A Design for Digital, Dynamic Clock Deskew

Charles E. Dike, Nasser A. Kurd, Priyadarsan Patra, Javed Barkatullah

Intel Corporation, Hillsboro, OR 97124

{charles.e.dike, priyadarsan.patra, nasser.a.kurd, javed.barkatullah} @intel.com

Abstract

Unintentional clock skews between clock domains represent an increasing and costly overhead in high-performance VLSI chips. We describe a novel yet easy-to-implement design that reduces skew between local clock domains dynamically or statically by sensing clock-delay differences and then tuning the clock of each domain relative to its neighbors. Lowering local clock skew is accomplished without compromising worst- case global skew.

Keywords: dynamic, mesh, deskew, clock, skew, VLSI

Introduction

Increased frequencies and higher levels of integration for microprocessors have posed significant challenges for clock generation and distribution. Large die area causes the distribution path to be long and widely dispersed across the die. This distribution produces large latency in the clock path which is greatly impacted by variations in loading on clock lines, temperature shifts, voltage swings, cross-talk and across die process fluctuations. These combine to create large clock skew and jitter that effectively shorten the clock cycle. Our goal is to reduce the clock skew to recover some of the clock cycle timing in order to improve performance.

With each generation of microprocessors the deskew schemes have gotten more complex. Initial clock deskewing schemes for the Intel Pentium® 2 [1] measured the timing relationship between two clock spines. Phase detectors sensed the difference and then delays within the appropriate clock path were adjusted to reduce the skew between the two clock domains. The Itanium® Processor [2] performed de-skewing in clusters using 8 digitally controlled delay-locked loops. Each DLL in turn was used to deskew three or four clock regions. Skew between regions controlled by separate DLLs was not constrained. At less than 1 GHz, this scheme was effective.

With an increase in the number of clock domains the Intel Pentium® 4 [3] used a modified method with three levels of phase detection with a total of 46 detectors. This was a static scheme that was subject to errors imposed by temperature and voltage variations. While effective for the Pentium® 4 with a medium clock frequency of 2-3+ GHz, it may not meet the needs of still larger die sizes running at much faster clock speeds. This proposed method was therefore developed to deskew a microprocessor using a mesh structure where the domain clocks use a distributed algorithm to adapt dynamically to temperature and voltage variations.

The H-tree deskew section describes one proposed deskew solution along with its weaknesses. In the mesh deskew section the algorithm is discussed and the general implementation scheme is explained along with timing considerations followed by the results and summary.

H-tree Deskew

A clock distribution network in a modern VLSI chip often takes the form of an H-tree. Figure 1(a) shows an H-tree clock distribution with the gray boxes representing phase detectors and buffers, the chip itself being divided into 16 domains labeled from A to P. The clock originates at box 4, is passed to the pair of boxes labeled 3, then goes to 2, 1, and then into the final distribution points represented by the white boxes. The desire is to match delays of each leg of the tree by construction. Unfortunately, variations in clock delays due to process, supply, temperature, and load can cause significant skews between neighboring leaves of the tree. This is the reason that deskew is needed. Figure 1(b) shows the upper right quadrant of Figure 1(a) in more detail. The buffers, which are tunable, are now displayed as separate from the phase detectors and the phase detectors are not in the clock path but sit on domain boundaries. The H-tree distribution concept can extend into each domain as shown by the gray lines. As an example, phase detector 1 (between domains C and G) receives input from clock points local to the detector from both domains. The phase detector then sends tuning information to the tunable buffers of both domains. This information tends to zero out the phase difference in the vicinity of the phase detector. Each phase detector reduces the skew between the two measurement points to within some

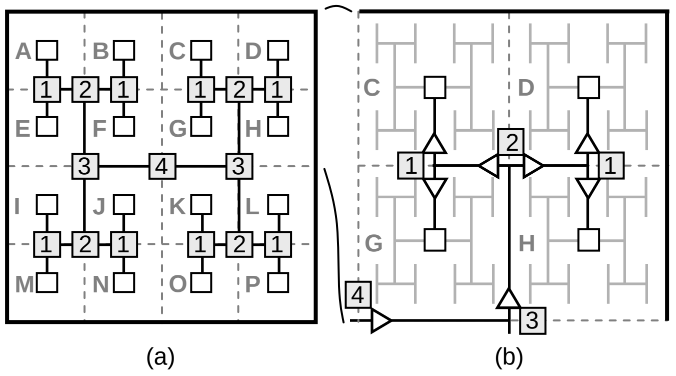

Figure 1: H-tree deskew

guardband **g**. In Figure 1 the path from C to G passes through one phase detector so the maximum difference between the domains is one guardband. In this manner the domain pair [C, G] is zeroed along with domain pair [D, H]. Continuing the zeroing, phase detector 2 then compares [C, G] with [D, H] and adjusts its local buffers. This process continues similarly through phase detector 3 comparing [C,D,G,H] with [K,L,O,P] and so forth.

