Design and implementation of an Asynchronous Controller for FPGA Based Asynchronous Systems

T.N.Prabakar
Saranathan College of Engineering,
Tiruchirappalli
Tamilnadu, INDIA

G. Lakshminarayanan
National institute of Technology,
Tiruchirappalli
Tamilnadu, INDIA

K.K.Anilkumar
Bahrain Technical University
Bahrain

ABSTRACT
In a clause of combinational circuits, the throughput can be increased, without (wave) pipelining, by introducing data dependent delay feature thus avoiding the worst case delay. That is, in circuits like multipliers and adders which are the basic building blocks of any DSP system; the processing delay can be varied according to the magnitude of the input data. This makes the circuit asynchronous and necessitates a controller to arbitrate the data. Systems like FIR filters, where a series of combinational multipliers are used, can be asynchronously pipelined with a controller regulating the data between stages. With this system level pipelining, speed of circuit level pipelining can be achieved provided the data are of low magnitude. In this paper, two controller architectures are presented to regulate the data flow between asynchronously pipelined stages. Firstly, as a stepping stone, Altera's soft-core NIOS processor [1] is used and secondly, an exclusive asynchronous controller is designed using HDL. These controllers are designed to suit asynchronous implementation in conventional FPGAs, to effectively handle repeated data and to perform self-test. These controllers issue the control signals to the various dual edge triggered pipelined registers to process the data in both the edges for further improving speed. In the HDL version of the controller, programmable delays are generated by a 'logic locked' high frequency counter without using delay elements. To verify the efficacy of these controllers 2 tap FIR filter is implemented using Braun array multipliers and adders. Thus, this approach consumes lower power and achieves data dependent throughput and also avoids the need for global clock signals and skew problems.

Categories and Subject Descriptors
B.7.1 [**Hardware**]: Integrated Circuits - VLSI

General Terms
Design

Keywords
Asynchronous, FPGA, Low power, Asynchronous Controller, Data Dependent Delay.

1. INTRODUCTION
The complexity in distributing a high speed accurate clock over a large circuit area without skew as well as the inherent high power consumption caused by the clock network has prompted designers to reconsider the role of asynchronous circuits [2]. A synchronous combinational circuit works on worst case delay while an asynchronous combinational circuit works on average case delay. But, when a synchronous or asynchronous system is pipelined, both should work theoretically with the same worst case delay, even though the hand shaking signals in asynchronous system contribute some excess delay. In synchronous systems the clock frequency governs the speed of the whole circuit, which is determined by its worse case delay. On the other hand, asynchronous circuits rely on the use of completion-detection methods to determine when a combinational logic block has completed its operation. Several methods of completion detection are available in literature. Asynchronous systems, based on dual rail coding techniques carry the disadvantages of a very high hardware overhead coupled with low operation speed while the systems based on bundled delay approach fails to exploit the data dependency of internal delays. In addition, conventional FPGAs are suitable only for synchronous implementations and lack of proper CAD tools negate the use of asynchronous systems. In this paper, two controller architectures are presented to subdue the challenges of using conventional FPGAs for asynchronous applications. With a controller in the system, the following advantages are realized:

i. When same data is repeatedly coming, the present output is again latched at the output and immediately the next data is read. This avoids the repetition of same processing.
ii. Controller is sensitive to both the edges and return to zero transitions is avoided.
iii. Duplicating delay logic is removed by a 'logic locked' counter structure with a delay resolution of 4 ns.
iv. Fine tuning delay, prohibits glitches passing between stages. Hence, the reduction in dynamic power consumption is achieved.
v. The controller is a general purpose and can be used for any application and any number of stages.
vi. The controller is suitable to implement asynchronous systems in currently available synchronous FPGAs.
vii. The controller has the built-in self test feature.
viii. Problems with synchronous circuits like clock skew are avoided.
ix. If the circuit is not under the clause of data dependent delay, setting all the delays as same, will give a simplest asynchronous pipelined architecture.
x. In addition to the above, the controller has the benefit of all the advantages offered by the asynchronous systems.

2. BRAUN ARRAY MULTIPLIER WITH DATA DEPENDENT DELAY

In DSP operations like correlation, convolution, and filter banks for multirate signal processing, multipliers are being used as a foundation blocks [3]. Out of various algorithms available for multipliers, Braun array multiplication algorithm is chosen for implementation to realize data dependent delay feature. The block diagram of an 8×8 pipelined Braun array multiplier is given in Fig.1.

