8.5 The Core Clock System on the Next Generation Itanium™ Microprocessor

Ferd E. Anderson, J. Steve Wells[1], Eugene Z. Berta[2]

Intel Corp., [1]Hewlett Packard Corp.
[2]Hewlett Packard Corp., Fort Collins, CO

The next generation of the Itanium™ processor family (code name McKinley) contains both a processor core and large on-die cache as shown in the die micrograph in Reference [1]. The core clock generator consists of a PLL, a ring oscillator for clock start up, and a bypass clock for debug (Figure 8.5.1). The core frequency is multiplied up from the system clock by a ratio-selectable divider in the synchronizer. This allows a wide range of system clock frequencies coupled with the highest possible core clock frequency. The synchronizer also generates frequency-divided clock qualifiers that are clocked via a latch pipe to the front side bus (FSB). These qualifiers combine with the core clock at the FSB to recreate the system clock frequency that is routed back to the PLL feedback input in order to create phase alignment with the system clock. A second test-mode feedback is selected when using stop-clock, on-die-clock-shrink (ODCS), and locate-critical-path (LCP) debug features so that the PLL is not aware of the clock manipulations and stays in lock [2].

The clock distribution is realized by a precision-engineered wire system, a balanced, multi-level H-tree. An H-tree design has historically been an attractive conceptual solution since it uses much less routing resource, and results in significantly less driver load that saves power. But the difficulty in finding a practical implementation method has deterred its use [3]. Due to the relatively long lengths and low resistance of the clock routes, inductance modeling is a requirement. Simulation results indicate a significant timing (±10% delay) impact due to inductance. A custom solution integrated into the simulation tools fulfills this requirement [4]. This clock system is measured to operate from 1.2GHz to over 2.0GHz for 1.2 to 2.0V power supply levels, respectively, for the 14059x18599μm core.

The clock distribution consists of a level one route (L1R) containing a core primary driver, repeaters, second-level-clock-buffers (SLCBs) and differential routing (Figure 8.5.2). The differential nature of the L1R results in reduced jitter from supply noise, injected common mode noise, and signal slew rates. The heavy shielding also reduces jitter by reducing coupled noise into the system. The level-two route (L2R) consists of single-ended clock distribution, and final local qualified buffers (gaters). The L2R is shielded to minimize jitter, and both levels of route are delay-tuned to minimize skew. The schematics of the primary driver, L2R repeaters, and SLCBs are shown in Figure 8.5.3. The primary driver is preceded by a single-to-differential converter (not shown), and the SLCB contains an additional current-starving delay adjustable input (not shown) for deskew and as a debug aid to help locate critical paths.

The physical geometry chosen for the L1R consists of a differential pair separated by a center noise shield and two outside noise shielding wires (Figure 8.5.4a). Addionally, there is a parallel N-2 layer for shielding noise from lower layers. This configuration provides shielding from inductive effects by locating ground current returns close to the clock signal wires. The L1R delivers the clock to all corners of the die within 52ps skew including PVT (as measured on one part).

Implementing the balanced H-tree route presents unique challenges between the clock routing team and the block designers. The solution is to implement the L1R by defining initial "reserved channels" for the major root sections of the tree based on defined load zones that are roughly the four quadrants of the core floorplan. The clock is first routed to a centrally located primary driver. From there, the route to each quad is engineered with the help of custom tools based on the combination of a GUI to the nominal layout tools and the custom inductance modeling tools. This methodology allows significant freedom for top-level floor plan movement and sufficient abstraction for individual block designers to make clocking at the top level effectively transparent to the rest of the design team.

The structure of the L2R is a width and length balanced side-shielded route in the conceptual form of a binary tree (Figure 8.5.4b). Leaf nodes represent the tap points to which designers connect their gaters, nodes represent connecting points, and edges represent the width of the route. Block designers are given considerable freedom in the placement of L2R tap points. Tap points are strapped where the skew impact is <5ps. The process of generating an L2R route for a local zone uses a mixture of proprietary GUI based tools (EZROUTE) and manual intervention. The capacitance data of the L2R tap points for each of the zones is fed into EZROUTE. Using a grouping algorithm, Elmore delay calculations, and summed capacitance values, the route designer created the link structure for the base tree (Figure 8.5.5).