There are a couple of difficulties with this deskew method. H-tree deskew is hierarchical so phase detector 1 must be zeroed prior to zeroing phase detector 2, 2 must precede 3, and 3 must precede 4. Further, the clock must pass through a chain of tunable buffers on the path to the local domains. Tunable buffers add additional latency to this clock path and thus could worsen clock skew and jitter. A potentially more serious problem is that across some domain boundaries that are physically adjoining, the electrically tuned path can be long. For example, domains B and C in Figure 1(a), are neighbors but the possible accumulated skew offset is the sum of the guardbands in seven phase detectors (7***g**). Thus a signal crossing between B and C will lose 7***g** in computation time to clock overhead potentially leading to significant performance loss.

Mesh Deskew

Figure 2 shows a mesh structure for deskewing. Each striped box represents a phase detector between two adjacent domains in the clock distribution network. The compensators (numbered white boxes) represent the tunable buffers in the final leaf nodes of the distribution network. The clock itself could be delivered to these white boxes using an H-tree distribution but the mesh deskew would not require tunable buffers until the clock arrived at the compensators. The compensators have up to four inputs, one from each cardinal direction, to control the clock domain timing. The boundary domains will have only two or three detector inputs. These inputs are essentially OR'ed to produce the control signal to adjust the delay buffer in a specific domain. Because only one phase detector separates neighboring domains, the maximum delta between any two neighbors is one guardband. Figure 3 is a flowchart diagram describing the algorithm for each clock domain. This distributed algorithm is simple to implement in hardware.

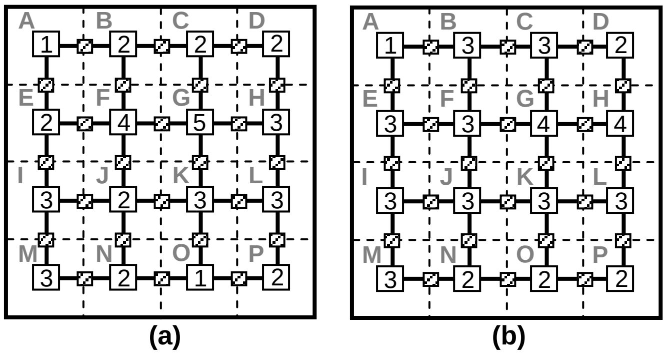

Figure 2: Mesh deskew

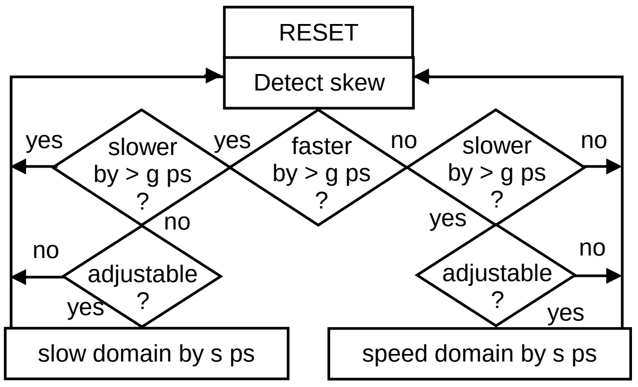

Figure 3: Mesh algorithm

The rules at the heart of the algorithm have four cases. Case 1 (2) - if a domain is slower (faster) than any neighbor by more than guardband **g** and not faster (slower) than any neighbor by more than **g,** that domain will reduce (increase) its delay by a step size **s** $\leq$ **g** on a clock edge.

Case 3 - if a domain is faster than at least one neighbor by more than **g** and also slower than at least one neighbor by more than **g** then that domain will maintain its original delay.

Case 4 - if a domain is separated by less than **g** from all its neighbors, that domain will maintain its original delay.

