行政院國家科學委員會專題研究計畫 成果報告

非同步 8051 處理器之研究與設計
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 94-2213-E-009-137-
執 行 期 間 ：94 年 08 月 01 日至 95 年 09 月 30 日
執 行 單 位 ：國立交通大學資訊工程學系(所)

計 畫 主 持 人：陳昌居

計畫參與人員：博士班研究生-兼任助理：鄭緯民
碩士班研究生-兼任助理：王端傑、蔡瑞夫

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 96 年 11 月 05 日

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告
□期中進度報告

非同步 8051 處器器之研究與設計

計畫類別： 個別型計畫 94-2213-E-009 -137
計畫編號：NSC 94－2213－E－009－137－
執行期間：94 年 08 月 01 日至 95 年 07 月 31 日

計畫主持人：陳昌居
共同主持人：
計畫參與人員：鄭緯民、王端傑、蔡瑞夫

成果報告類型(依經費核定清單規定繳交)：■精簡報告 □完整報告

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，■一年□二年後可公開查詢

執行單位：交通大學資訊工程系

中 華 民 國 96 年 11 月 05 日

中英文摘要：（低耗電 low power、非同步電路 asynchronous circuit、8051 微控制器 8051 microcontroller、Balsa 語言、FPGA）

近來可攜式裝置的使用越來越普遍，因此低耗電的設計成為重要的目標，由於資料驅動的特性使得非同步電路適用於低耗電設計，我們會提出一個新的非同步 8051 微控制器的解碼器設計，這是由於 8051 是最普遍使用的解碼器之一，而且往往在其應用上低耗電特性是相當重要的。

在本論文中電路設計使用 Balsa 語言，一種以 CSP（Communication Sequential Process）為基礎的非同步電路硬體描述語言並且可以合成非同步電路，由 Balsa 可以合成適用於 Xilinx 合成器的 Verilog netlist，我們可以比較非同步與同步電路在 Xilinx FPGA 上的表現或使用其它 CAD 工具來實現晶片設計。

Recently mobile devices have been popularly used, and low power is becoming an import subject. With the data-driven feature, the asynchronous circuit is suited to be used for low-power design. We will propose a new decoder design of the asynchronous 8051 microcontroller because the 8051 is one of the most popular microcontroller and is often used in applications where low energy consumption is important.

The circuit is a complied VLSI-program, using Balsa as VLSI-programming language which is a CSP-based asynchronous hardware description language and synthesis tool. A Verilog netlist for XST (XILINX Synthesis Tool) is generated by Balsa. We will compare asynchronous 8051 and synchronous 8051 in XILINX FPGA and then use Cadence tools and Synopsys tools to synthesis the layout of the circuit.

報告內容：

Research in asynchronous circuit design can be traced back to the mid 1950s, however, because of testability and easy to design issues, synchronous design becomes the major technology of digital circuit design. However, in the late 1990s projects in academia and industry demonstrated that it is possible to design asynchronous circuits which exhibit significant benefits in nontrivial real-life examples, and therefore commercialization of the technology began to take place.

We hope to design a new decoder with low power and high performance features, and thus the asynchronous design technology is the one we chose for this purpose.

The architecture of the pipelined asynchronous 8051

The architecture of asynchronous pipelined 8051 is show in figure 1

There are five stages of our pipeline, and an interface between the processor core and the RAM. The IF (instruction fetch) stage fetches instructions from ROM. The ID (instruction decode) stage decodes the instruction and handles the branch instruction. The OF (operand fetch) stage fetches operands from RAM. The EXE (execute) stage execute instructions according to opcodes of instructions. The WB (write back) stage write back the result into RAM.

There are three basic components in the IF stage- mem interface, buffer, and fetcher ctrl. Mem interface is a arbitrator to arbitrate requests from one of the buffers. Buffers are controlled by fetcher ctrl. According to the control signal, buffers prefetch instructions from the external ROM or provide the target byte which fetcher ctrl needs. Fetcher ctrl receives the value of the program counter, and check if it is hit in one of the buffers or miss. If it is hit, fetcher ctrl sends a request of read to the hit buffer, and if it is miss, fetcher ctrl sends a request of prefetch to all buffers. In addition, if the buffer is read the last byte, fetcher ctrl would send a request of prefetch.