The grouping algorithm is based on a simple heuristic that takes into account only distance, Elmore delay between points, and the level of the tree. The tool generates a balanced H-tree assuming a constant width of wire based on the level of the tree. Manual intervention is required to optimally link together the higher parts of the tree where width modulation is required due to design constraints. This process is followed by wire width modification (balancing) iterations based on Elmore delay until a mismatch skew of <10ps is achieved using detailed RLC wire models (Figure 8.5.6). The balancing process follows a bottoms-up approach by first reducing the within-zone skew, followed by balancing out zone-to-zone skew, typically with only a few quick Elmore iterations.

The resulting L2 routes have 10ps maximum mismatch and 25ps maximum one-sigma skew, based on Monte-Carlo PT variations. The total power usage for the core L2R routes and SLCBs is <800mW.

Acknowledgements:
The authors thank Ian Young, Simon Tam, Sam Naffziger, and Wayne Kever for valuable assistance.

References:
[1] S. Nafzigger, G. Hammond, "The Implementation of the Next-Generation 64b Itanium Microprocessor," ISSCC Digest of Technical Papers, Paper 20.3, Feb. 2002.
[2] S. Tam, et al., "Clock Generation and Distribution for the First IA-64 Microprocessor," IEEE J. of Solid-State Circuits, vol. 35, no. 11, pp. 1545-1552, Nov. 2000.
[3] Phillip J. Restle, et al, "A Clock Distribution Network for Microprocessors," IEEE J. of Solid-State Circuits, vol. 36, no 5, pp. 792-799, May 2001.
[4] A. E. Ruehli, "Inductance Calculation in a Complex Integrated Circuit Environment," IBM Journal of Res. & Dev., 1972.

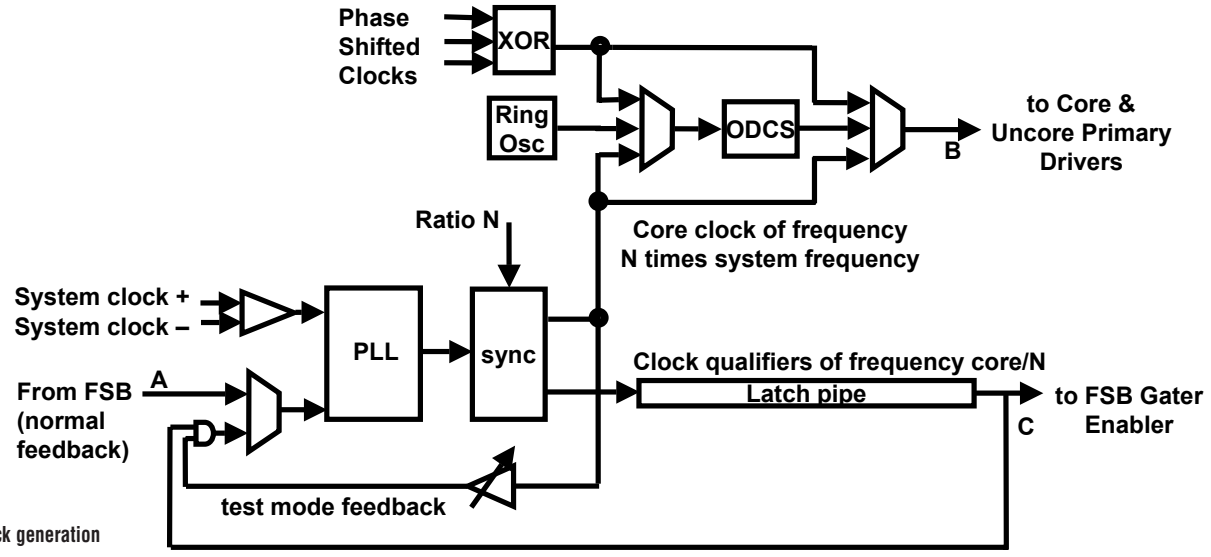


Figure 8.5.1: Clock generation diagram.