Figure 2(a) shows an *ungroomed* mesh deskew. In this example, let guardband **g** be $1+\delta$ and step size **s** be 1. The compensators with the smallest numbers represent the fastest domains. Domain G is an example of case 1, domain J is an example of case 2, domain K is an example of case 3, and domain L is an example of case 4. Adjustment criteria is collected for all domains simultaneously on a clock edge and later applied to all domains simultaneously. Figure 2(b) shows the die after a single adjustment. Notice that domains A and D, which initially required no adjustment, now require a tweak.

Ultimately the mesh will resolve to a *stable* position, where clock skew between two neighbors is less than or equal to the guardband **g**. This example only requires one more iteration wherein A will increase to 2 while its neighbors B and E decrease to 2, and D will increase to 3 while H decreases to 3. All other domains will remain stable in the last iteration.

Figure 4 shows the local compensator receiving inputs from each of the four cardinal phase detectors. The compensator works by shifting ones to the right when the local domain is fast, thus slowing the local domain. If the local domain is slow, zeroes are shifted to the left. Once a limit (either fast or slow) is reached, any attempted shifting in the same direction has no impact on the delay line. If both shift right and shift left signals are generated the shift register does not change. The number of delay adjustments **d** is dependent on the number of bits in the shift register and adjustable buffer. While not a requirement, the design described here starts with the shift registers set to the middle position. The initial adjustment to any delay buffer can work to either slow it down

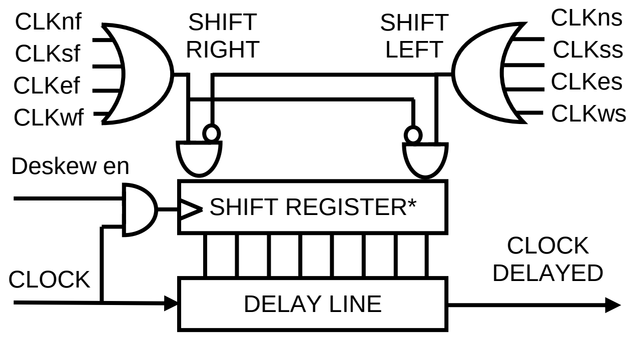

Figure 4: Compensator

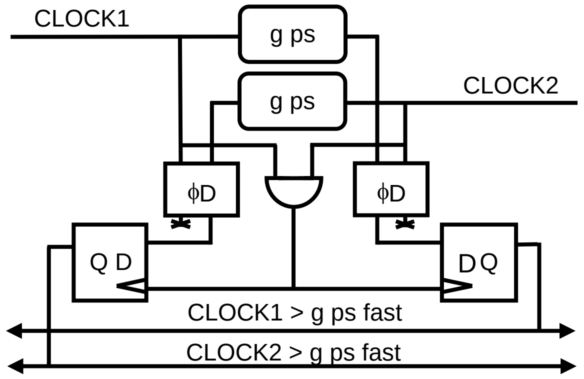

Figure 5: Delay detector

or speed it up. This being said, when the shift register begins in a middle position it will typically resolve to a stable state faster than a deskew scheme where the shift registers are initialized to one end or the other.

The values of **g**, **d**, and **s** need to be chosen based on several trade-offs. Large guardbands have a negative impact on the clock cycle because the guardband must be subtracted from the communication time. On the other hand, a small guard-band is more susceptible to the effects of jitter since jitter between two sensed clocks could cause false triggering of the delay detector. The step size **s** combined with the number of adjustment iterations possible **d,** determines the maximum adjustment range. Thus, a step size just less than the guardband **g** allows the largest adjustment range. Discretion is needed in the choice of step size because if **s** is larger than **g** the circuit is unstable. A reasonable design will have a step size approaching the guardband size with the adjustment iterations chosen to meet a maximum across-die skew target. Of course, the larger **d** costs in terms of area and power.

The delay detector (Figure 5) latches the outputs of a pair of phase detectors (ϕ**D**). The devices receive identical clocks but the transistor sizing in the inputs has been adjusted to change the thresholds in order to produce a guardband of **g** ps between the inputs. Simulations on a 0.13-micron process indicate that a guardband between 3 ps and 12 ps can be easily designed.