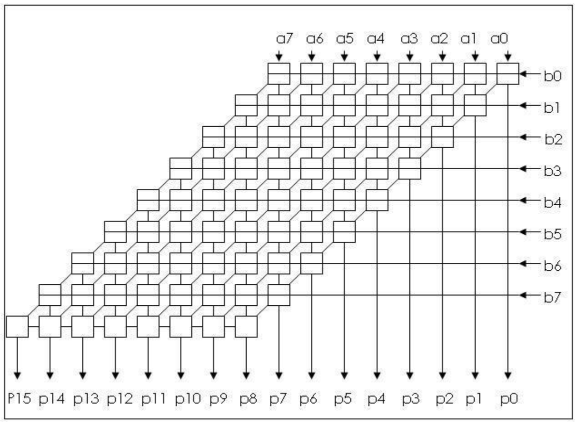


Fig 1. 8×8 pipelined Braun Array Multiplier

The process delay estimation is performed by finding the magnitude of the inputs. Referring to the 8×8 Braun array multiplier in Fig.1, if the inputs are “00000010” and “00000010” the output is “0000000000000100”. In this case, bits up to P2 only are used. In that case, the delay is less. When the input data has all the 8 bits, then comparatively higher delay is needed to receive all bits up to P15. So, the position of first ‘1’ from the MSB in both the operands decides how many bits are in the output and the processing delay. The following Table.1 depicts this relation. In Fig 2.a the magnitude is low and also varying and hence, the throughput is higher where as in fig 2.b the magnitude is larger and fixed and hence throughput is lower. Hence, with a controller in the system, the dynamic data dependent delay control can be achieved. Similar works are referred in [6] and [7].

Table 1. Data Magnitude and No. of stages used with Prefix result

S.No	Data A (8Bits)	Data B (8Bits)	Prefix	Result	Stages Used
			(16 Bits)		
1	0000	xxxx	0000 0000		0
	0000	xxxx	0000 0000		
2	0000 0001	0000 0001	0000 0000 0000 000	1	1
3	0000 001x	0000 01xx	0000 0000 000	1 xxxx	4
4	0000 1xxx	0000 1xxx	0000 0000	1xxx xxxx	7
5	1xxx xxxx	1xxx xxxx		1xxx xxxx xxxx xxxx	8

3. INTASYCON – I BASED ASYNCHRONOUS PIPELINED SYSTEM

The proposed asynchronous controller is named as “INTASYCON” (stands for INTelligent ASYnchronous CONtroller) which is an HDL based hardware module used to control the asynchronous data flow inside an asynchronous circuit [4] and [5]. The data manipulations inside the controller, as shown in Fig. 2, are described below:

1. The controller has an in-built free running counter which increments every 4 ns.
2. The source and destination of data are assumed to memories. After the first start signal, the controller itself fetches the new data from the memory. The controller receives the data and verifies it is not a repeating data. If it is a new data, the controller fetches the current value in the counter.
3. The magnitude of the data is analyzed and for every stage, the delay counts are selected based on the input magnitude.
4. The delay values are added with the initial count value so that the completion time of all the stages can be estimated.
5. A critical case happens when the previous data consumes more time and the current data consumes less time. In that case, the current data has to wait in the n^{th} stage till the previous data gets manipulated in the $n+1^{th}$ stage. In conventional asynchronous circuits, this is taken care by the Muller-C element and acknowledgement signal. Here, since the controller dictates the end point of stages it compares the current data’s end point (of n^{th} stage) and previous data’s end point (of $n+1^{th}$ stage) and which is greater is chosen as the current data’s endpoint (of n^{th} stage).
6. The process continues till the data ceases.