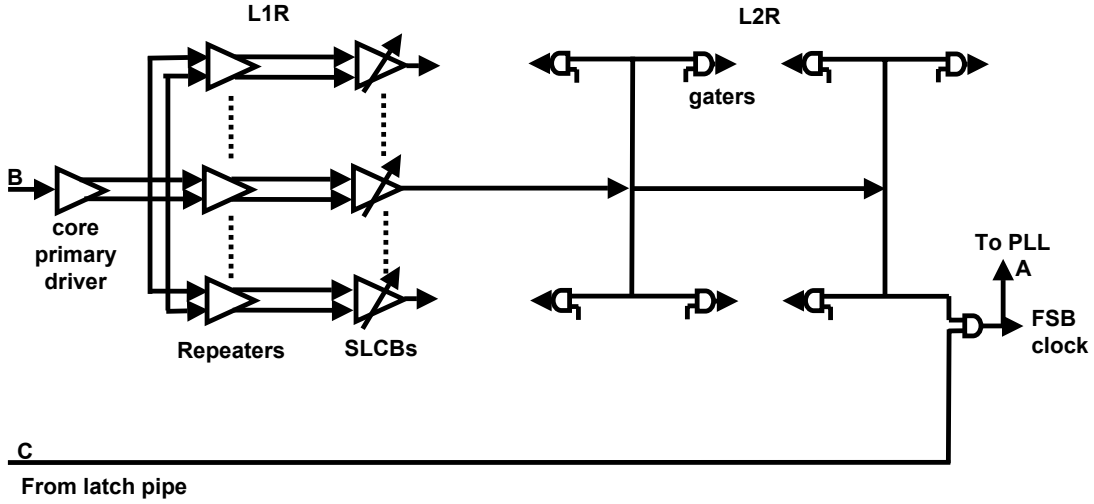


Figure 8.5.2: Core clock level one and level two distribution.

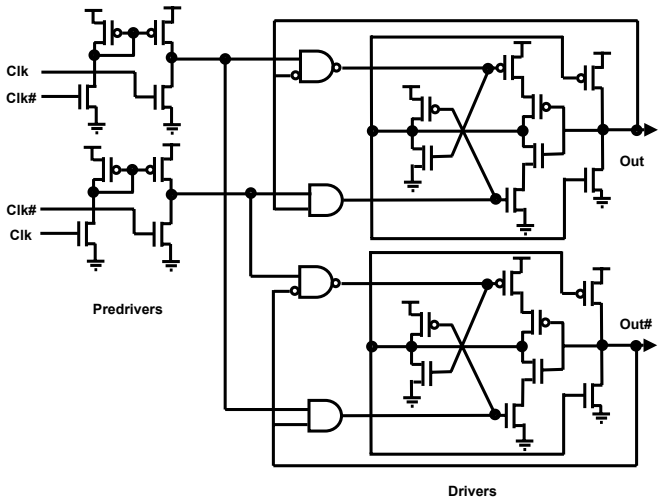


Figure 8.5.3: Primary driver/repeater/SLCB schematic.

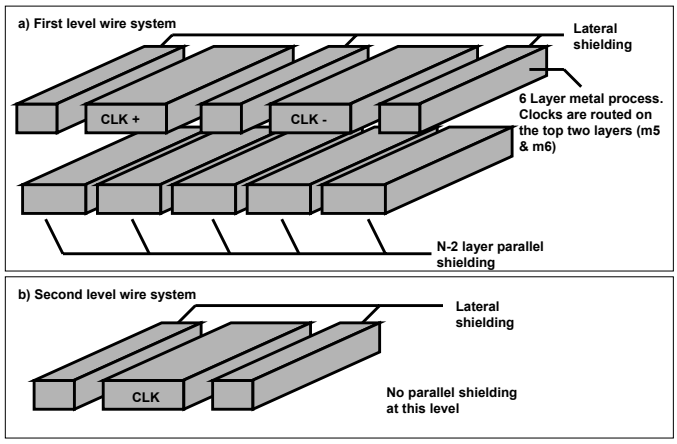
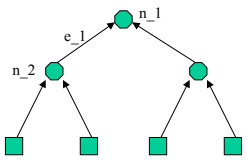


Figure 8.5.4: First and second level route geometry.



Leaf Node: Location and routes to local gaters determined by block designer.
Cap load of gaters extracted.
ActD to represent simulated delay.
AvgD to represent average delay.

Branch Node: Location of branch in L2R determined by tool.
ActD to represent simulated delay.
AvgD to represent average delay.

Link: Width and length of route between nodes determined by tool.

De-skew Algorithm applied per iteration:

AvgD(n) = [AvgD(left child of n) + AvgD(right child of n)]/2
ActD(n) = simulated delay to n from root of tree
Wid(e) = width of an edge
Del(edge) = delay between nodes connected by edge
Del_to_Wid(edge, delay) = estimated width to cause delay between edge
CONST = convergence factor

Wid(e_1) = Del_to_Wid(e_1, Del(e_1) + CONST * ActD(n_2) * (1 - AvgD(n_1)/AvgD(n_2)))

Figure 8.5.5: Data representation of level two route.

