Gusev Stanislav

Asynchronous circuits as alternative way of digital computing

Introduction

Today, all digital circuits are synchronous. Global clock is used to define the moment, when all data is ready and can be latched. This seems like a good solution. So, all CAD tools and technologies are oriented to synchronous design.

We can see a rapid increasing of complexity of Systems-On-Chips and integration degree of crystals. Now, it’s a problem to distribute a clock all over the crystal and ensure suitable clock skew in all points. For this purpose, it is necessary to use special buffers. So, synchronization can takes up to 30 percent of the crystal area and power consumption.

This is not the only disadvantage of the synchronous circuits. Global synchronization artificially limits the performance of the circuit. The clock period time can’t be less then maximum combination delay all over the circuit. Moreover, synchronization makes all triggers switching, even if there aren’t needs to latch new data. This leads to extra power consumption.

Now, institutes and companies are searching for principles and methods of digital computing without clock. For example, in the late 90 Philips made commercial asynchronous processors for their pagers and first mobile phones. In 2004 the first fully asynchronous ARM-handshake processor was produced.

This paper contains description of some modern methods of clockless digital systems design.

The main principle of asynchronous design

The main principle of asynchronous circuits design is handshaking. Registers can define moments, when they are ready to transmit or receive data without global synchronization. Two additional signals, request and acknowledgement, are implemented for this purpose (fig.1).

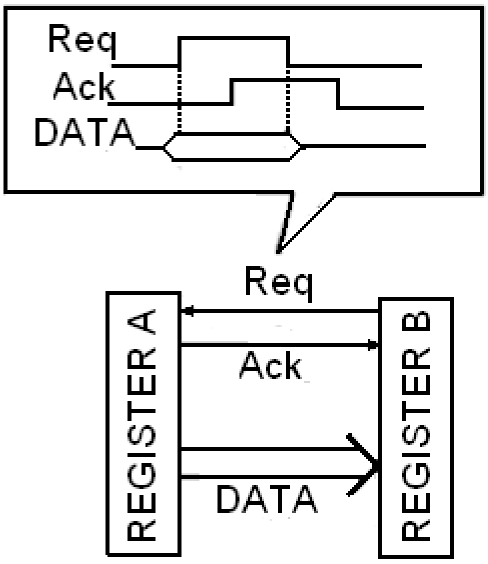


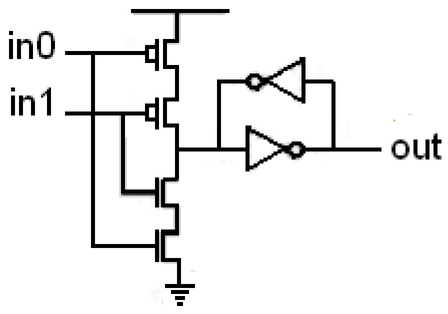
Figure 1. Handshake principle illustration

Register A, when it has data to transmit to register B, activates request signal. After that, Register B, save this data and answer with acknowledge signal. Next data transfer will start with request signal again.

C-element

Let's see how handshaking can be realized in schematic. There are a set of schematics methods to realize handshake protocol. All of them use C-element – the main element of asynchronous schematics.

The first circuit of C-element was invented by Muller and it is called Muller element (fig 2 (a)). This is not the best implementation of C-element. But this circuit is good to illustrate the C-element structure.



a	b	y (n)
0	*0*	*0*
0	*1*	y (n-1)
1	*0*	y (n-1)
1	*1*	*1*

(a) *(b)*

Figure 2. Circuit (a) and truth table (b) for C-element

The truth table for this element is simple (fig 2 (b)). When all of these inputs are logic 1 its outputs turns to 1 state too. When all inputs are logic zero out turns to zero state too. Otherwise, element saves its previous state.

4-phase handshake protocol

The most commonly used handshaking protocol is 4-phase bundled-data protocol.

The whole transmit or processing data cycle consists of 4 phases. For example, circuit at figure 3 is in the state of rest at a start moment. At first, the sender (first cascade R1), when it has data to transmit, sets request signal high with the data. C-element of receiver (second cascade R2) has a high inverted acknowledgement signal on its C-element. High level of acknowledgement and high level of request will lead to high level out of C-element. This is enable signal for latching data and acknowledgement signal for previous cascade. So, receiver latches the data and sets acknowledge high. First cascades input request signal is in low state by this time. So low input request and low inverted acknowledge from the next stage makes output of first cascades C-element low.

The last step of 4-phase transmit handshake cycle is taking acknowledge low by receiver. At this point the next communication sequence may be initiated. The control part of this circuit is called Muller pipeline.