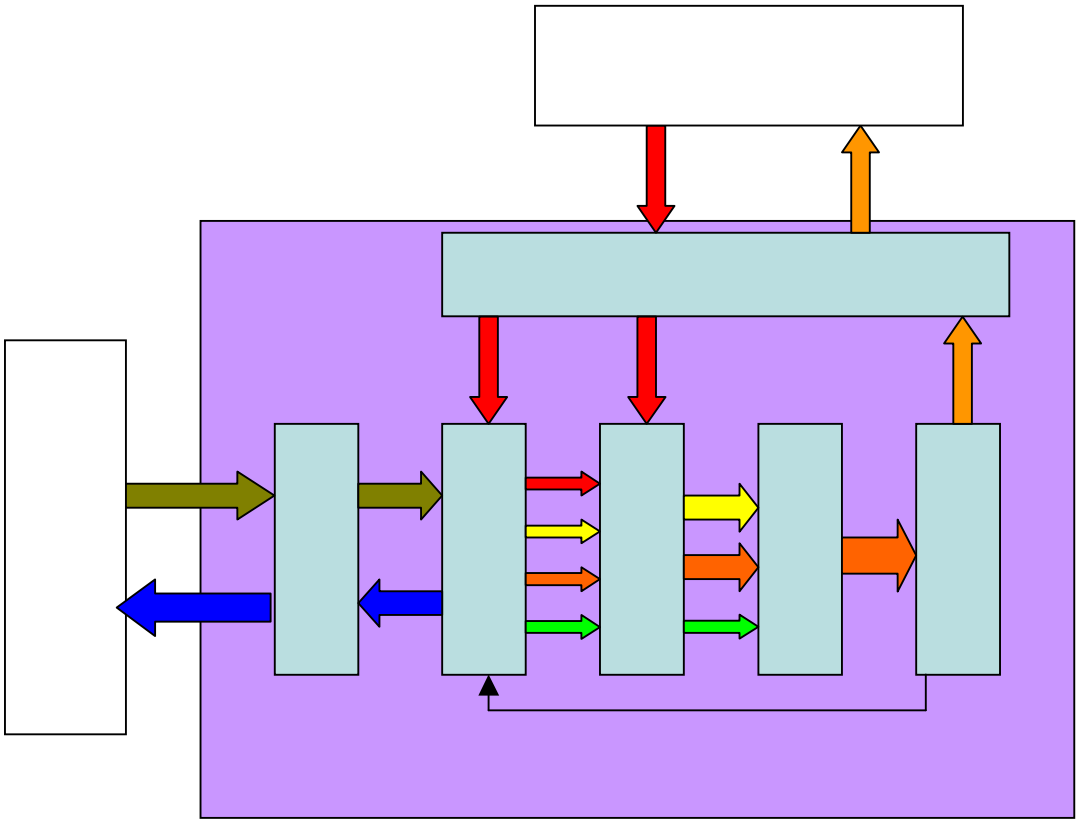

Figure 1 The architecture of asynchronous pipelined 8051

In asynchronous systems, there exists a problem that the inputs of a merge circuit may come simultaneously. In Balsa, we can use an arbitrator component to solve this problem. The *mem interface* may receive a address for one of the buffers and access the ROM according to this address. However, it would happen that both buffers send addresses to the *mem interface* and the circuit might be error. Thus, we use the “arbitrate description” to resolve this problem.

For each buffer, it receives the target address and the action signal from *fetcher ctrl*. If the

action signal is read, it returns the target byte according to the address. If the action signal is write, it fetches 32 byte data which start from the address.

Fetcher ctrl controls all the buffers. It fetches the value of the program counter first. Then it checks if the target byte exists in one of the buffers. If there is a buffer which has the byte, Fetcher ctrl sends a read request and the address to the buffer and then passes the target byte to the ID stage. If the target byte is the last byte of the buffer, fetcher ctrl will send a write request to the buffer. However, if no buffer has the target byte, fetcher ctrl will flush all the buffers.

The ID stage is divided into ID1 and ID2 two stages. In the ID1, it fetches the first byte of an instruction, decodes this instruction, determine the remained bytes, abstract the opcode, and generate the control signal of this instruction. In the ID2, it would fetch remained bytes and provide completed control signals for the OF stage. If the current instruction is a branch instruction, the ID2 stage would calculate the target address and handle the branch action.

