Logical Left Shift

$in[n-1:0]$, $shift\_ammount[clog_2(n)-1:0]$, $out[n-1:0]$

in   shift_ammount   Schem uses n=4 fer my sanity

3 2 1 0

IDEA: Concat in with bus of 0s.
Better fer generate statements.

out[0]

out[i]

out[2]

out[3]

in_ext

6 5 4 3 2 1 0   shift_ammount
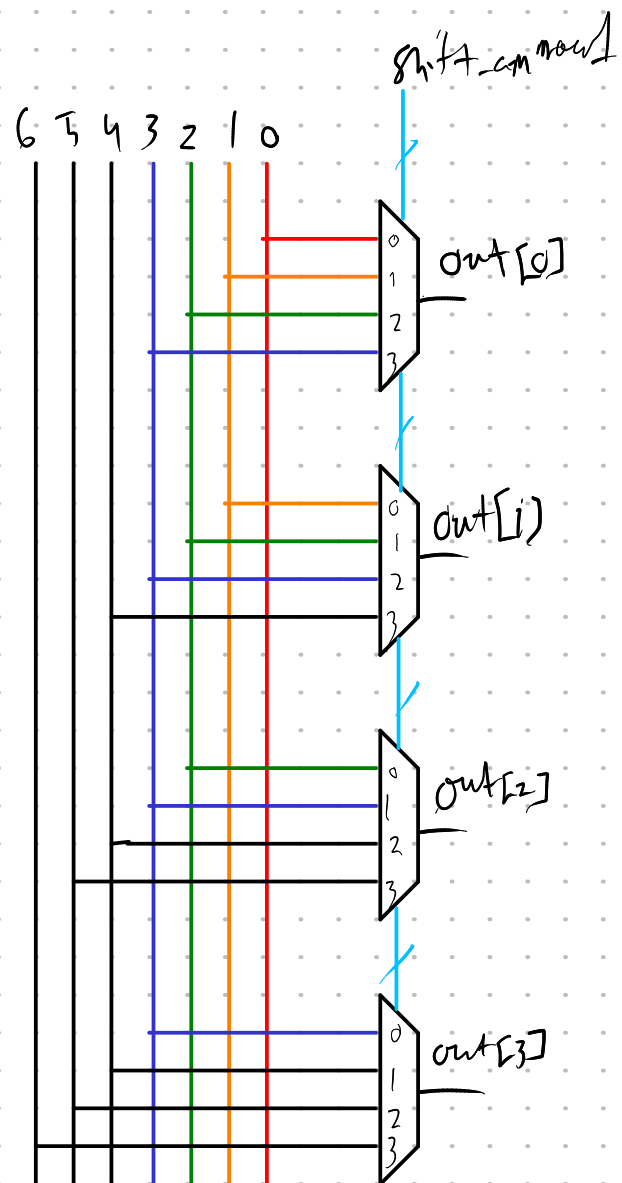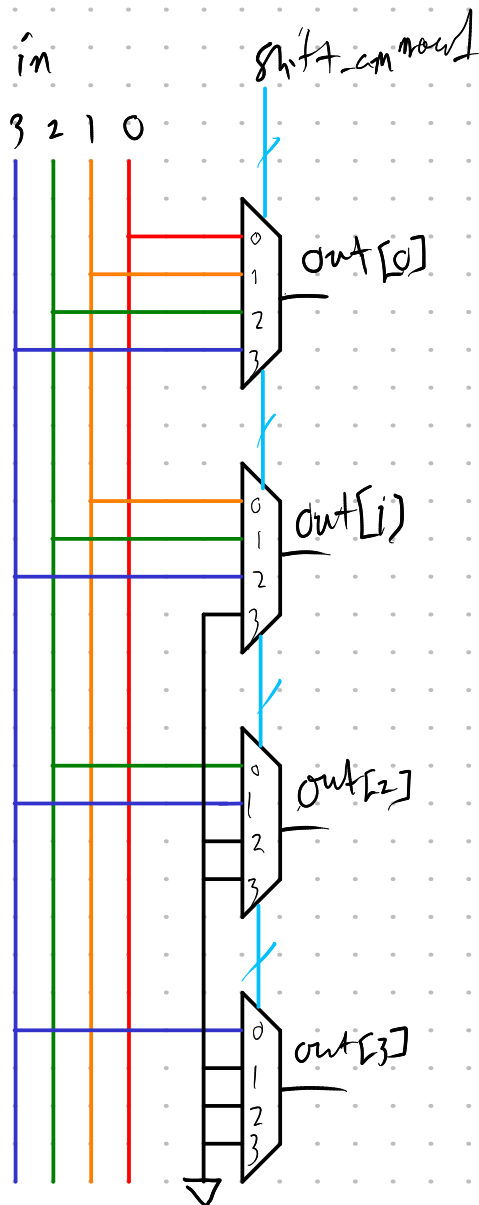
mux0
out[0]

