

Time analysis:

I will analyze how much each state (which refers to a clk cycle) takes time to identify the max time needed to be accounted for by the clk.

START: initialize registers so

time needed = time biggest register + time biggest multiplexer

ENTER LOOP:

loading K and getting value of stacknumel[k] so the 2nd operation is critical path

time needed = multiplexer time + time for register + comparator time; here we need comparator time to know to which state to go after this clock cycle from stacknumel[k] value.

GET ix,adrs,stacknumel,w:

getting ix and stacknumel and adrs happen in parallel while w needs to wait for adrs

time to get ix = 2*register+multiplexer

time to get stacknumel = multiplexer+register+subtractor(same as adder)

time to get adrs = 2*register+multiplexer

time to get w = (time to get adrs + adder of adrs) + Memory + register = 11.93 + 12 + 0.35 + 0.25 = 24.53 ns

so clearly time to get w is critical path in this state so time needed = time to get w = 24.53

GET stackix,parent:

time to get stackix = multiplexer + register + adder = 6.166 ns

parent is ignored in code so I didn't put it in the path

and we also increment address here so

time adrs = adder + register + multiplexer = $2 * \log_{\text{base } 2} \text{ of } (18) + 1.5 + 0.35 + 0.25 + 0.9 = 11.33$ ns

so time adrs > time stackix

so time adrs is critical path for this state

GET nc:

we increment adrs and we also load number of children from memory and see the value of comparator on k in parallel

comparator = $0.4 * \log_{\text{base } 2} \text{ of } (4) = 0.8$ (k is 4 bits wide).

increment adrs = 11.33 ns

loading nc = register + adder + Memory = 22.43 ns

so loading nc is the most time in this state.

ROOT W:

we load w into wpath[k]

time = $2 * \text{register} + \text{adder} + \text{multiplexer} = 11.41 \text{ ns}$

GET Weight[k-1]:

load into weight[k-1] register its value so

time = $\text{adder} + \text{multiplexer} + 2 * \text{register} = 11.41 \text{ ns}$

GET Weight[k-1]+W:

addition was done in previous state and we need the comparators of nc which also are done in parallel

comparator time = $0.4 * \log_{\text{base } 2} \text{ of } (3) = 0.63 \text{ ns}$

time = $\text{register} + \text{multiplexer} = 1.5 \text{ ns}$

so time = 1.5 ns of this state.

LEAF NODE:

loading pathmin and wpathmin

time wpathmin = $\text{multiplexer} + 2 * \text{register}$

time pathmin = $\text{register} + \text{multiplexer}$

so time = 2.1 ns

1Child/2Children/2Children2/3Children/3Children2/3Children3/4Children/4Children2/4Children3/4Children4:

increment adrs is followed by a memory read and then inserted into stack adrs register so this is definitely critical path in this state

time = $\text{adder of adrs} + \text{Memory} + \text{multiplexer} + \text{register} = 9.83 + 12 + 0.9 + 0.35 + 0.25 = 23.34 \text{ ns}$

DECREMENTK:

get nextk from k-1 and compare to see if done

time = $\text{multiplexer} + \text{register} + \text{adder} + \text{comparator} = 7.8 \text{ ns}$

DONESTAGE:

decrement register minnodes and store it again and compare

time = $\text{register} + \text{adder} + \text{comparator} = 5.9 \text{ ns}$

So, the critical path delay is getting W in the **GET ix,adrs,stacknumel,w** state which takes 24.53 ns so our clk cycle better be 25 ns or so to account for this delay.