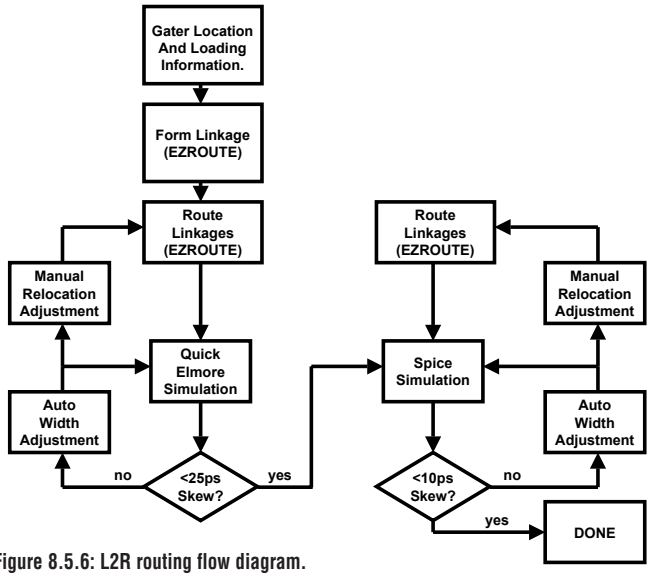


Figure 8.5.6: L2R routing flow diagram.

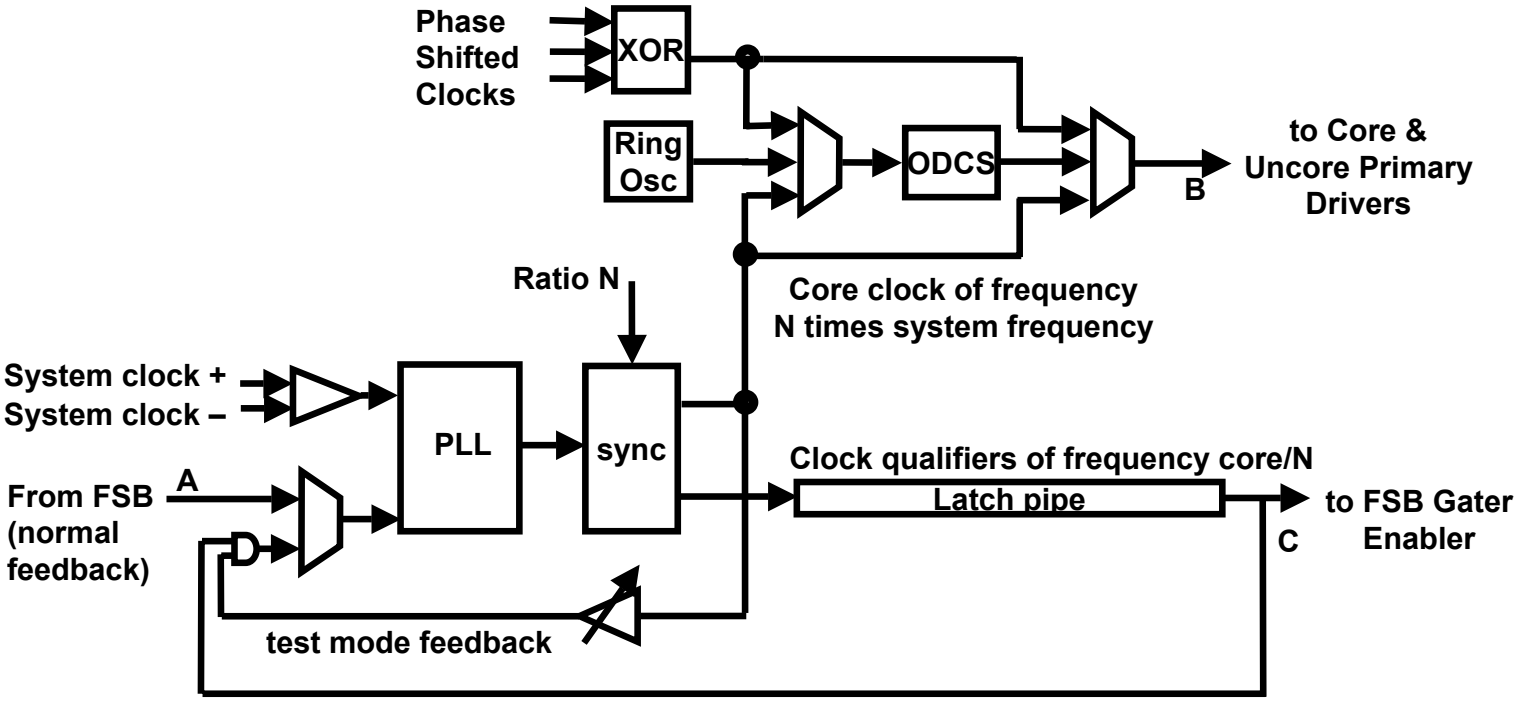


Figure 8.5.1: Clock generation diagram.

