

Chip Design - better asynchronous than synchronous?

Solving the problem with NCL - an extended Boolean algebra

As SoCs get bigger and performance is more demanding, the normal synchronous implementation gives designers severe problems. Synchronizing different clock domains, the current spikes generated by fast switching and contact drop inside the chip are all complex. Why not implement some or all of the design asynchronously? Null Convention Logic (NCLTM) technology, developed by Theseus Logic, may help find the right answers.

Design problems

Boolean logic is an abstract description of the digital part of the chip and does not take into account reality: for example, it does not provide for timing delays or race conditions. These have to be resolved by the engineer using experience to guide design, using synthesis and simulation tools and by test. Up to a certain chip complexity this can be planned for in the schedule, but with the ever increasing number of gates used in

FIG. 1 Synchronous State Machine (FSM Finite State Machine), status held in registers

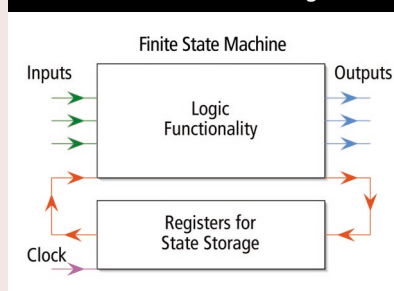
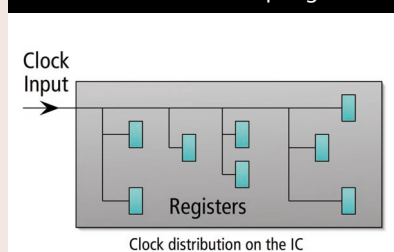


FIG. 2 Clock Tree, synchronizing all of the on-chip registers



modern SoCs this is getting more difficult and less predictable, often delaying design schedules.

Another issue is the use of Intellectual Property (IP), using existing functionality including recycled parts of older designs or bought-in from third parties. In both cases the inner workings may be unknown, as the supplier describes only the functionality. With synchronous design this can lead to significant timing problems.

Worst-case slows design

All – or nearly all – SoCs are designed in synchronous technology (Fig. 1). Using the latest silicon technology with very small line widths can lead to serious issues, including:

- **Current spikes:** in a synchronous system all of the internal registers use the same clock and switch at the same time. This leads to on-chip current spikes that, in turn, cause voltage drops on the supply. These drops can be so large that gates change state and cause false signals. Some of these effects can be tested only after manufacturing.
- **Clock tree complexity:** As the same clock controls all registers, a clock line must connect all the registers on the IC (Fig. 2). Long connections need additional buffers to maintain the sharp clock edges, costing extra real estate and introducing clock delays. The clock tree is always active and consuming current, even if the chip is in an idle state waiting for new data to be processed. It is often forgotten that the clock line is the longest connection on a chip only comparable with the length of the reset line.
- **“Worst-case” design rules:** All the tools involved in the design are normally set up assuming worst case, such as the lowest supply voltage, highest temperature and any parameters involved in silicon manufacturing. And the design tools will add extra safety margins. This helps to assure functionality but ensures that the chip will never run at the maximum speed possible for the silicon technology used.

It is as though a 100 metre runner were only allowed to run at a maximum speed defined by the coach, not the speed that the runner has the potential to achieve.

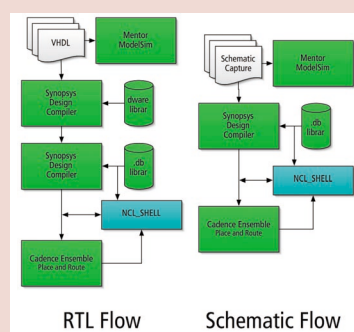
A better and less power hungry way would be to have the IC process data at maximum speed and then switch to power save mode.

There are solutions

There are various routes to address these issues, most of them using various flavours of asynchronous design. If we look at logic design in general, not just synchronous design, Table 1 shows some of the possibilities which are in use already – mostly if there is no other solution. Synchronous is not the opposite of asynchronous but a special case of asynchronous design. (On the other hand, one has to accept, that it was the synchronous paradigm enabling the fast growth of complexity on-chip.)