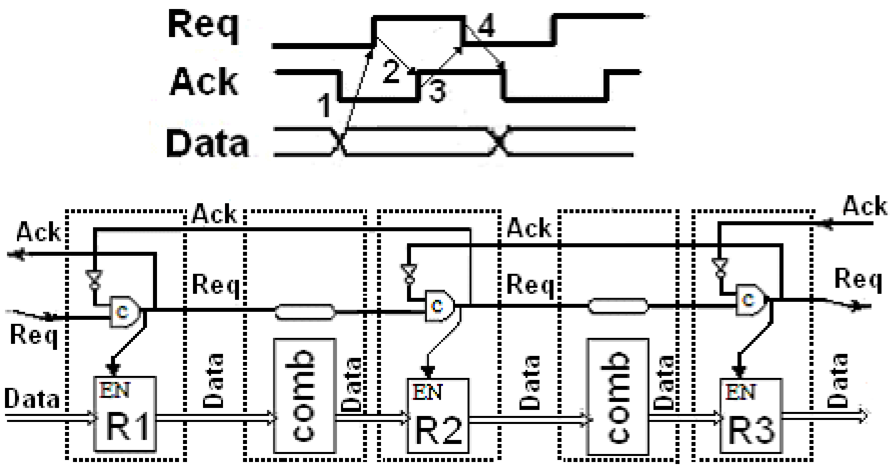


Figure 3. 4-phase handshake protocol

Noticed, that delay of request signal between registers has to be more than combination logic delay. Otherwise, request will be seen earlier than valid data, and wrong data will be latched.

The main disadvantage of such circuit is four C-element delays, which are needed for complete protocol cycle.

2-phase bundled-data protocol

In the general case, this type of protocols is faster then 4-phase. Transmitter sets request signal with the data. When acknowledgement signal changes its state, data is latched. The whole protocol circle is done. Another circle starts with the next changing state of request signal.

As the backbone control circuit, we can also use Muller pipeline in a 2-phase bundled-data pipeline. But we have to use special type of triggers to latch data. This element is like a multiplexer with two control inputs C and P. It should capture data only when its inputs aren't equal. Otherwise, it saves its previous state and lets input data become valid. So, at the first moment C and P are "00". Request signal makes C-element output high, and CP equal to "10". Element latches input data.

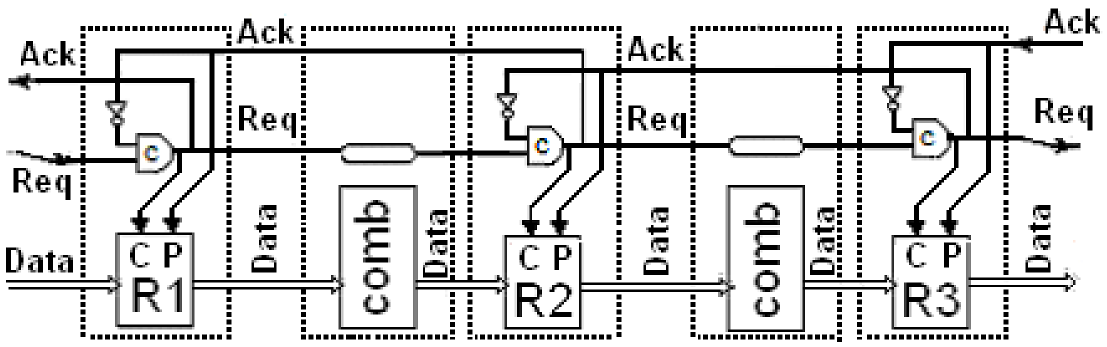


Figure 4. 2-phase handshake protocol

Ideally, the 2-phase bundled-data protocol should be faster than the 4-phase, but result depend on a particular design. A 2-phase bundled-data circuits are more complex and takes more area than 4-phase.

Moreover, there are delays in a request chains too. Designing such a circuit, it is necessary to calculate combination path and implement special buffers, whose delays are bigger than combinational path delay.

Dual-rail handshake protocols

Another way to organize communication between memory elements is mix request signals in data paths. For example, we need to transmit 10 data bits. Every bit implies information: is it ready or not? Acknowledgment signal is one for all data courses. It sets if all data bits are ready. Data bits can't change if acknowledgement is in ready.

We can implement extra information for each logic bit of data if we have two wires for each logic data bit. The state 00 means that data isn't ready, 01 is a logic zero, 10 is a logic one, 11 is a forbidden state.