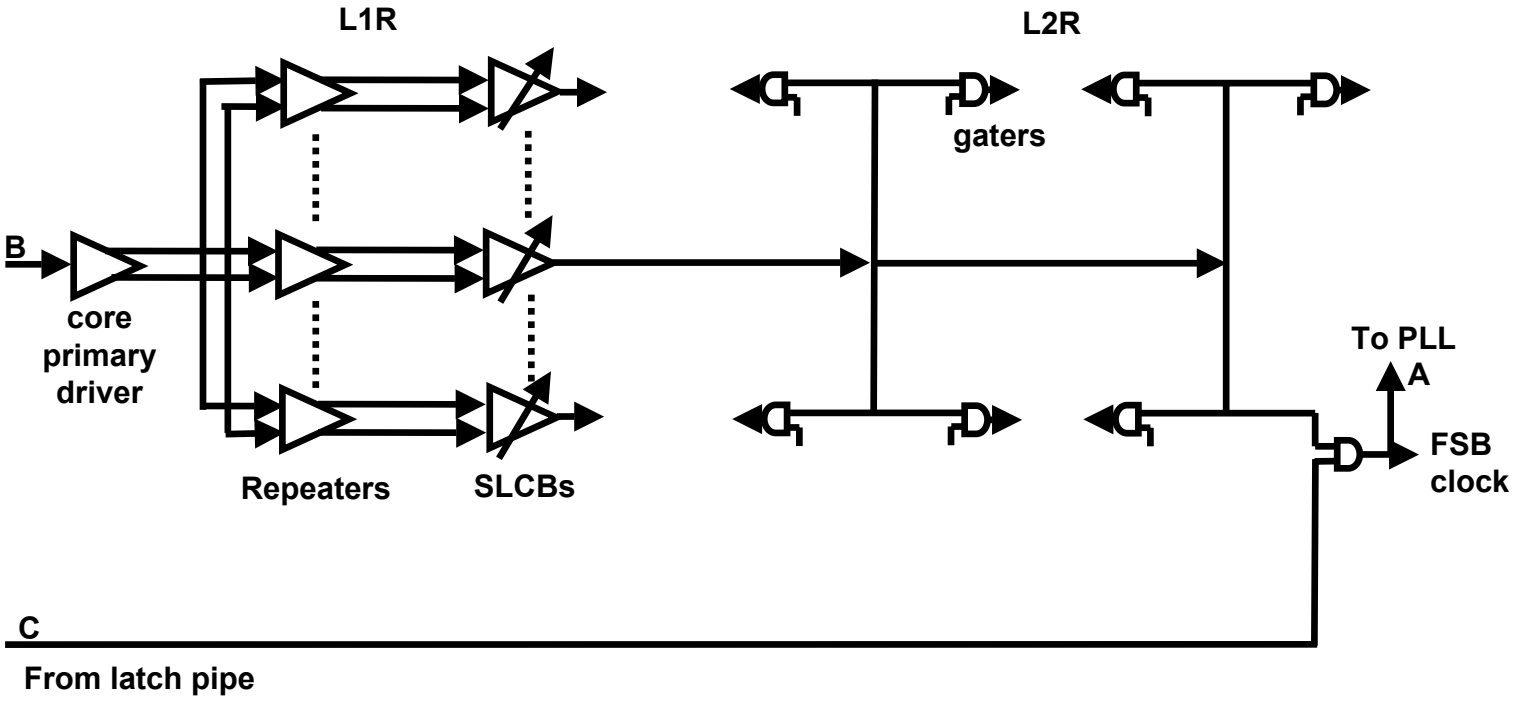


Figure 8.5.2: Core clock level one and level two distribution.

