การออกแบบอาร์ทีซีชิพโดยใช้ภาษาวีเอชดีแอล

คมศิษย์ เต็มดี[1] และ บรรจง ปิยธำรง[2]
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทคัดย่อ

บทความนี้นำเสนอวิธีการออกแบบวงจรดิจิตอลขนาดเล็กจนถึงขนาดใหญ่อย่างมีประสิทธิภาพซึ่งรวดเร็วกว่าในอดีต โดยการใช้ภาษา VHDL (VHSIC Hardware Description Language) ซึ่งในบทความนี้ได้ทำการออกแบบชิพสร้างฐานเวลาจริง (RTC Chip) สำหรับการใช้งานกับไมโครโปรเซสเซอร์ หรือ ไมโครคอนโทรเลอร์ ซึ่งโครงสร้างภายในประกอบด้วย เวลาจริง, ปฏิทิน และ Static RAM อีก 24 byte ซึ่งไมโครโปรเซสเซอร์ กับตัว อาร์ทีซีชิพ จะทำการติดต่อสื่อสารข้อมูลกันแบบขนาน ในส่วนของเวลาจริง และปฏิทิน สามารถแสดงค่าของ วินาที, นาที, ชั่วโมง, วัน, เดือน และปี และวันสุดท้ายของเดือนที่น้อยกว่าจะถูกปรับอย่างอัตโนมัติ รวมทั้งปีอธิกสุรทิน ที่ 4 ปีจะมีหนึ่งครั้ง

[1] *นักศึกษาทดลองวิจัยปริญญาโท ภาควิชาวิศวกรรมคอมพิวเตอร์*
[2] *อาจารย์ประจำภาควิชาวิศวกรรมคอมพิวเตอร์*

RTC Chip Design Using VHDL Language

Komsit Temdee[1] and Bunjong Piyatamrong[2]

King Mongkut's Institute of Technology Lardkrabang

Abstract

This paper presents a modem method in designing digital system using VHDL (VHSIC Hardware Description Language). It is used to describe hardware for the purpose of developing, modeling, simulation and testing in design process. In this paper, the VHDL model of RTC Chip (Real Time Clock) is created for using with microprocessor and microcontroller. The RTC Chip contains a real time clock/calendar and 24 byte of static RAM. It communicates with a microprocessor via a simple parallel interface. The real time clock/calendar provides seconds, minutes,hours, days, dates, months and years information. The end of the month date is automatically adjusted for monthes with are less than 31 days and it is automatically adjusted the end of the month, date for months which have days less than 31 and also corrected for the leap year.