Figure 5 illustrates FIFO circuit based on a dual-rail 4-phase interface. So, we can see a set of data/request signals, and one acknowledgement for stage. But our task is more complex if we have to process dual-rail encoded data bits. Let`s, for example, implement AND function between the first and the second dual-rail logic bits.

Figure 5 depict the truth table for dual-wire and element. The state 11 is a forbidden state for inputs and 11 strings were deleted.

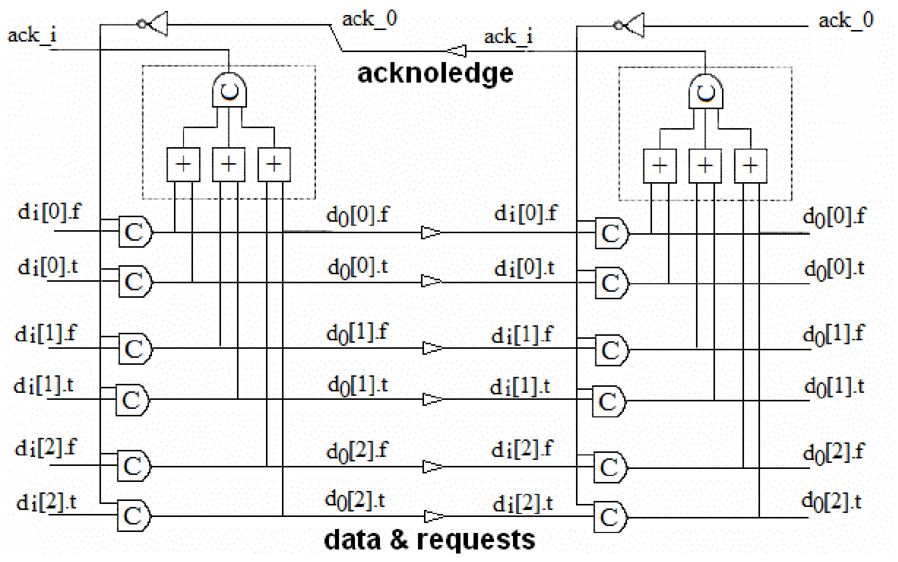


Figure 4. Dual rail pipeline

Output of the circuit will be non-zero in case, if both of the inputs are non-zero state. So, we can write equation for all outputs and construct a dual-rail AND circuit.

a.f a.t	b.f b.t	y.f y.t
00	00	00
00	01	00
00	10	00
01	00	00
01	01	01
01	10	10
10	00	00
10	01	10
10	10	10

y.f = a.t & b.t |
a.t & b.f |
a.f & b.t

y.t = a.f & b.f

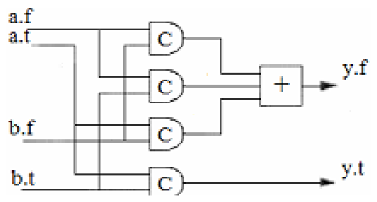


Figure 5. Dual rail AND element constructing

We can see implementation of our AND element on figure 6. So, the second cascade has two inputs. The first is AND function of previous cascade outputs. And the second input remains a simple pipeline wire without data processing.

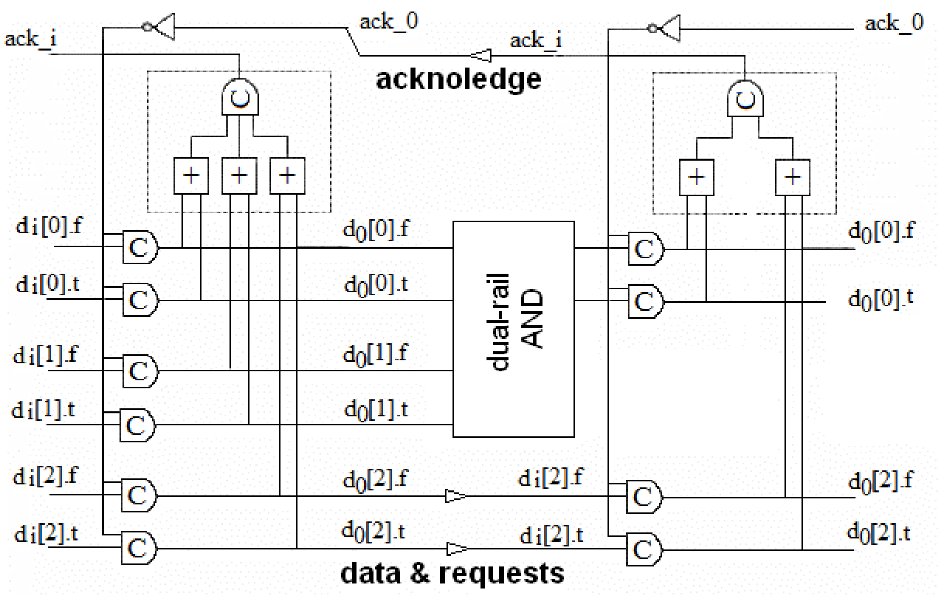


Figure 6. Dual rail AND element implementation

This type of circuits doesn't need delay implementation in the requests chains. So no matter to take into account delays for data processing.