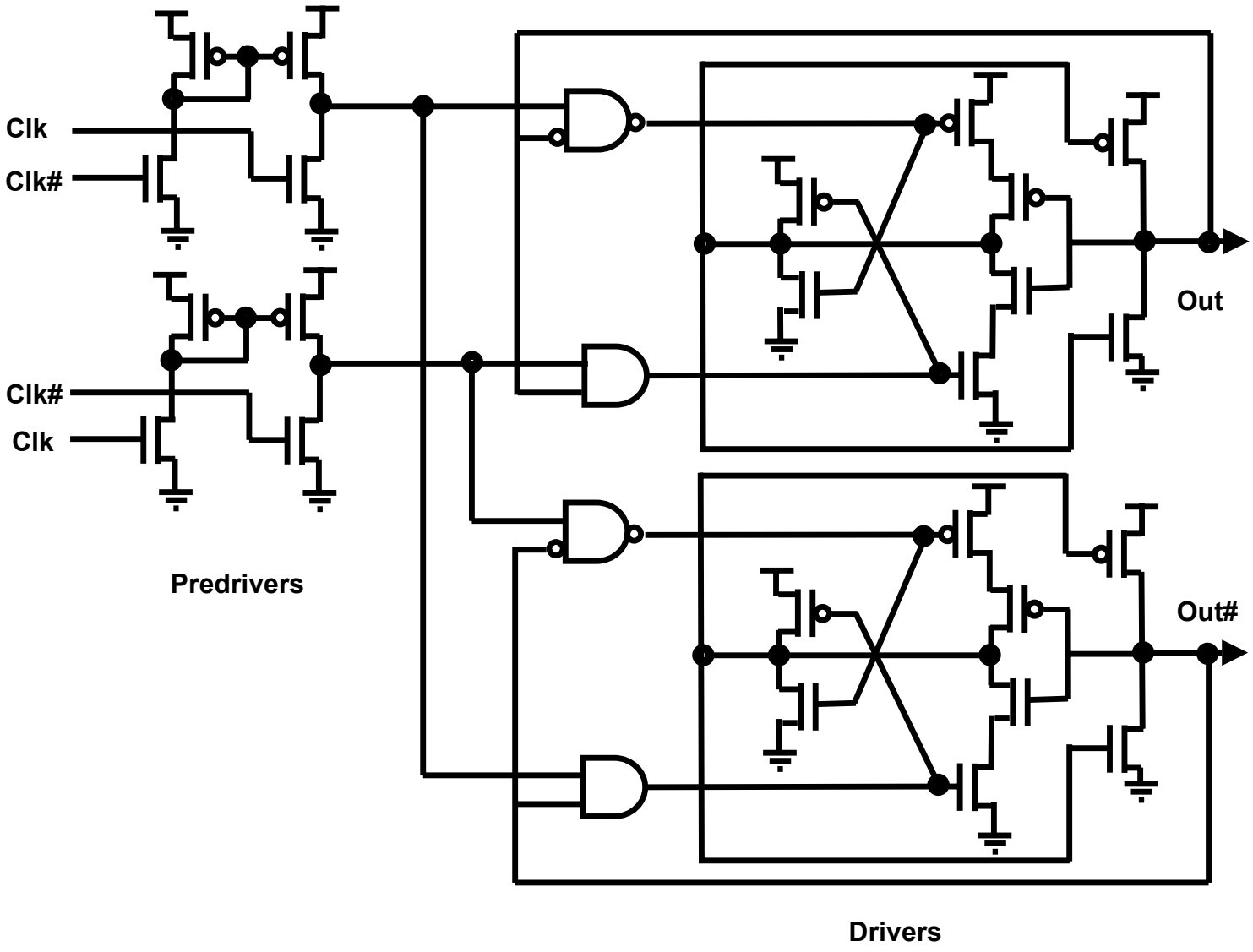


Figure 8.5.3: Primary driver/repeater/SLCB schematic.

