ASSIGNMENT 7

32-bit Booth's Algorithm Multiplier (Part 2: Control Circuit/Finalizing the design):

For this lab, you will be designing the controller circuit for the Booth's algorithm multiplier. Then you will connect the datapath and controller together (which will complete the Booth's algorithm multiplier design).

Diagram of the inputs and outputs of a Booth's algorithm multiplier:

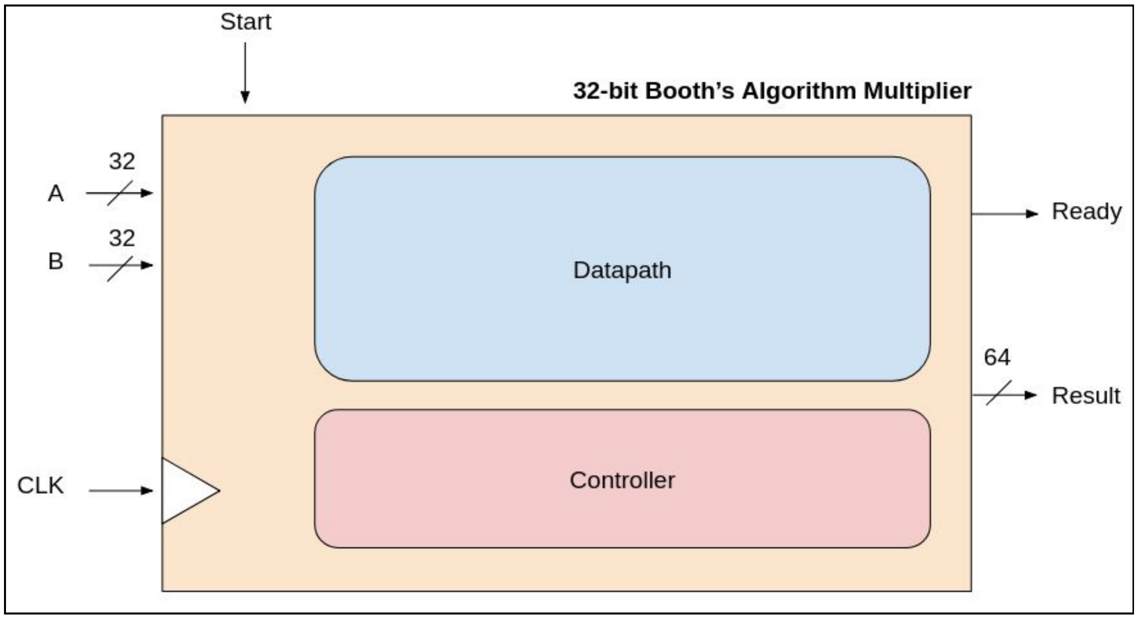
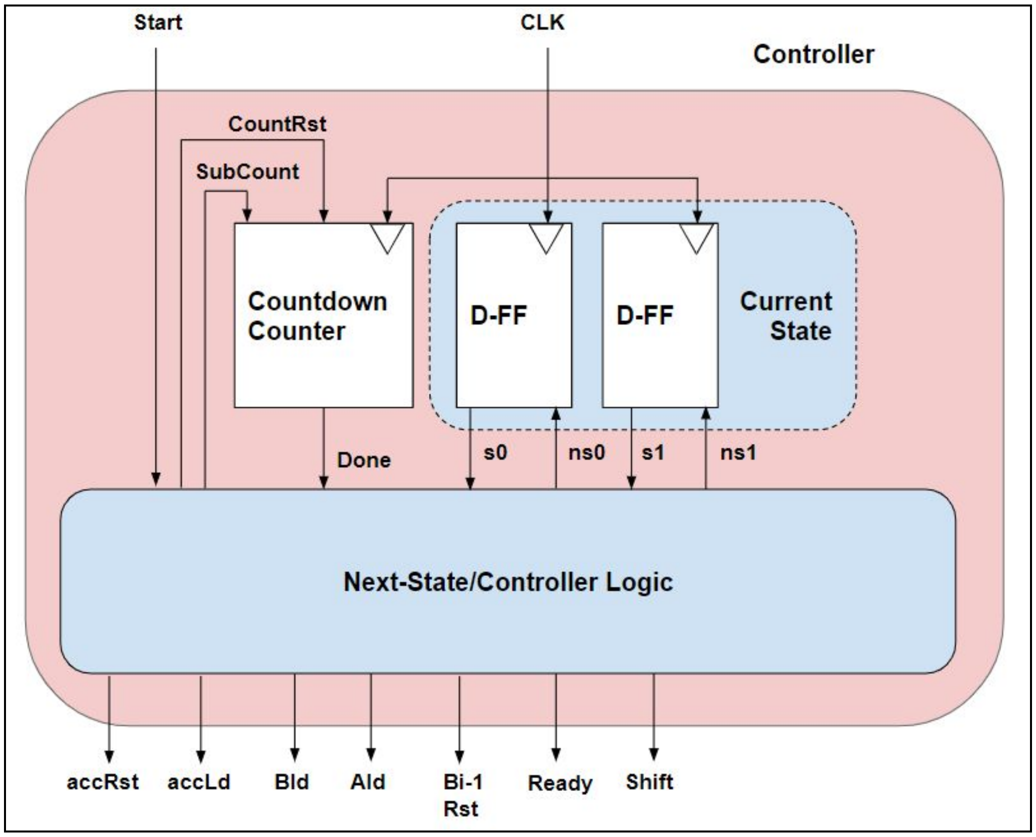