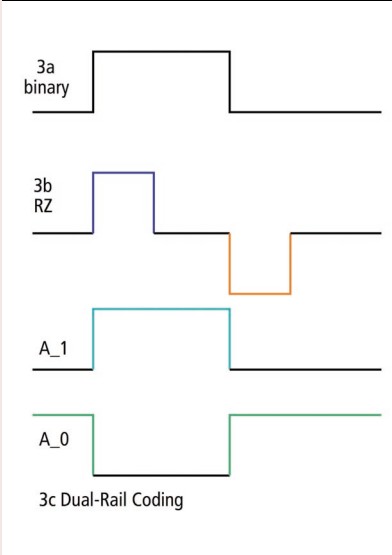
To take the next step in design complexity

NCL design flow based on standard tools



Theseus has prepared a design flow based on commercially available standard tools and is as close as possible to a normal VHDL design flow: description in VHDL, simulation with Mentor ModelSim, synthesis using Synopsys Design Compiler and Cadence Ensemble for placement.

FIG. 3 3 different coding methods:
a) binary,
b) RZ - return to zero,
c) Dual Rail



requires extensions to the design tool chain to enable three aspects:

- using the silicon for more functionality but with less power consumption
- getting the design done more quickly
- avoiding respins of designs to correct design errors

One way to achieve this is to start from the basis of Boolean Logic and extend it to cope with the real world. This is the approach of Theseus Logic with NCL (Null Convention Logic) technology. The designer continues to use the same HDL design languages and lets the NCL Shell translate into asynchronous synthesis executed by gates, much like using a C compiler to translate a C program into machine language for microprocessor execution.

The main points of NCL

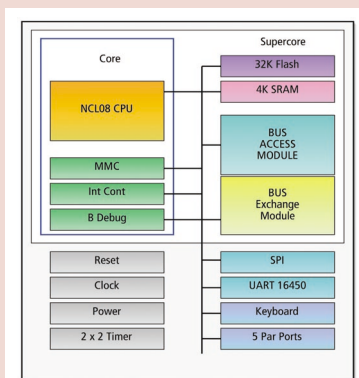
How do these NCL extensions to Boolean logic work? The following overview concentrates on principles, not the details, but it is the principles that make this type of technology the only way forward in increasing complexity on-chip while maintaining design schedules.

The first step is to include the clock into each wire and then to throw the clock away completely. A radical way of achieving synchronisation is to embed the clock signal into every signal. This is not new: many variations, for example RZ (return to zero), are used in serial transmission. Fig. 3a shows the digital signal and

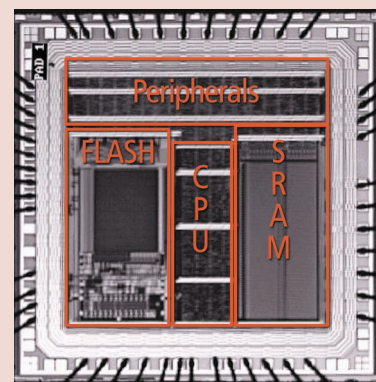
TABLE. 1 Some aspects of asynchronous design implementation

Aspect	Description
Data transfer via signal level	HIGH and LOW as in conventional logic. The transition from one level to the other can cause problems depending on the slope. This intermediate state defines a third level, e.g. "unknown."
Data transfer using the signal edge	The signal is accepted only after a transition. Signal levels and slopes are not important anymore, they only cause some delay.
Data bundling	Data buses or groups of signals are accompanied by an additional clock line
Dual-Rail signal transfer	The HIGH and LOW signals are split up and carried via separate wires as active signals
Speed independence	The system computes independent of gate speed variations caused by supply voltage, temperature or manufacturing tolerances. Connection delays still cause sensitivity.
Delay insensitive	This implementation will achieve complete independence of any delays; only performance changes.

NCL086P32 8 Bit Microcontroller using Core in NCL



Instruction compatible with STAR08 from Motorola, it can run the same code as the 6805 and 6808 families. The NCL implementation consists of the NCL08 core, memory interface, interrupt unit and bus interface. It is extended to the Super Core by adding 32K FLASH, 4K RAM and a Local Bus Access Interface. This Super Core will



be the basis for additional custom designs.

Additional functions on-chip: keyboard interface, two 2 channel timers, UART, SPI interface and 32 bidirectional IOs. Packaged in a 44 pin QFP; the main difference to a synchronous implementation are 40% less power consumption and 11 dB less noise.

Fig. 3b the conversion into RZ: a positive transition of the binary signal generates a positive pulse; a negative transition leads to negative pulse. If there is no activity, no current flows. The binary data is recovered at the receiving end without transmitting the clock and delays on the line are not relevant anymore.