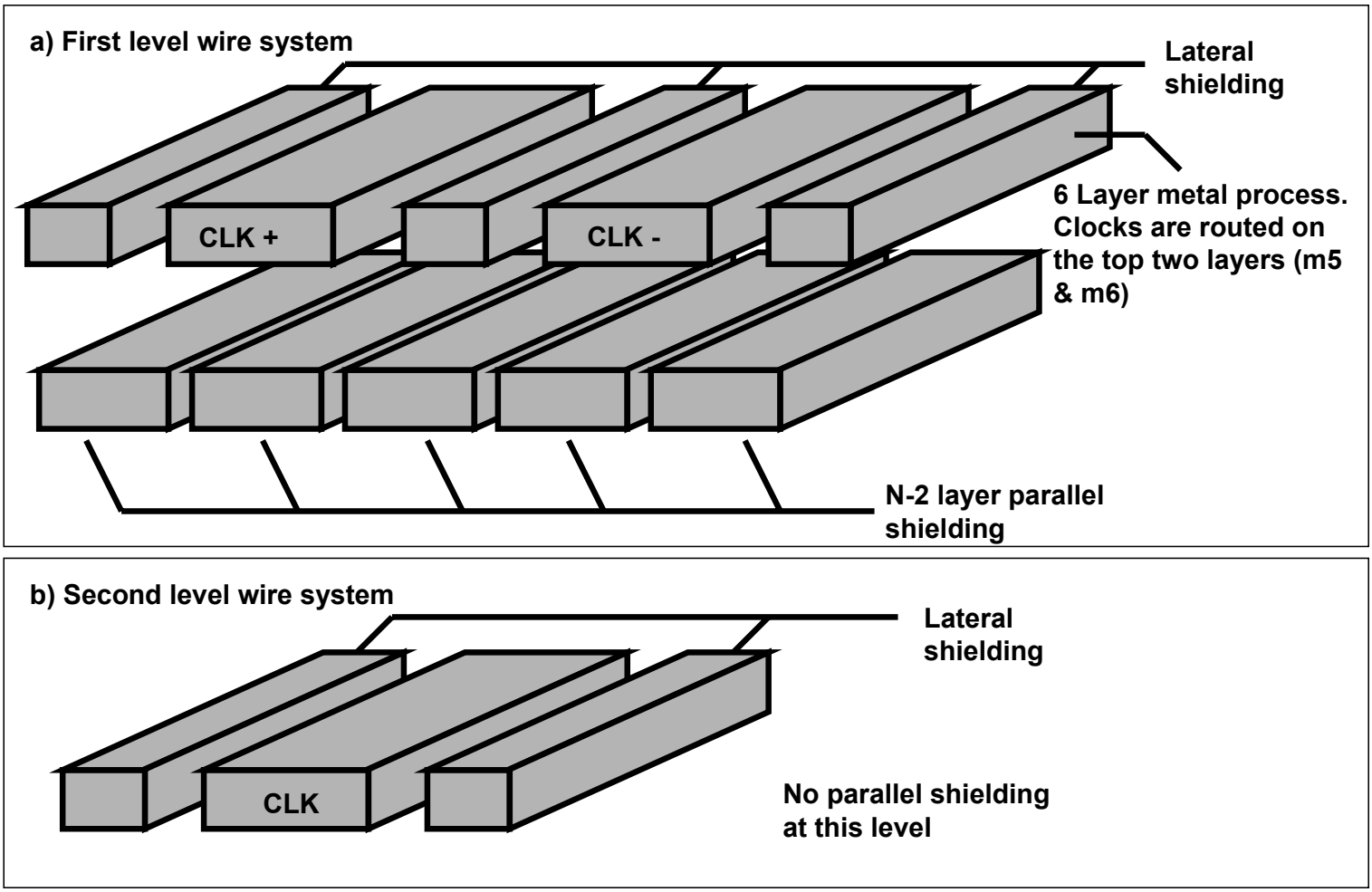
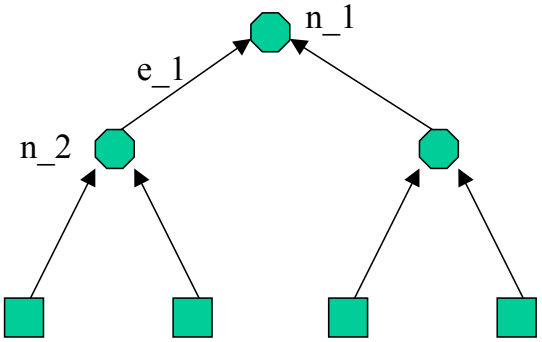


Figure 8.5.4: First and second level route geometry.



Leaf Node: Location and routes to local gaters determined by block designer.
Cap load of gaters extracted.
ActD to represent simulated delay.
AvgD to represent average delay.

Branch Node: Location of branch in L2R determined by tool.
ActD to represent simulated delay.
AvgD to represent average delay.

Link: Width and length of route between nodes determined by tool.