Diagram of Controller portion of Booth's algorithm multiplier:



State-Transition Table for Next-State Logic Equations:

Present s1	State s0	Start	Done	Next ns1	State ns0	accRst	accLd	Bld	Ald	Bi-1 Rst	Ready	Shift	CountRst	SubCount
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0	1	0
0	0	1	0	0	1	1	0	1	1	1	0	0	0	0
0	0	1	1	0	1	1	0	1	1	1	0	0	1	0
0	1	0	0	1	0	0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	1	0	1	0
0	1	1	0	1	0	0	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	1	0
1	0	0	0	0	1	0	0	0	0	0	0	1	0	1
1	0	0	1	0	0	0	0	0	0	0	1	0	1	0
1	0	1	0	0	1	0	0	0	0	0	0	1	0	1
1	0	1	1	0	0	0	0	0	0	0	1	0	1	0
1	1	0	0	x	x	x	x	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x	x	x	x	x

x = Don't Care

The given (already filled-out) state transition table is given above. With that information, solve for the 11 boolean expressions using either K-Map, Quine-McCluskey (tabular) method, etcetera.

Boolean equations to derive:

ns1 =

ns0 =

accRst =

accLd =

Bld =

Ald =

Bi-1 Rst =

Ready =

Shift =

CountRst =

SubCount =

The Controller module will have the following 3 states: **Idle (00)**, **Load (01)**, **Shift(10)**

NOTE: *Bit 1* represents the **s1** variable and *bit 0* represents the **s0** variable for the above state explanation.

Hints For Countdown Counter Module:

- For the design of the "Countdown Counter", make a 6-bit internal register and initialize it to equal 32 in decimal (i.e. 100000 in binary), **SubCount** will be the 1-bit "flag/signal" to tell the counter to *decrement* on the next rising edge of the clock signal, **CountRst** will be the 1-bit "flag/signal" to *reset* the internal register back to 100000, if the internal register at any point equals zero, set the **Done** output "flag/signal" to 1 (if the internal register is any other value other than zero, set this output to 0)
- You can initialize registers that are inside modules with the **initial begin … end** block, set your internal register to whatever value you need inside this initial block (very similar to using the initial block in a testbench)
- Why this counter sets a **Done** "flag/signal" when the counter counts-down from 32 (in decimal) to zero, is because this is how many iterations/clock-cycles it will take to perform a 32-bit Booth's Algorithm signed multiply operation

You should have these separate modules in the end:

- D-FF
- Countdown_Counter
- Next-State/Controller Logic

Once all of the above sub-modules are created and once you have testbenched each sub-controller module individually, create 2 new modules:

- Controller
- Datapath

These two modules will "house/contain" your sub-modules you all designed from the past lab as well as this lab (i.e. the datapath submodules created from last lab will be instantiated inside the datapath module and the controller submodules will be instantiated inside the controller module).

Once those two modules are created, **it would be wise** to testbench those two modules before moving on to the final module (in other words the debugging, if needed, would be easier for you), but feel free to skip this stage if you are sufficiently confident in your design.

Last step, make the following module:

- Booths_Signed_Mult32

At this last step, instantiate the following 2 modules: **Datapath** and **Controller.** At this point, you will finalize the remaining few connections that need to be made. If all goes well, you should have a working 32-bit signed multiplier

Also, the 64-bit result output will be the concatenation (aka the combination) of the **Acc. Reg** and the **B reg** where the **Acc. Reg** represents the *upper 32-bits* and the **B Reg.** represents the *lower 32-bits.*

Lab Assignment:
Design a 32-bit Booth's signed multiplier in Verilog. Create testbenches (controller sub-modules as well as final design) to test your design for various input combinations. In your **compressed zip file**, provide Verilog code for each module, test bench codes for the controller sub-modules and the final design, and waveform screenshots for those testbenches.

Due Date: Deliverables—Mar. 19 11:59pm (submit on BlackBoard)
Demo---Mar. 20 (during lab class)