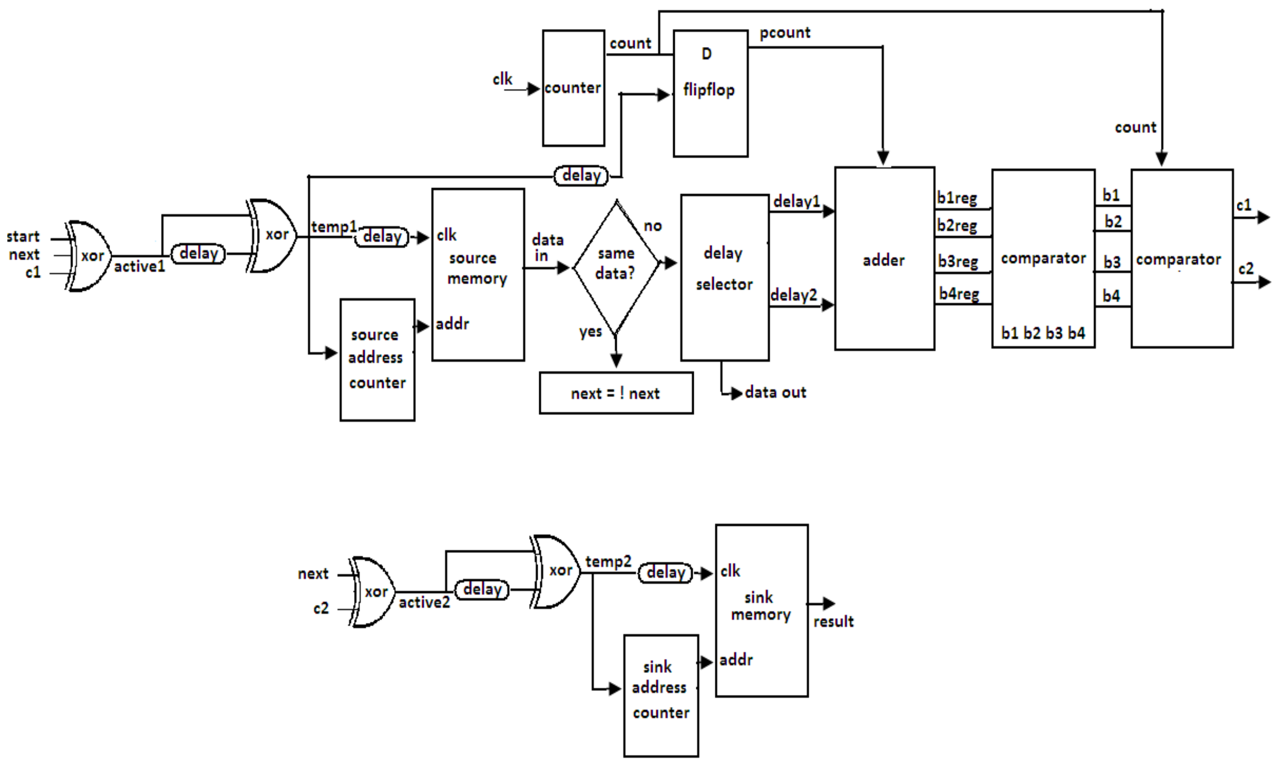


Fig. 2 INTASYCON with source and sink memory

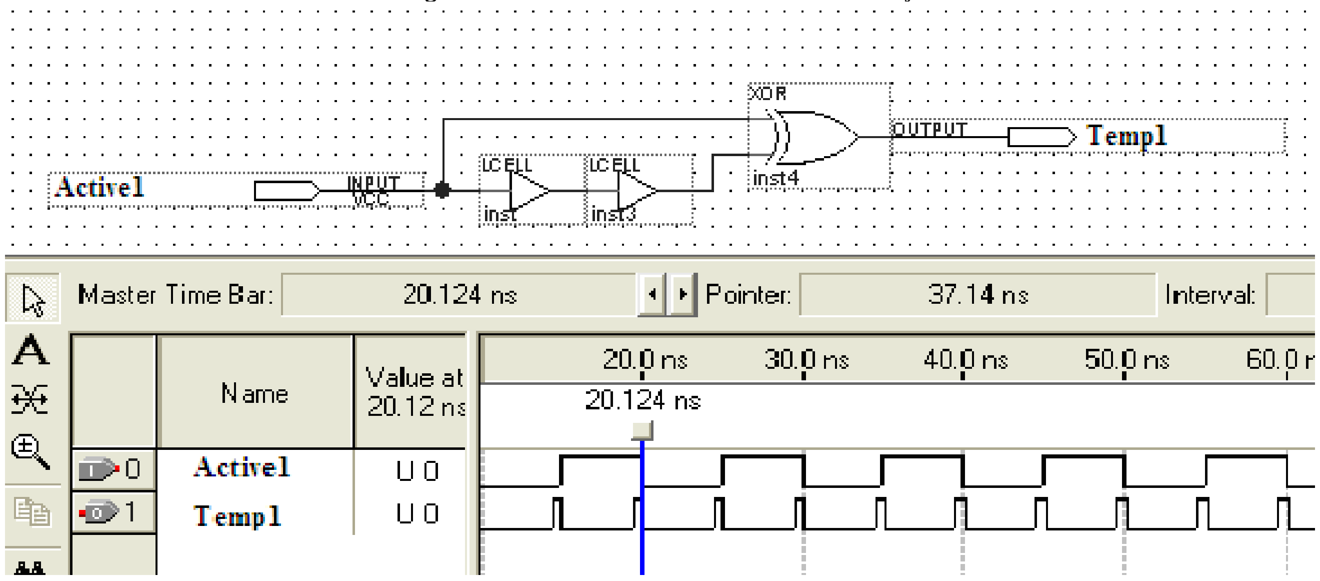


Fig. 3.a & 3.b Dual Edge to Positive Edge conversion (Used for using Conventional Memory with dual rail Asynchronous process)

3.1 Circuit Description

1. The process is started by a leading edge transition of ‘start’. Since ‘next’ and ‘c1’ are at 0 states, positive edge in ‘start’ will drive ‘active’ to toggle its state.

2. ‘Active1’ is dual edge sensitive, but conventional memories are positive edge triggered, hence a delay with XOR gate converts dual edge sensitive into positive edge sensitive as in fig. 3.a & 3.b. ‘Temp1’ is positive edge sensitive signal and given as the clock after delay to source memory where input data are stored.

The same is connected to an address counter to generate addresses.

3. The data in releases the data which is compared with the previous data. If it is a repeating data, 'next' is toggled which is connected to xor gates as shown in the fig.6. and is used to trigger both memories. In the sink memory, the previous output is latched again and in source memory next data will be released because of this toggle in 'next'.
4. If the data is new, the delay selector, depending upon the magnitude of the data gives out delays for various stages. In this Fig. 2. it is for two stages. And the data has given out at 'dataout'. The data processing need not wait for this control manipulations. The data will get simultaneously processed in the stage-1.
5. 'Temp1' after suitable delay is used to pick the current value from the free running counter. This 'pcount' is the counter value when the data enters into the process.
6. Since, there are two stages, two data will be available in them and therefore 4 delays are available at any time. Hence, signals b1reg (=pcount+delay1) and b2reg (=pcount+delay2) are completion of data1 and signals b3reg and b4reg are completion counts of data2.
7. Assume data2 is under process and first stage will over at b3reg. The second stage cannot be opened if previous data1 has not gone out of stage2. Hence, this b3reg is compared with b2 which gives second stage completion of data1.
8. Likewise, b1, b2, b3 and b4 are the obtained after comparison.
9. These values are compared with present counter value and when counter reaches the end point calculated, c1 and c2 toggles.
10. c1 indicates the completion of stage1 and hence next data can be taken from memory. So, c1 is also connected to input XOR gate.
11. c2 is used to trigger the output memory so that the output from stage2 can be properly latched.