mux1
out[i]

mux2
out[2]

mux3
out[3]
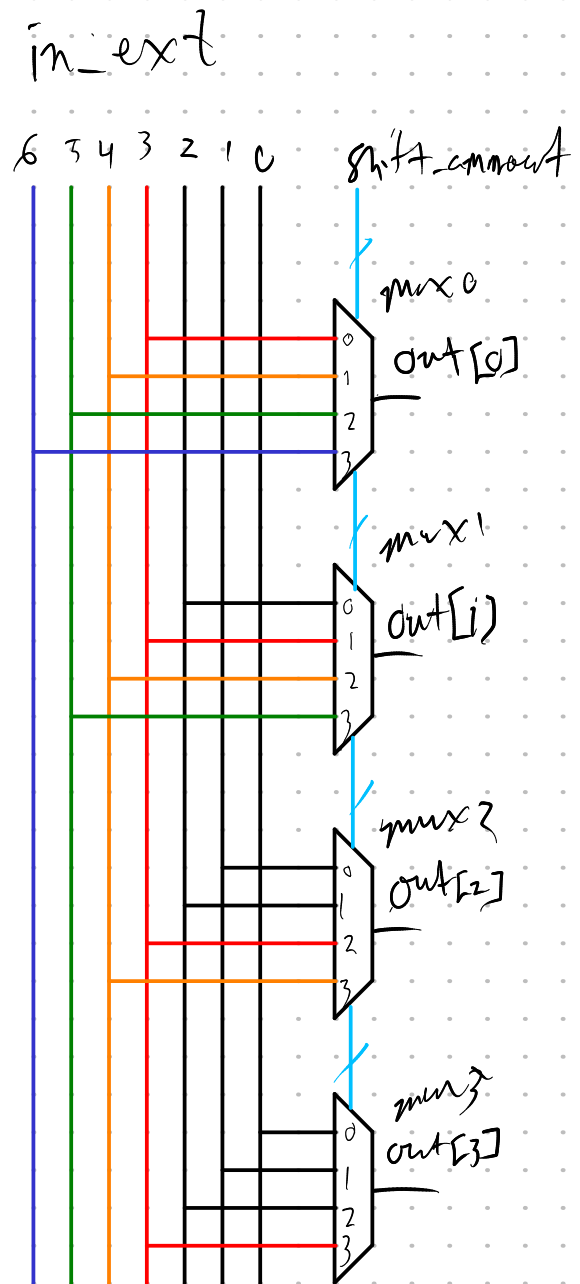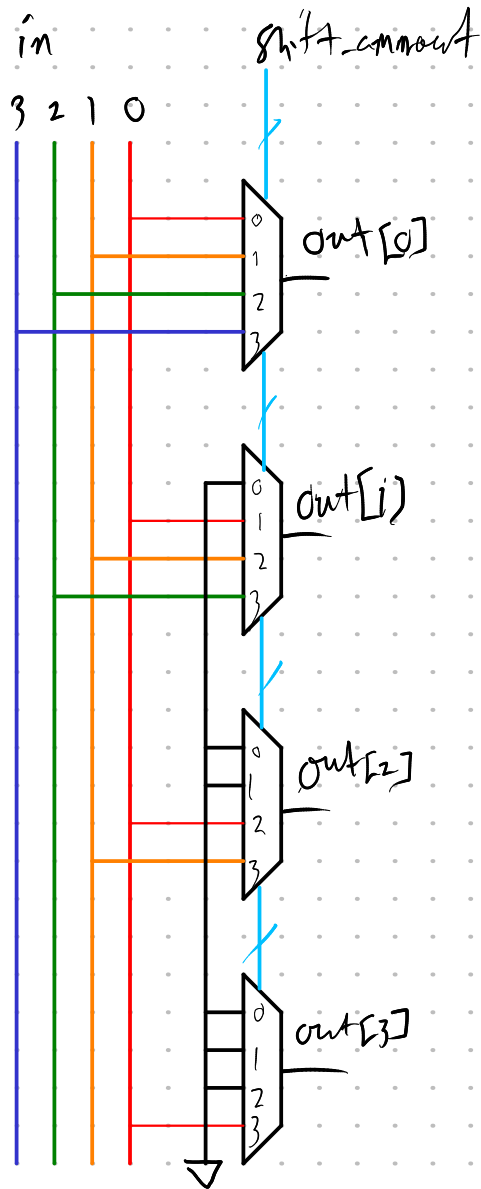
Logical Right Shift