Synchronous and asynchronous design compare

We designed two circuits of parallel multiplier. One is synchronous and another is 4-phase bundled-data self-timed circuit (fig. 7).

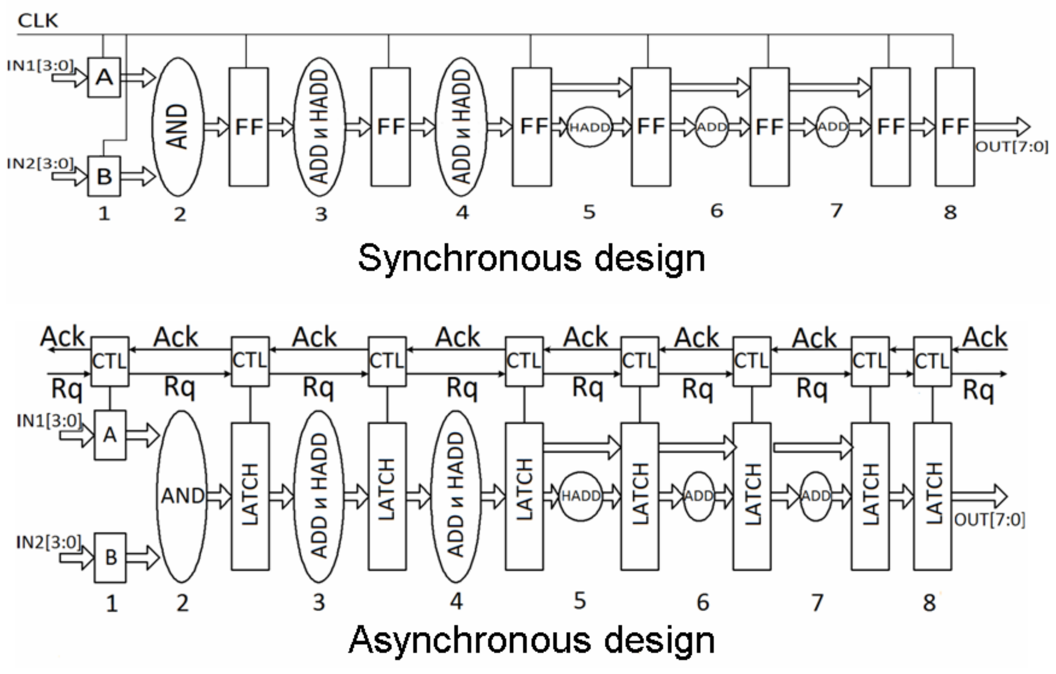


Figure 7. Synchronous and asynchronous multiplier design

4-bits multiplicand and multiplier are inputs of both circuits. Every stage of circuits calculates particular sum of product. 7-bits product of multiplication is their outputs.

We have modeled both of them using cadence design system. And we made up a comparative table by the testing results (table 1).

	Synchronous design		Asynchronous design	
	single	stream	single	stream
Period (equivalent for asynchronous) (ns)	3,66	3,66	2,8	3,4
Max temp (C)	66		>200	
Avr. current (mA)	12,02	20	0,29	14
Max current (mA)	18,9	65.12	5,95	30.39

Table 1. Synchronous and asynchronous design comparative table

The first column contains results for self-timed circuit, and the second is for synchronous. Every column has two sub-columns. The first sub-columns, titled single, represents results for ones calculation. It means that one set of inputs was calculated and every time only one stage of circuits was busy. The second sub-columns, called stream, contains stream calculation tests results. In this case, calculations were performed in continuous mode, and all stages at that moment were busy.

So, we can see, temperature stability of asynchronous circuit is better. Furthermore, asynchronous circuit is faster in the first case. Its operating speed is determined by actual local latencies. But in case of stream calculation performance of both circuits types are equal.

Power consumption is much better in a case of single calculation and about 30 percent better in a continuous mode.

.