12. c1 and c2 are also dual edge sensitive signals and hence, dual edge triggered flip-flops are used to transmit data between stages.

With this type of controller in the system, the data dependent delay is used but for every data two comparison and two addition operations are needed. This also consumes time. The initial count is taken when the data enters but for determining the final count which will not collide with previous data, these operations are necessary. To reduce this complexity, these values are readily stored in a memory and this architecture is presented in the next section as INTASYCON – II.

4. INTASYCON – II BASED ASYNCHRONOUS PIPELINED SYSTEM

In order to avoid the computation time of the controller, the Second variation in the INTASYCON – II, as shown in Fig. 4, uses minimum computation to find out the end points since all the data are stored in different memories. The data manipulations inside the controller are described below:

1. The controller has an in-built free running 6 bit counter which counts every 4 ns.
2. The source and destination of data are assumed to memories. After the first start signal, the controller itself fetches the new data from the memory. The controller receives the data and verifies it is not a repeating data. If it is a new data, the controller fetches the current value in the counter.
3. The magnitude of the data is analyzed and graded as low, medium or high.
4. As a two stage case two data will be available in two stages. A critical case happens when the previous data consumes more time and the current data consumes less time. In that case, the current data has to wait in the nth stage till the previous data gets manipulated in the n+1th stage. In conventional asynchronous circuits, this is taken care by the Muller-C element and acknowledgement signal. Here, since the controller dictates the end point of stages it compares the current data's end point (of nth stage) and previous data's end point (of n+1th stage) and which is greater is chosen as the current data's endpoint (of nth stage).
5. Low, Medium and High memories are used to store this data. When previous data is of high and current data is medium or when previous data is medium and current data is low, then in order to avoid collision HIGH1 memory is used. Similarly when the previous data is high and current data is low then HIGH2 memory is used.
6. To resolve this issue, different memories are used which dictates the delay values as shown in the following table 2.