Metastability [4] is an issue in this design because of the phase detector timing in Figure 5. A phase detector is essentially an S-R latch that allows only the first arriving signal to pass through to an output. If two signals arrive simultaneously the phase detector could enter an unstable (metastable) state causing a longer than normal time to resolve. This can produce an ambiguous signal in the flops just as they are clocked and drive them into a metastable state. A metastable output from the delay detector could propagate into the shift register. This can cause the shift register to scramble its data and cause random delays to occur in the delay buffer. To prevent this from occurring, additional settling time could be added on the output of the delay detector.

A point that needs some discussion is the notion of mode lock [4][6]. Mode lock is where the clocks between neighbors are shifted so dramatically as to cause the neighbors to attempt to lock on different clock cycles. In the deskewing mesh, if this were the case the circuit would become unstable and could not recover (this would be a problem for any other deskew scheme also). This could happen if the initial difference between the neighboring tiles were ~½ clock period or greater. In Figure 5, if clock 1 arrived much later than clock 2, the delay detector could interpret the result as clock 2 arriving later than clock 1 because the delay detector got shifted by a clock cycle. Once this occurs the circuit cannot correct the problem. Practically, a circuit would not be expected to have such dramatic skews and the adjustment budget would be less than ½ period at any rate. But, if this were a concern, perhaps the simplest way to avoid the problem would be to pre-tune the circuit at a slow frequency thus preventing mode lock. In other words, if the two clocks in figure 5 were precisely 180^o out of phase at a specific frequency (assuming a 50% duty cycle), by cutting the frequency of the clocks in half, the phase difference would become 90^o (the rising clock edges would remain separated by the same amount of time). Enabling the mesh at the lower frequency would allow the deskew circuitry to reduce the phase difference to a point where the frequency could then be increased and tuned again.

Results

Circuit simulations were done with guardband **g** and step size **s** set to about 3.3 ps and 1.5 ps respectively. With **d** at 16, the maximum adjustment across domains was **d*s** =24 ps. Figure 6 shows the timing of the clock at one node of a 16 domain mesh relative to its bottom, left, right and top neighbors prior to activating the deskew. The center domain is labeled D34 and is removed from its farthest neighbor by 9.94 ps. The nodes displayed in Figures 6 and 7 are labeled in Figure 8. After adjustment Figure 8(b) shows the worst case local skew for D34 with its neighbors drops to about 2.5 ps. This adjustment took about 25 iterations to settle to a stable state.

The maximum number of iterations required for stability is dependent on the initial clock divergence over the whole die. We have developed an empirical formula of **m*n+m+n+(d/2)(m+n-2)**

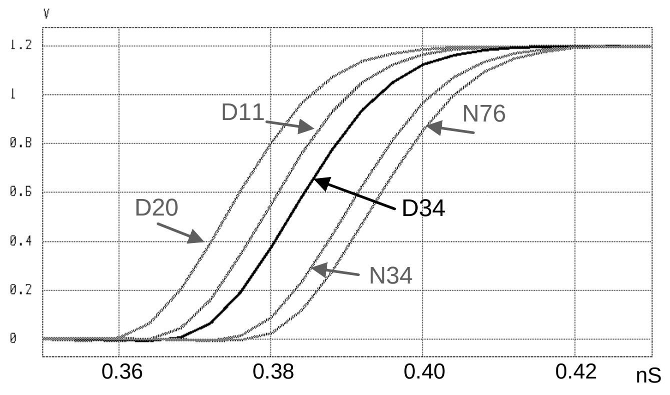

Figure 6: Local clock timing before deskew

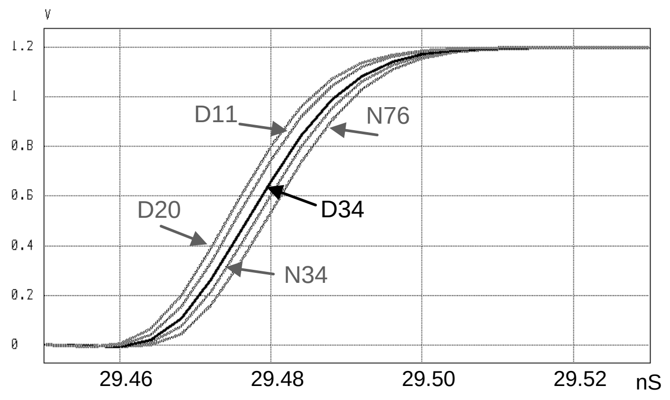

Figure 7: Local clock timing after deskew

to determine this number. Here **m** and **n** are the clock domain counts in the vertical and horizontal directions. This formula is accurate for 10 x 10 meshes or smaller but we have discovered that for larger meshes the formula underestimates the maximum number of iterations. In our previous example we were using a 4 x 4 mesh. The bound above was based on the assumption that the deskew compensator started with the shift registers centered. If the shift registers were at their limits at initialization the bound would be **m*n+m+n+(d)(m+n-2).**

