Doubly Logarithmic Depth Trees

In a doubly-logarithmic depth tree, the number of level-i nodes is equal to

$$\begin{aligned}
2^{2^{n-1}} 2^{2^{n-2}} \ldots 2^{2^{n-i}} &= 2^{2^{n-1}(1+1/2+1/2^2+\ldots+1/2^{i-1})} \\
&= 2^{2^{n-1} \cdot \frac{1/2^i-1}{1/2-1}} \\
&= 2^{2^{n-1} \cdot \frac{1/2^i-1}{-1/2}} \\
&= 2^{2^{n-1}(2-1/2^{i-1})} \\
&= 2^{2^{n-1}(2-2^{-i+1})} \\
&= 2^{2^n-2^{n-i}}
\end{aligned}$$

Each such node of level i has $2^{2^{n-(i+1)}}$ children in level $i+1$. Consider the square of this number which is $2^{2(2^{n-(i+1)})}$ and corresponds to the number of processors assigned to these children in algorithm MAX2. As the total number of level i nodes is $2^{2^n-2^{n-i}}$ the total number of processors assigned to level $i+1$ nodes is then given by the following quantity.

$$\begin{aligned}
2^{2^n-2^{n-i}} \cdot (2^{2^{n-(i+1)}})^2 &= 2^{2^n-2^{n-i}} 2^{2(2^{n-(i+1)})} \\
&= 2^{2^n-2^{n-i}} 2^{2^{n-i}} \\
&= 2^{2^n-2^{n-i}+2^{n-i}} \\
&= 2^{2^n} \\
&= N
\end{aligned}$$

Note: In class I used m for n and n for N.