Table 2. Data Magnitude with Delay & Memory Selection

Magnitude of Previous Data	**Magnitude of Current data**	**Collision**	**Stage -1 Delay**	**Stage -2 Delay**	**Memory Used**
Low	Low	No	Low	Low	LOW
Low	Medium	No	Medium	Medium	MEDIUM
Low	High	No	High	High	HIGH
Medium	Low	Yes	High	Low	HIGH1
Medium	Medium	No	Medium	Medium	MEDIUM
Medium	High	No	High	High	HIGH
High	Low	Yes	High	Low	HIGH2
High	Medium	Yes	High	Medium	HIGH1
High	High	No	High	High	HIGH

7. As a case study, 8×8 Braun array multiplier is considered. The combinational implementation takes 28.414 ns as critical path on a Cyclone II FPGA. The input data range is divided into low, medium and high as follows:
    a. Low – When the result is within 8 bits
    b. Medium – When the result is within 12 bits
    c. High – When the result is all 16 bits
8. The corresponding delays are assigned as 15, 24 and 30 ns. When the counter runs at 3 ns, the count values for these delays will be 5, 8 and 10. If the data of 10 (Low range data) enters

when the count is equal to 50, the first stage is over by 55[initial (50) +delay (5)] and second stage will be over by 60.

9. Since, the counter used is a 6 bit counter, the data range is from 0 to 63. Hence, for address=50, the data should be 55 and 60.

pcount (Initial)	Low		Med		High		High 1		High 2	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
0	5	10	8	16	10	20	10	15	10	18
1	6	11	9	17	11	21	11	16	11	19
2	7	12	10	18	12	22	12	17	12	20
Continues…										

The following table 3 gives the memory contents.

Table 3. Memory Contents of all memories (pcount indicates the count when data enters and c1, c2 indicate the process completion depending on data magnitude)

10. When the present value of the counter reaches 25 and 30, the first and second DETFF are opened to allow the data to the second stage and to output respectively.
11. The process continues till the data ceases.

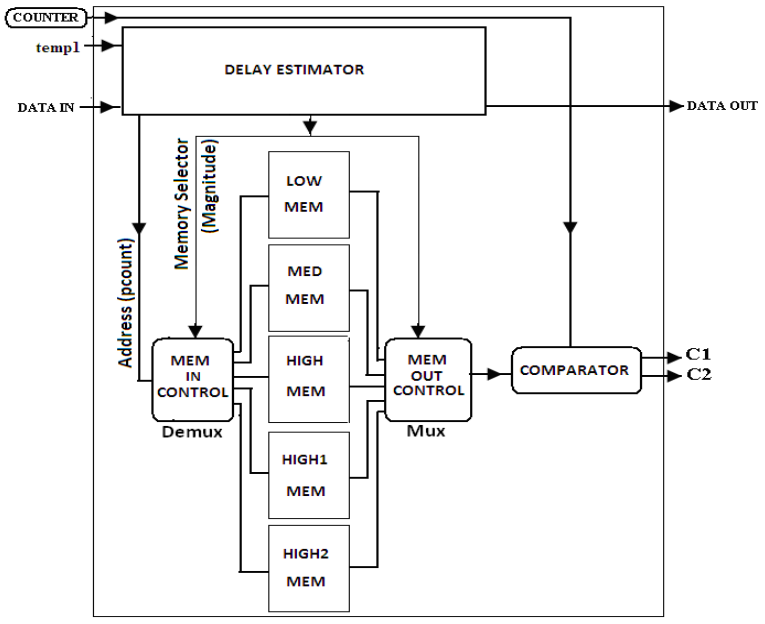
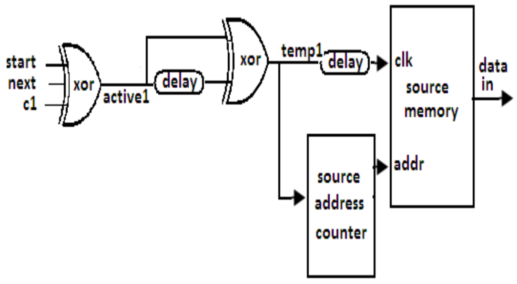