Figures 8(a) and 8(b) use the lower left hand corner domain as the reference point by which all other clock skews are measured. In Figure 8(a) the total skew before adjustment between the fastest and slowest domains across the die is 23 ps and the worst case local skew is about 11 ps. The maximum skew across the die that can be compensated for is (**m**+**n**-1)(**g**-**s**)+**d*s**. Finally, Figure 8(b) shows the global clock timing after deskew with the local and global skew minimized to about 3 ps and 13 ps, respectively.

Summary and Conclusion

The paper demonstrates a complete deskew mesh network which minimizes clock skew to within an arbitrarily small guardband while still settling to a stable state. One major advantage of the mesh deskew over other deskew configurations is its ability to lower local clock skew to a minimum guardband **g.** In contrast, the local skew in the H-tree deskew method could be as high as (2***h**+1)***g** where **h** is the number of hierarchal levels. Furthermore, the mesh scheme requires tunable final leaf nodes that add minimum latency. The latency penalty for the H-deskew is much higher and depends on the number of hierarchies. For a mesh network, the number of phase detectors asymptotically approaches two per domain as the number of domains grows. With 16 domains, 24 phase detectors are needed for a fully populated mesh. With 64 domains, 112 phase detectors are needed.

13.94	21.38	18.39	22.98
11.39	N34 16.80	21.83	14.40
D20 1.41	D34 9.87	N76 19.81	11.40
0	D11 6.45	8.49	8.99

(a)

7.22	7.80	10.13	12.88
4.62	N34 5.39	8.27	11.31
D20 1.49	D34 4.31	N76 6.76	9.14
0	D11 2.66	5.16	7.29

(b)

Figure 8: Timing - before and after

The basic principles for mesh design have been outlined. The authors recognize the potential to improve the implementation presented in the paper and are continuing with that work. A study of the effects of jitter needs to be performed in order to determine the final size of the guardband. This could impact the choice of making the mesh dynamic or static. A static version would deskew during the reset/startup of the chip and then hold the skew state during normal operation. This would only compensate for process variations and design deviations. A dynamic deskew would remain enabled during normal operation and constantly adjust for voltage and temperature variations.

The H-tree clock deskew configuration can lead to both large local and large global skews. On the other hand, the mesh deskew is designed to minimize the local skew without compromising global skew. Also, the mesh deskew is stable with all physically local skews reduced to at most one guardband. At high frequencies the importance of global deskew is diminished because signals traveling long distances must pass through clocked repeaters along the way. However, the mesh deskew method additionally guarantees an upper bound of **n*g** on the clock skew if a signal crosses **n** clock domains with a guardband **g**.

Bibliography/References

[1] G. Geannopoulos and X. Dai, "SP 25.3: An Adaptive Digital Deskewing Circuit for Clock Distribution Networks," ISSCC, 1998. Digest of Technical Papers. pp 400 –401.

[2] S. Tam, S. Rusu, U. N. Desai, R. Kim, J. Zhang, and I. Young, "Clock Generation and Distribution for the First IA-64 Microprocessor," *IEEE J. Solid State Circuits*, vol. 35, no. 11, pp 1545-1552, Nov. 2000.

[3] N. Kurd, J. Barkatullah, R.Dizon, T. Fletcher, and P. Madland, "A Multigigahertz Clocking Scheme for a Pentium 4 Microprocessor," *IEEE J. Solid State Circuits*, vol. 36, no. 11, pp 1647-1653, Nov. 2001.

[4] V. Gutnik and A. P. Chandrakasan, "Active GHz Clock Network Using Distributed PLLs," *IEEE J. Solid State Circuits*, vol. 35, no. 11, pp 1553-1560, Nov. 2000.

[5] L. Kleeman and A. Cantoni, "Metastability Behavior in Digital Systems," *IEEE Design and Test of Computers*, pp.4-19, Dec. 1987.

[6] G. A. Pratt and J. Nguyen, "Distributed Synchronous Clocking," *Parallel and Distributed Systems, IEEE Transactions on*, Volume: 6 Issue: 3, March 1995, pp 314 –328.