$in[n-1:0]$,   shift-ammount $[log_2(n)-1:0]$, $out[n:0]$

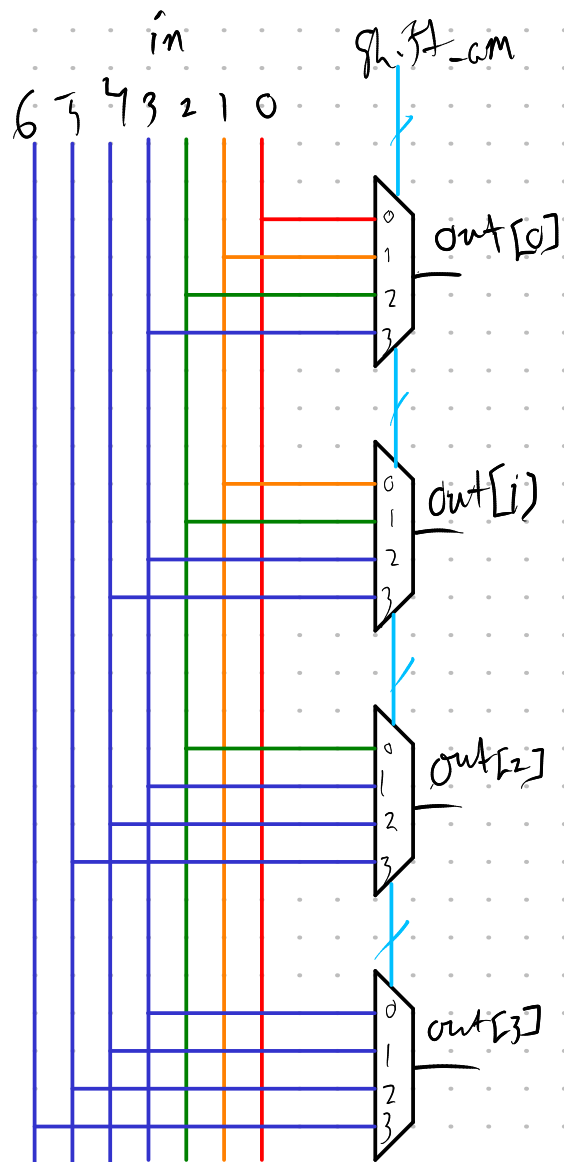Schem using $n=4$ for my sanity

Idea: Concatinate with $0s$
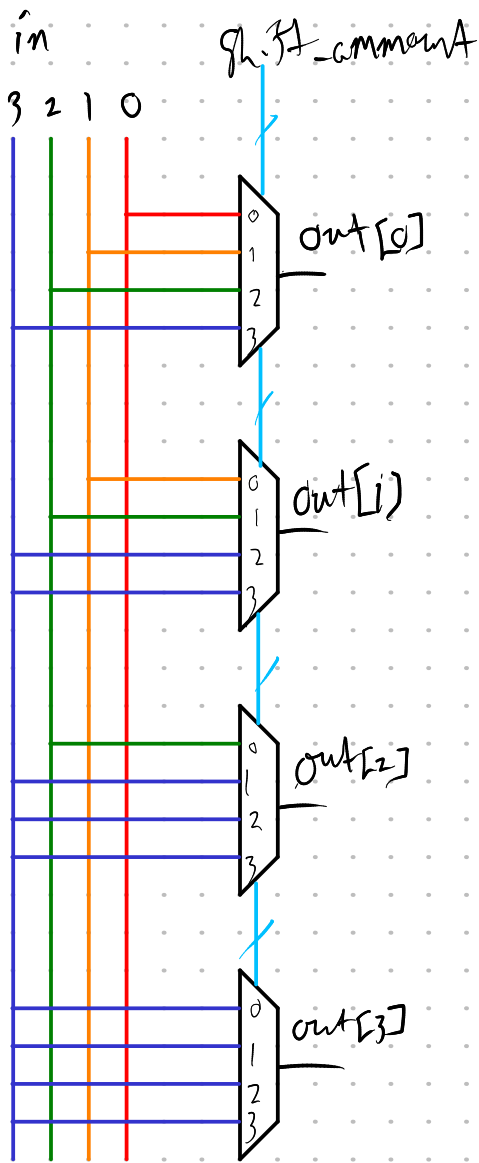
Arithmatic Shift Right

$in[n-1:0]$, shift_amount $[clog_2[n]-1:0]$, out $[n-1:0]$

Same using $n=4$

Idea: extend bus again

ALU

ALU_AND = 4'b0001, ⎫
ALU_OR = 4'b0010, ⎬ Bitwise
ALU_XOR = 4'b0011, ⎭
ALU_SLL = 4'b0101,
ALU_SRL = 4'b0110,
ALU_SRA = 4'b0111,
ALU_ADD = 4'b1000, ⎫
ALU_SUB = 4'b1100, ⎬ less than signed
ALU_SLT = 4'b1101, ⎭
ALU_SLTU = 4'b1111 — less than, unsigned

in: a, b, Control

out: result, overflow, zero, equal

Added sillyness:
if more than $clog_2 N$ bits in b,
overflows (use a mux)