Fig. 4a. Architecture of INTASYCON-II



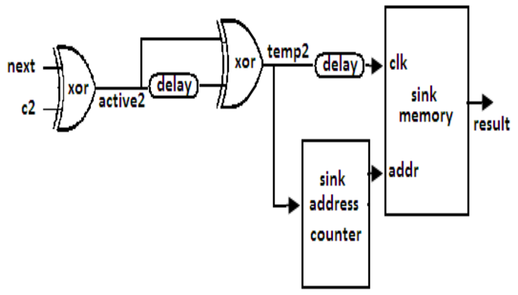


Fig. 4b. Data input section (fetching data from source memory)

Fig. 4c. Data Output Section (storing result in sink memory)

4.1 Circuit Description

1. The process is started by a leading edge transition of ‘start’. Since ‘next’ and ‘c1’ are at 0 states, positive edge in ‘start’ will drive ‘active’ to toggle its state as shown in fig. 4b.
2. ‘Active1’ is dual edge sensitive, but conventional memories are positive edge triggered, hence a delay with XOR gate converts dual edge sensitive into positive edge sensitive. ‘Temp1’ is positive edge sensitive signal and given as the clock after delay to source memory where input data are stored. The same is connected to an address counter to generate addresses.
3. The data in releases the data which is compared with the previous data. If it is a repeating data, ‘next’ is toggled which is connected to XOR gates and is used to trigger both memories. In the sink memory, the previous output is latch again and in source memory next data will be released because of this toggle in next.
4. If the data is new, the delay selector, depending upon the magnitude of the data gives out delays for various stages. In this figure it is for two stages. And the data has given out at ‘dataout’. The data processing need not wait for this control manipulations. The data will get simultaneously processed in the stage-1 as shown in fig. 2b.

1. ‘Temp1’ after suitable delay is used to pick the current value from the free running counter. This ‘pcount’ is the counter value when the data enters into the process.
2. This ‘pcount’ is used as the address to fetch the end points from any one of the 5 memories as shown in fig. 2c. This is a straight forward operation and consumes no time. Both c1 and c2 end points are stored in an appended fashion (like 5560, referring to the previous example in III.). The separated memory output will be end points and these values are compared with present counter value and when counter touches the end point calculated here, c1 and c2 toggles.
3. c1 indicates the completion of stage1 and hence next data can be taken from memory. So, c1 is also connected to input XOR gate.
4. c2 is used to trigger the output memory so that the output from stage2 can be properly latched as shown in fig. 4c.
5. c1 and c2 are also dual edge sensitive signals hence, dual edge triggered flip-flops are used to transmit data between stages.

5. INTASYCON BASED 2 TAP 16 BIT FILTER

The Fig.5 shown is INTASYCON based FIR filters in which the three stages are given as two multipliers and one adder. The Braun array multiplier with data dependent delay is placed in both multiplier sections. For the 16 bit adder, a series of carry ripple adder is designed so that depending upon the magnitude the data can be taken out at any time. After preprocessing of data, the data is exposed to the first stage (× b0). Based on the data magnitude, C1, C2, and C3 are issued. C1, C2 and C3 operate DETFFs to pass the data between stages. The completion times of all the stages for a particular data are known and hence, the controller verifies whether the previous data has come out from a particular stage before the next data is presented.

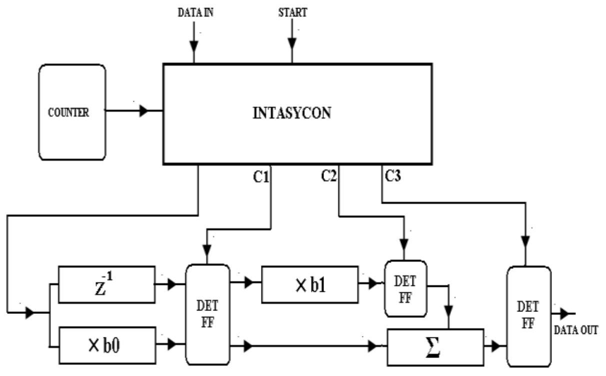


Fig. 5. INTASYCON Based 2 Tap FIR Filter

6. IMPLEMENTATION RESULTS

The system is implemented on a Altera Cyclone II FPGA and the results are given in Table 4.