De-skew Algorithm applied per iteration:

AvgD(n) = [AvgD(left child of n) + AvgD(right child of n)]/2
ActD(n) = simulated delay to n from root of tree
Wid(e) = width of an edge
Del(edge) = delay between nodes connected by edge
Del_to_Wid(edge, delay) = estimated width to cause delay between edge
CONST = convergence factor

Wid(e_1) = Del_to_Wid(e_1, Del(e_1) + CONST * ActD(n_2) * (1 - AvgD(n_1)/AvgD(n_2)))

Figure 8.5.5: Data representation of level two route.

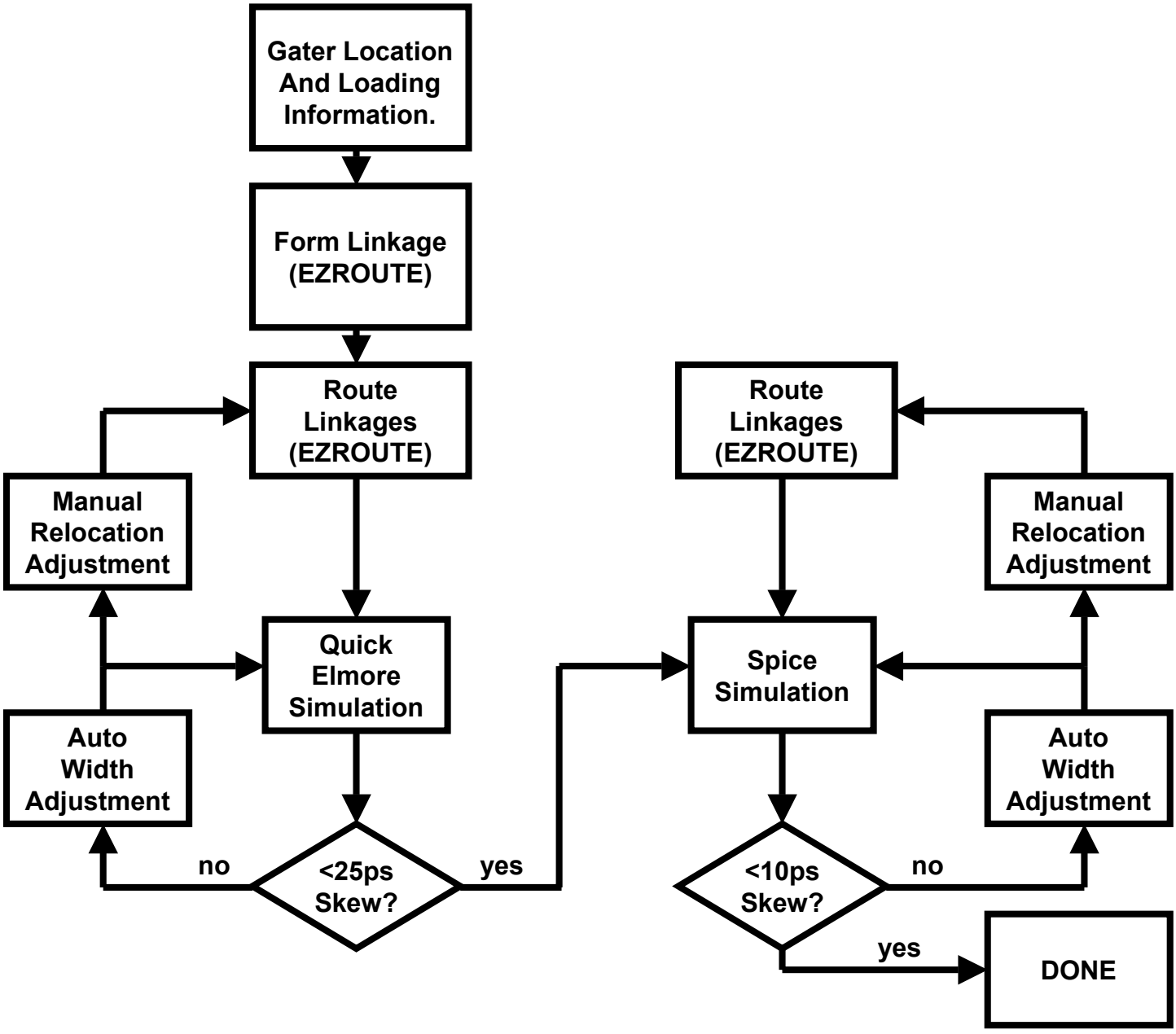


Figure 8.5.6: L2R routing flow diagram.

