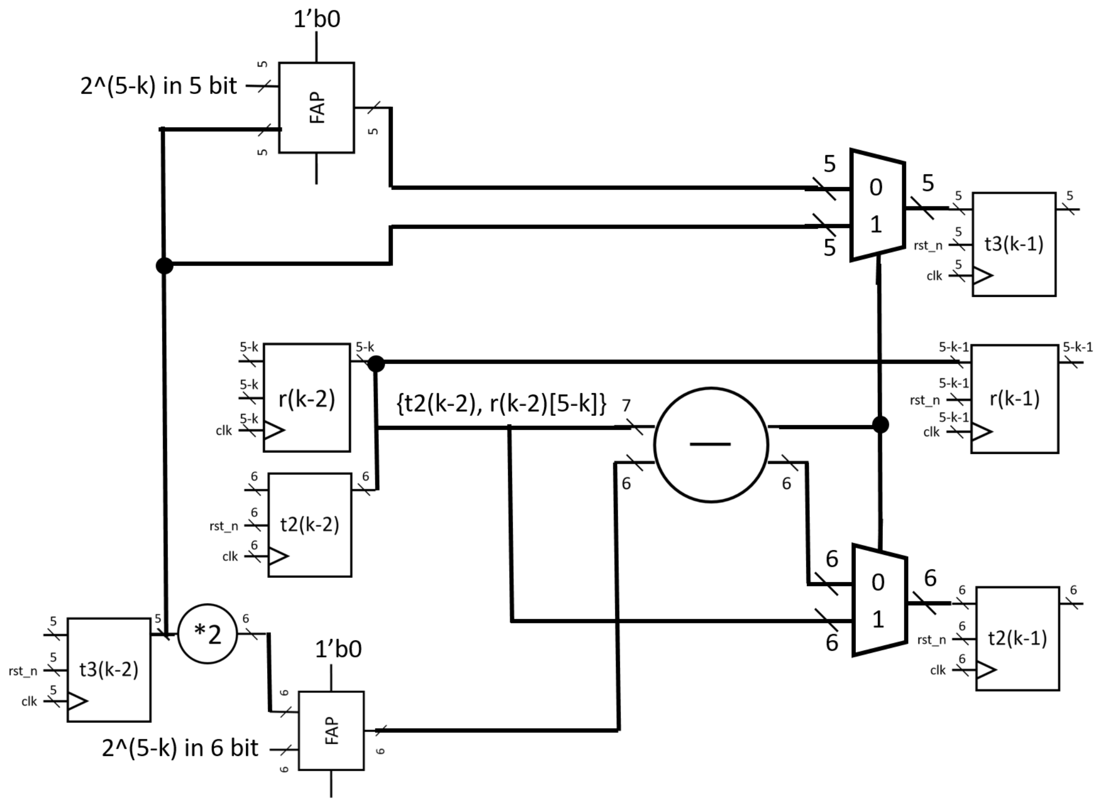
Total circuit diagram:

In the pipeline structure, we have 5 similar stages: k = 1,2,…,5

First stage (k=1) :



k = 2,...,5:



Note that in the 5th stage, we don't construct r(5-1). T3(5-1)_Q is assigned to the final square root.

There is also 6 QD-connected FD2 between 1'b1 and o_finish, pulling o_finish up 6 cycles after the inputs are fed.

Critical path: the one passing through the subtractor in each stage.

Discussion:

在這次的作業中，我使用了 pipeline strategy，而數學上則是使用了長除法開根號的思維: (a+b)^2 = a^2 + b(b+2a)；在數位電路實作上必須將我們熟悉的十進位改成二進位制，所以這個演算法的核心就是: t3(square root)增加 2^(5-k), radicant 就會增加 2^(5-k) * (2^(5-k)+2*t3)；我們就是如此一位一位地往下比，若剩餘的 radicant 還足夠減去 2^(5-k) * (2^(5-k)+2*t3)，則 square root 就增加 2^(5-k)，這樣一來，做完第五個 stage 的 t3 就是我們要的 square root 啦。

而為了減少 transistor 數量，這次我參考了相關的論文，發現在數學上可以證明每個 stage 只會用到 radicant 的 6bit，而且之前用過的就不會再用到，不用浪費記憶體保存，因此在 t2 與 r 的設計上就省去非常多不必要的 flip flop.

另外，原本有考慮像第三次作業一樣做比較器，但後來發現減法器是決定 critical path 最重要的因素，就算比完不用減也還要等其他 stage 減法器，因此改成直接用減法器產生 sign bit，藉此判斷大小，也縮短了 critical path.

Reference:

A New Algorithm for Designing Square Root Calculators based on FPGA with Pipeline Technology (Xiumin Waug, Yang Zhang, Qiang Ye, Shihua Yang)