Synchronous Vs Asynchronous Design

Introduction:

Much of today's logic design is based on two major assumptions: all signals are binary, and time is discrete. Both of these assumptions are made in order to simplify logic design. By assuming binary values on signals, simple Boolean logic can be used to describe and manipulate logic constructs. By assuming time is discrete, hazards and feedback can largely be ignored. However, as with many simplifying assumptions, a system that can operate without these assumptions has the potential to generate better results.
Asynchronous circuits keep the assumption that signals are binary, but remove the assumption that time is discrete. This has several possible benefits:

No clock skew - Clock skew is the difference in arrival times of the clock signal at different parts of the circuit. Since asynchronous circuits by definition have no globally distributed clock, there is no need to worry about clock skew. In contrast, synchronous systems often slow down their circuits to accommodate the skew. As feature sizes decrease, clock skew becomes a much greater concern.

Lower power - Standard synchronous circuits have to toggle clock lines, and possibly pre-charge and discharge signals, in portions of a circuit unused in the current computation. For example, even though a floating point unit on a processor might not be used in a given instruction stream, the unit still must be operated by the clock. Although asynchronous circuits often require more transitions on the computation path than synchronous circuits, they generally have transitions only in areas involved in the current computation.
Note that there are some techniques in synchronous design that addresses this issue as well.

Average-case instead of worst-case performance - Synchronous circuits must wait until all possible computations have completed before latching the results, yielding worst-case performance. Many asynchronous systems sense when a computation has completed, allowing them to exhibit average-case performance. For circuits such as ripple-carry adders where the worst-case delay is significantly worse than the average-case delay, this can result in a substantial savings.

Easing of global timing issues : In systems such as a synchronous microprocessor, the system clock, and thus system performance, is dictated by the slowest (*critical*) path. Thus, most portions of a circuit must be carefully optimized to achieve the highest clock rate, including rarely used portions of the system. Since many asynchronous systems operate at the speed of the circuit path currently in operation, rarely used portions of the circuit can be left un-optimized without adversely affecting system performance.

Better technology migration potential - Integrated circuits will often be implemented in several different technologies during their lifetime. Early systems may be implemented with gate arrays, while later production runs may migrate to semi-custom or custom ICs. Greater performance for synchronous systems can often only be achieved by migrating all system components to a new technology, since again the overall system performance is based on the longest path. In many asynchronous systems, migration of only the more critical system components can improve system performance on average, since performance is dependent on only the currently active path. Also, since many asynchronous systems sense computation completion, components with different delays may often be substituted into a system without altering other
elements or structures.

Automatic adaptation to physical properties - The delay through a circuit can change with variations in fabrication, temperature, and power-supply voltage. Synchronous circuits must assume that the worst possible combination of factors is present and clock the system accordingly. Many asynchronous circuits sense computation completion, and will run as quickly as the current physical properties allow.

Robust mutual exclusion and external input handling - Elements that guarantee correct mutual exclusion of independent signals and synchronization of external signals to a clock are subject to *meta-Stability* . A metastable state is an unstable equilibrium state, such as a pair of cross-coupled CMOS inverters at 2.5V, which a system can remain in for an unbounded amount of time [2]. Synchronous circuits require all elements to exhibit bounded response time. Thus, there is some chance that mutual exclusion circuits will fail in a synchronous system. Most asynchronous systems can wait an arbitrarily long time for such an element to complete, allowing robust mutual exclusion. Also, since there is no clock with which signals must be synchronized, asynchronous circuits more gracefully accommodate inputs from the outside world, which are by nature asynchronous.

With all of the potential advantages of asynchronous circuits, one might wonder why synchronous systems predominate. The reason is that asynchronous circuits have several problems as well**:**

Asynchronous circuits are more difficult to design in an ad hoc fashion than synchronous circuits. In a synchronous system, a designer can simply define the combinational logic necessary to compute the given function, and surround it with latches.

By setting the clock rate to a long enough period, all worries about hazards (undesired signal transitions) and the dynamic state of the circuit are removed. In contrast, designers of asynchronous systems must pay a great deal of attention to the dynamic state of the circuit. Hazards must also be removed from the circuit, or not introduced in the first place, to avoid incorrect results.

The ordering of operations, which was fixed by the placement of latches in a synchronous system, must be carefully ensured by the asynchronous control logic. For complex systems, these issues become too difficult to handle by hand.

Unfortunately, asynchronous circuits in general cannot leverage off of existing CAD tools and implementation alternatives for synchronous systems.

For example, some asynchronous methodologies allow only algebraic manipulations (associative, commutative, and De-Morgan's Law) for logic decomposition, and many do not even allow these.

Placement, routing, partitioning, logic synthesis, and most other CAD tools either need modifications for asynchronous circuits, or are not applicable at all.

Finally, even though most of the advantages of asynchronous circuits are towards higher performance, it isn't clear that asynchronous circuits are actually any faster in practice. Asynchronous circuits generally require extra time due to their signaling policies, thus increasing average-case delay.