When ID1 receives the instruction byte, it would determine that this instruction is regular or non-regular. This could decrease the size of the multiplexer. Then, according to the instruction, ID1 generates the signals needed by the following stages such as the remained bytes, the opcode, the read signal, and the write signal. In order to decrease the area cost, we use the shared procedure in Balsa, which would construct only one component whatever times this procedure is called.

In the ID2 stage, it will fetch the remained bytes first. To avoid the race condition between the ID1 stage and ID2 stage, we use the “handshake enclosure” description in Balsa to promise that ID2 fetches the remained bytes before ID1 fetches a new instruction.

After fetches all remained bytes, ID2 would transform these bytes into suitable operands and pass all signals to the OF stage. If the instruction is a branch instruction, ID2 would calculate the target address and change the PC value if the branch is taken.

We had already implemented an asynchronous pipelined 8051 with Balsa. The Balsa program was compiled into a handshake component netlist, and finally this netlist was converted to a verilog gate-level netlist for Xilinx FPGA. With the gate-level netlist, we used other CAD tool to implement this circuit and do some simulation.

Because we wanted to implement the circuit in Xilinx FPGA, first we got the gate-level netlist by Balsa. Second, we imported this netlist into Xilinx ISE, a CAD tool for Xilinx FPGA. Then we added "keep hierarchy" description for each handshake component to avoid the optimization of CAD tool because the optimization will break the timing constraint. Finally we followed the standard design flow of the Xilinx FPGA, and burned the design into FPGA. All the flow is shown in Figure 2.