Implementing a similar concept on-chip will make the influences of temperature, supply

voltage and process variations combine at only one point: the final throughput or performance at maximum speed of the silicon technology used. Each activity chip is started either externally or internally, the data is processed and the outputs set and then the system jumps back into the quiescent state drawing minimum current.

Unfortunately this leads to a logic with

FIG. 4 Binary input split up into Dual-Rail with additional AND gates for NCL

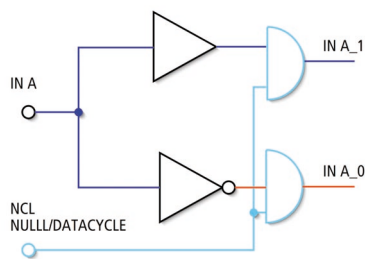
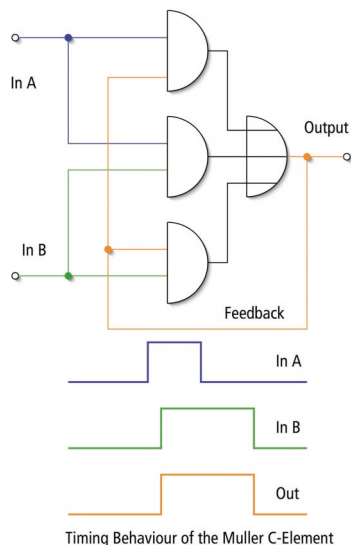


FIG. 5 The Muller C-Element works delay insensitive: HIGH when both on, LOW when both off.



Timing Behaviour of the Muller C-Element

three voltage levels (+, 0, -) which is not available as a commercial process. Instead, Dual-Rail implementation (Fig 3c) uses 2 wires per signal and moves from levels to activities, similar to RZ. One wire is used for + and one for -, 0 stays the same. (Having both lines active at the same time is not allowed.)

With Dual-Rail logic, systems are designed differently. An AND gate in NCL (Fig. 6) has to deal with Dual-Rail activities rather than logic levels.

The first step in a conversion splits a binary signal to Dual-Rail (Fig. 4) and adds gates required in NCL for the additional NULL state. Only one of the outputs is active at any time – or both are inactive.

The next step is to design a “decision point”

TABLE. 2 Basic NCL functions: AND, Or, NOT.

NCL AND	B TRUE	B FALSE	B NULL
A TRUE	true	false	previous
A FALSE	false	false	previous
A NULL	previous	previous	null

NCL OR	A TRUE	A FALSE	A NULL
B TRUE	true	true	previous
B FALSE	true	false	previous
B NULL	previous	previous	null

NCL NOT	A
B TRUE	false
B FALSE	true
B NULL	null

TABLE. 3 Comparison of transistor usage .

Module	synchronous	NCL	Ratio
AddrConv	41	62	1.5
X2VHD	395	826	2.1
Decoder	1010	1804	1.8
32x16_FIFO	1691	1123	0.7

TABLE. 4 Transistor usage in manual vs. synthesis

Function	Manual	Synthesis	Ratio in %
MUX	40	40	100
AND4	66	68	103
SET_CNT	238	208	87
SHIFT	506	248	56
SYN_State	1008	814	81
BIT_CNT	1059	794	75

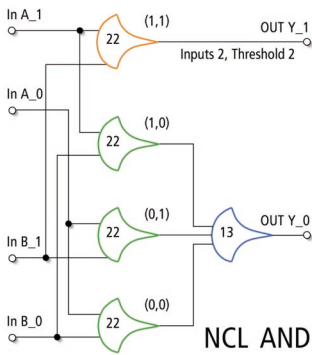
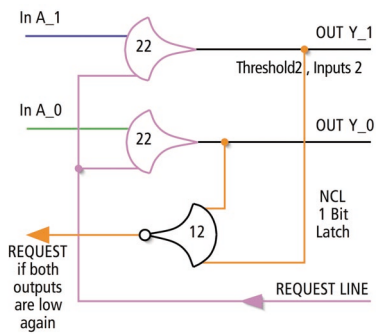
which combines “activities”. The Mueller C-element (Fig. 5) is probably the most well known implementation. The biggest difference to a conventional gate is the feedback path that guarantees delay insensitivity: normally both inputs are LOW, so the output is LOW. The first HIGH activity arrives and prepares the gate. The second HIGH activity switches the output and sets the feedback path HIGH, holding the output HIGH even after the first signal goes LOW again. Both signals have to return to LOW to reset the output. The actual timing relationship of the input signals is not relevant anymore, as long as there is sufficient overlap to generate the correct output signal. Now the NCLAND can be understood: Both inputs active achieves HIGH, the other three combinations ORed result in LOW as

in the AND gate.