Table 4. Comparison of Schemes

S No	Implementation Type	Comb Reg-isters	Logic Reg-isters	Memory	Time to Process
1	Synchronous Pipelined	670	56	2048	5.8 μs
2	Conventional Asynchronous Pipelined	673	40	2048	8.8 μs
3	INTASYCON – I	938	209	2048	5.3 μs
4	INTASYCON- - II	855	185	3648	5.1 μs

The synchronous pipelined system is implemented as per the conventional scheme. The asynchronous pipelined system is implemented using Bundled data protocol scheme. In bundled data protocol, two problems are to be addressed. One is the accumulation of delay in the passage of acknowledgement signals from final stage to first stage and the other is bundled time constraint, that is designed delay should exactly match the delay of the combinational circuit. The rows 3 and 4 of the table give the implementation results using INTASYCON I and II respectively. In all the cases, a data set of 0 to 255 numbers with 20% repeating data is taken as the test data. To process this test data, the time consumed is given in the last column.

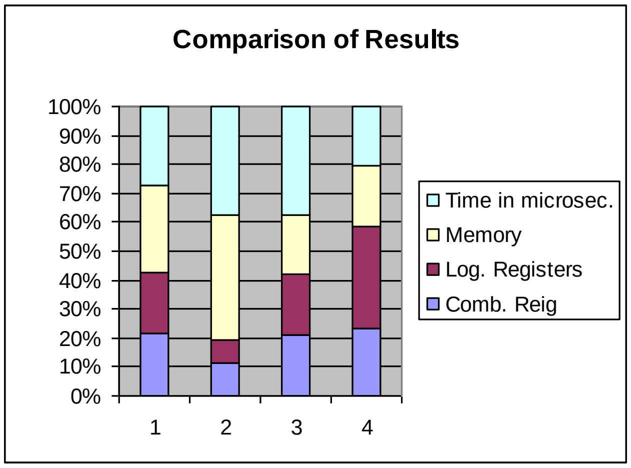


Fig. 6. Normalized comparison results

From the results, as seen in the normalized form, the INTASYCON based system operates faster than synchronous pipelined system because for all data, worst case delay is not considered. Hence, with system level pipelining, circuit level pipelining can be achieved. The memory is used to store input and output data. INTASYCON – II needs more memory to store the delay values.

7. CONCLUSION

Asynchronous circuits have many potential advantages over their synchronous equivalents including lower latency, lower power consumption, less noise, design reuse and lower electromagnetic interference (EMI). In the INTASYCON system, the addition of controller a) increases the area because of controller architecture but compensates by removing the pipelining registers, b) reduces speed when compared to a pipelined architecture but compensates by providing data dependent delay.

The control signals generated by these controllers avoid a global synchronizing signal used in synchronous systems. The data dependent delay assures the delay to be varied according to the data and hence, the throughput of the system is further increased.

The overhead is comparably higher for a 2 Tap FIR filter but, the same INTASYCON controller can be used for even higher order circuits with minimum modification. Hence, as the complexity of the application increases, the overhead due to INTASYCON is acceptable.

8. REFERENCES

[1] “Nios II Processor Reference Handbook,” 2008. Altera Corporation, ver 8.0, May.

[2] Jens Sparsø, Steve Furber, “Principles of asynchronous circuit design”, Kluwer academic Publishers, 2001.

[3] K.K. Parhi, “VLSI Digital Signal Processing Systems”, Wiley Interscience

[4] T.N.Prabakar, G. Lakshminarayanan, Dr. K.K.Anilkumar, “SOPC based asynchronous pipelined DCT with self test capability”, IEEE International Conference on Microelectronics, Egypt, 2007.

[5] T.N.Prabakar, G. Lakshminarayanan, Dr. K.K.Anilkumar, “Asynchronous Pipelined Multiplier with Intelligent Delay Controller”, IEEE International Conference on System on Chip Design, Korea, 2008.

[6] S.M.Nowick, “Design of a low-latency asynchronous adder using speculative completion”, IEE Proc. –Comput. Digit. Tech., Vol. 143, No. 5, September, 1996.

[7] S.M.Nowick, Kenneth Y. Yun, Peter A.Beerel, Ayoob E.Dooply, “Speculative Completion for the Design of High-Performance Asynchronous Dynamic Adders” pp 210-223,1997.