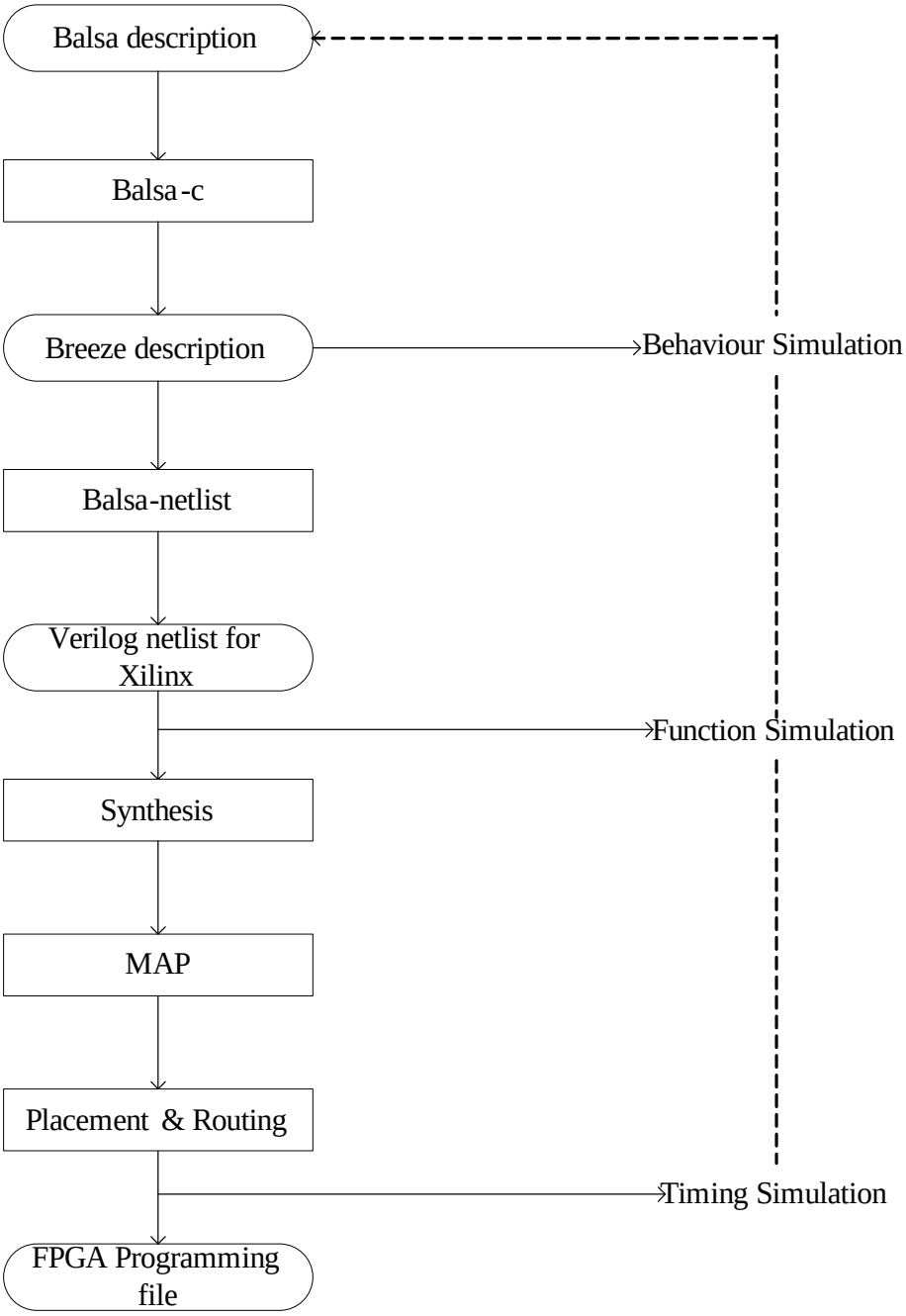

Figure 2. The FPGA design flow

參考文獻：

[1] A. Bardsley, D. A. Edwards, **"The Balsa Asynchronous Circuit Synthesis System"**

[2] Yuan-Teng Chang, **"SA8051: An Asynchronous Soft-core Processor for Low-Power System-on-Chip Applications",** *2005.*

[3] Cota, E.F.; Krug, M.R.; Lubaszewski, M.; Carro, L.; Susin, A.A, "**Implementing a self-testing 8051 microprocessor",** *Integrated Circuits and Systems Design, 1999. Proceedings. XII Symposium on 29 Sept.-2 Oct. 1999 Page(s):202 - 205*

[4] van Gageldonk, H.; van Berkel, K.; Peeters, A.; Baumann, D.; Gloor, D.; Stegmann, G.;" **An asynchronous low-power 80C51 microcontroller",** *Advanced Research in Asynchronous Circuits and Systems, 1998. Proceedings. 1998 Fourth International Symposium on , 30 March-2 April 1998 Pages:96 – 107*

[5] Intel, **"MCS51 Microprocessor Family User's Manual: Intel"**, *1994*

[6] Martin, A.J.; Nystrom, M.; Papadantonakis, K.; Penzes, P.I.; Prakash, P.; Wong, C.G.; Chang, J.; Ko, K.S.; Lee, B.; Ou, E.; Pugh, J.; Talvala, E.-V.; Tong, J.T.; Tura, A, "**The Lutonium: a sub-nanojoule asynchronous 8051 microcontroller",** *Asynchronous Circuits and Systems, 2003. Proceedings. Ninth International Symposium on 12-15 May 2003 Page(s):14 – 23*

[7] Je-Hoon Lee; Won-Chul Lee; Kyoung-Rok Cho, "**A novel asynchronous pipeline architecture for CISC type embedded controller, A8051",** *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on Volume 2, 4-7 Aug. 2002 Page(s):II-675 - II-678 vol.2*

[8] Chelcea, T.; Nowick, S.M.; "**Resynthesis and peephole transformations for the optimization of large-scale asynchronous systems",** *Design Automation Conference, 2002. Proceedings. 39th , 10-14 June 2002 Pages:405 – 410*

計畫成果：

We have successfully completed the behavior simulation in Balsa with small benchmarks such as GCD and Fibnacci Code. The simulation results are shown in the following.

1. The performance of the decoder

Because we integrate a cache-like buffer with our decoder, we need to know the effects with different kinds of buffers. We first measured the issue rate with different numbers of buffers. The buffer size is 32 bytes, and because the programs of GCD or Fibnacci Code are small, the benchmark is 256 instruction of the additions. The result is shown in Table 1.

The numbers of buffers	The consumed time (in Balsa units)	The normalized result
0	155,062,000	20.55
1	11,896,100	1.58
2	7,544,200	1
3	10,012,600	1.33

Table 1. The comparison of different numbers of buffers

Second, we measured the issue rate in different sizes of buffers. The buffer size are 8 bytes, 16 bytes, 32 bytes and 64 bytes. The result is shown in Table 2.

The size of buffer (byte)	The consumed time (in Balsa units)	The normalized result
8	7,801,000	1.03
16	7,629,800	1.01
32	7,544,200	1
64	7,501,400	0.99

Table 2. The comparison of different sizes of buffers

Finally, we measured the performance of the whole pipelined asynchronous 8051 with different sizes of the buffer. The result is shown in Table 3.