The simplest gate in NCL is the inverter: here the activity lines are just crossed.

A latch function is implemented by using Request and Acknowledge (Fig. 7). Normally the input requests data. As soon as one input is active it is latched and changes the input request line. After completed processing the incoming REQUEST line changes to low and resets the latched data and a new cycle can start.

One potential issue could be resetting threshold gates when there is normally always an active rail. NCL copes this with by constantly switching the complete system between a DATA (or ACTIVE) phase and a NULL phase. Data is applied and generates processing. As soon as the processing has finished and data

FIG. 6 NCL AND gate**FIG. 7** NCL register keeps incoming information until reset via REQUEST LINE

reaches the end, a NULL phase follows before data can again be applied.

For the 100m runner: NULL – getting ready for the run. DATA – the results arrive at different times. NULL – preparation for the next run and so on.

Testing

There is a general belief that an asynchronous implementation of a system is more difficult to test than Synchronous. This is not the case with NCL. Before it can toggle from ACTIVE to NULL the processing has to be completed. If the processing isn't complete, then no toggling takes place. Designs in NCL are often easier to test compared to synchronous implementations as errors in the design will stop the circuit from toggling and the engineer can trace the design error.

The main points

As NCL does not need the complex clock tree, design is easier and faster. Less activity on the chip results in reduced current consumption. Asynchronous implementation works better with analogue circuitry on the same chip,

Advantages of using NCL in specific Applications

Smartcards

These contain personal and card specific custom data in memory which are needed for access purposes. With a smartcard designed in synchronous technology, it is possible to guess at the stored data by "listening" to current consumption and EMI. As transmission protocol and interaction activities are known, it is possible to assign a meaning to the different extracted patterns. This is a very complex task but possible. Using NCL changes the picture: the different parts of the chip work independently and speed changes with voltage and temperature. Listening results look like random noise and emit less EMI, leading to increased protection against deciphering of the chip data.

Smartcard example

Infineon owns an NCL license and has implemented the DES algorithm with 64 bit data block length using a 56 bit key. The code consists of 5000 lines VHDL that were synthesised using the NCL shell. The chip has been manufactured in a 0.13m CMOS process and is currently under evaluation.

Low current consumption systems

NCL technology improves performance in all areas. Direct improvement is less current consumption in the digital circuits compared to existing designs. Less current peaks produce indirect benefits of better

performance in all four analogue functions on-chip (microphone amplifier, A/D, D/A and output amplifier).

Increased performance for ICs with fast analogue and digital functions

NCL generates on-chip random noise rather than synchronous current peaks that degrade the specifications of analogue on-chip parts. The implementation of digital functionality in NCL will always reduce noise. The example of a microprocessor shows values 11dB lower noise compared to a synchronous implementation.

Better sensitivity for wireless ICs

Less noise generated by the on-chip logic will improve the specifications of amplifiers, mixers and VCOs. One example would be in Bluetooth technology for wireless transmission to microphone and earpiece. Less current consumption reduces battery weight on the ear and less noise eases design of the analog blocks.

Interpolating D/A converters

These products are used in high volume in mobile phone base stations. The main blocks are digital FIR filters containing multipliers. Even though the data is transmitted synchronously, NCL design leads to reduced noise levels in the DAC block that is fed with the digital results.

achieving less spikes and better performance of the analogue circuitry, important for high speed mixed signal chips. Using asynchronous logic as "IP glue" reduces clock dependence and makes the inner workings of the IP less important for the designer.

On the other hand, NCL needs more gates and wires, which increases chip size for the logic. And engineers are used to the synchronous design and test methods.

The real question is not: synchronous or NCL. A better way to phrase it would probably be: where are the advantages of reduced design time and easier implementation of internal or external IP lead to so much of a saving, that the additional investment of the learning phase is minor compared to the gain.

Useful Links:

NCL technology:

www.Theseus.com

Synthesis:

www.synopsys.com/

Place and route:

www.cadence.com

NCL Processor:

www.theseus.com

Simulated examples:

www.exemark.com