The size of buffer (byte)	The consumed time (in Balsa units)	The normalized result

8	13,681,400	1
16	13,681,400	1
32	13,681,400	1
64	13,681,400	1

Table 3. The performance of the whole pipelined asynchronous 8051 with different sizes of buffers

2. The performance of the pipelined asynchronous 8051

The comparison of performance and cost of single-cycle asynchronous 8051 and pipelined 8051 is shown in Table 4. We use the pipelined asynchronous 8051 with two 32-byte buffers to compare with the single-cycle 8051, SA8051[2], which was designed by our laboratory last year.

	The consumed time (in Balsa unit)	The normalized time	The cost (in Balsa unit)	The normalized cost
SA8051	24,891,300	3.30	210,513.5	0.43
PA8051	7,544,200	1	494,752.25	1

Table 4. The comparison between the 1-cycle asynchronous 8051 and the pipelined asynchronous 8051

3. Area cost

We use the Xilinx ISE 6.3i to synthesize our PA8051 processor, and the target FPGA chip is Xilinx FPGA Spartan-IIE 300 ft256.

The gate and path delays of every part of PA8051 are shown in Table 5. The ID stage is the most dominant stage of the whole design, taking half of the total cost of PA8051.

The biggest part is the ID stage. That is because that there are 256 cases of instructions. Even though we divided the instructions into regular and non-regular instructions, they still need to be multiplexed and that causes the cost so big.

	Slice	gate	minimum path delay(ns)
IF	1007	13987	757
ID	5353	61973	721
OF	564	7086	34
EXE	1284	16938	174
MEM_INTERFACE	1098	13217	125
RAM_READ_ARBITOR	57	1051	28
WB	232	2977	40
TOTAL	9595	117229	

Table 5. The Cost of Every Part of 8051

出席國際學術會議心得報告

計畫編號	NSC 94－2213－E－009－137－
計畫名稱	非同步 8051 處器器之研究與設計
出國人員姓名 服務機關及職稱	陳昌居 交通大學資訊資訊工程系副教授
會議時間地點	September 3-6, 2006, Wuhan and Three Gorges, China
會議名稱	The 3rd International Conference on Ubiquitous Intelligence and Computing (UIC-06)
發表論文題目	None

一、參加會議經過

9 月 2 日 早上經香港到武漢，晚上參加 reception 。

9 月 3 日 opening ceremony 之後有三個 keynote speeches、中午搭車到宜昌，晚上搭江輪。

9 月 4 日 早上觀光、下午時間論文發表。

9 月 5 日 全天論文發表。

9 月 6 日 早上論文發表，中午抵達重慶，晚間搭機經香港回台灣

二、與會心得

該會議有四大主題：1. Ubiquitous Intelligent/Smart Objects、2. Ubiquitous Intelligent/Smart Environments、3. Ubiquitous Intelligent/Smart Systems、4. Personal/Social/Physical Aspects，涵蓋內容有：* Electronic Label, Card, E-Tag and RFID * Embedded Chips, Sensor & Actuator * MEMS, NEMS, Mote & Biometric Device * Everyday Good, Artifact, Robot, etc. * Smart Appliance and Wearable Device * Material, Textile, Cloth, Furniture, etc. * Emerging Intelligent/Smart Objects * Embedded Software and Agents * Room, Home, Office, Laboratory, etc. * Building, Library, School, Campus, etc. * Shop, Clinic, Hospital and Health Care * Street, Yard, Park, Ground, City, etc. * Vehicle, Road, Traffic & Transportation * Land, Pool, Space and Hyperspace * Learning, Sport, Entertainment, etc. * Novel Intelligent/Smart Applications * Sensor, Ad Hoc & Intelligent Network * Knowledge Representation and Ontology * Wearable, Personal and Body Area Systems * OS, Middleware and Intelligent Association * Intelligent Service Architecture, Grid & Mesh * Massive Agents, Swarm/Amorphous Systems * Proactive, Autonomic and Organic Systems * Novel Intelligent/Smart Systems * Real/Cyber World Modeling and Semantics * End-User Interface, Control & Programming * Social/Natural/Physical Model of UI & SW * User/Object Identity and Activity Recognition * Security, Privacy, Trust and Legal/Policy Issues * Emotional, Ethical and Psychological Factors * Implication and Impact of UI and SW * Relations between Real and Cyber Worlds。內容相當廣泛豐富，不虛此行。