[1] Graduate Student, Department of Computer Engineering

[2] *Lecturer,* Department of Computer Engineering

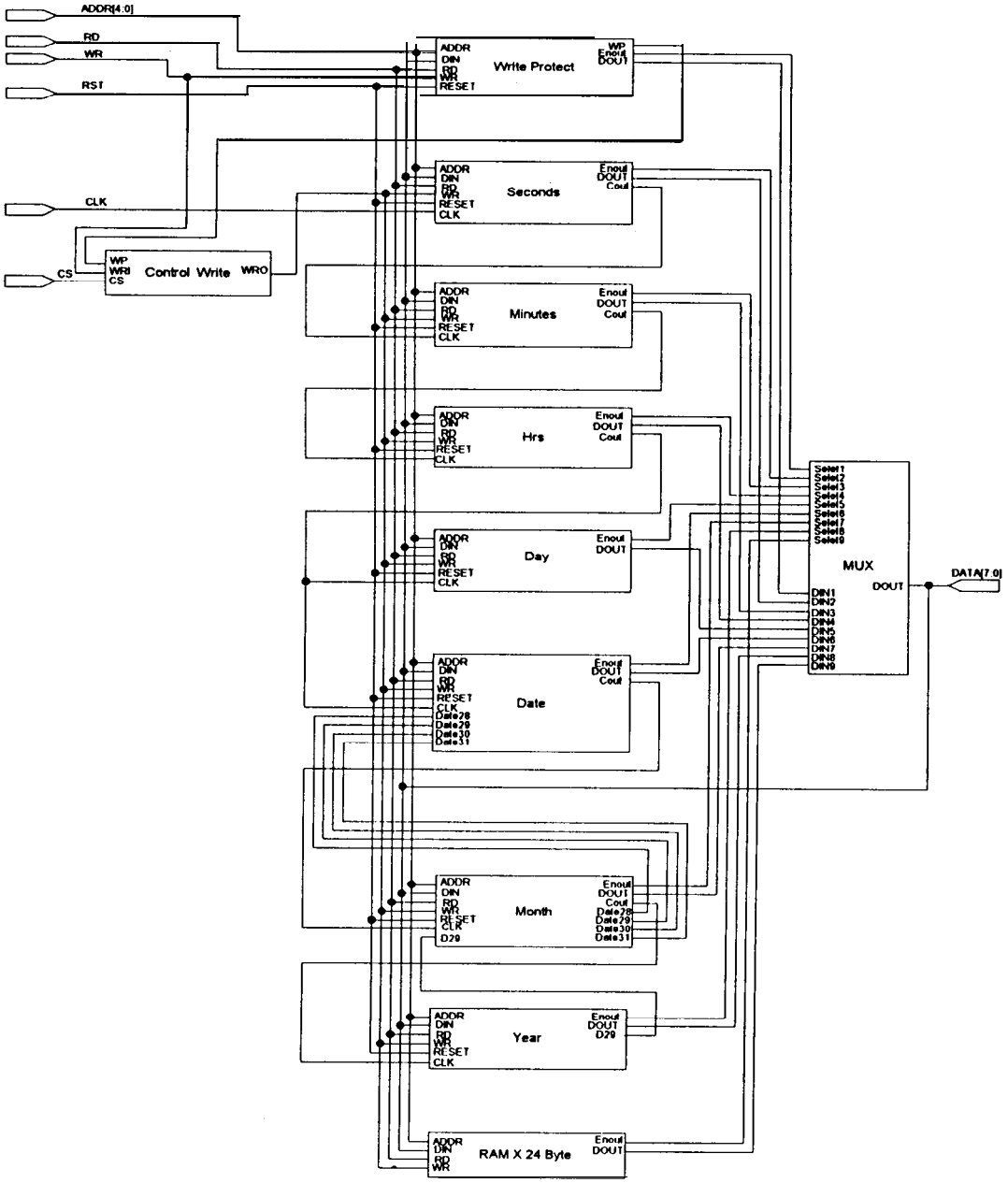
1. บทนำ

ในปัจจุบัน RTC Chip (Real Time Colock Chip) เป็นอุปกรณ์ที่สำคัญมาก ซึ่งนำไปประยุกต์ใช้งานกันอย่างแพร่หลาย เช่นระบบควบคุมเกี่ยวกับอุปกรณ์ในโรงงานอุตสาหกรรมหรือจะเป็นนาฬิกาที่ใช้ในการแสดงผลของเวลา และอุปกรณ์ที่ใช้ควบคุมเวลาได้หลายๆ อย่างภายในบ้านรวมไปถึงคอมพิวเตอร์ที่ใช้งานกันอยู่ทั่วไป บทความนี้จึงได้ทำการออกแบบ RTC Chip ซึ่งโครงสร้างภายในของ RTC Chip ได้ทำการออกแบบให้เหมือนกับ DS 1202 Serial Timekeeper Chip แตกต่างกันที่ออกแบบ และทำการปรับปรุงให้สามารถติดต่อสื่อสารข้อมูลกับอุปกรณ์ภายนอกแบบขนาน แทนแบบเดิมที่เป็นแบบอนุกรม ซึ่งใช้เวลาในการรับส่งข้อมูลนานกว่า เพราะทำการรับส่งข้อมูลครั้งละหนึ่งบิทเท่านั้น และจำเป็นต้องรับส่งข้อมูลครั้งละหลายๆ บิท เพื่อที่จะให้ได้ข้อมูลที่สมบูรณ์ หลังจากนั้นจึงจะนำไปใช้งานได้ แต่การรับส่งข้อมูลแบบขนานสามารถทำให้การติดต่อกับ RTC Chip เพียงแค่ครั้งเดียวเท่านั้นก็สามารถนำเอาข้อมูลที่ได้ไปใช้งานได้ทันที ซึ่งเหมาะสมกับงานควบคุมที่ต้องการความเที่ยงตรง เป็นอย่างมากประกอบกับโครงสร้างภายใน หรือ Register ต่างๆ ของ DS 1202 ที่ง่ายต่อการทำความเข้าใจ จะช่วยทำให้การใช้งานง่าย และมีประสิทธิภาพมากยิ่งขึ้นในการออกแบบได้ทำการพิจารณาแล้วว่า การใช้ภาษา VHDL ในการออกแบบและบรรยายการทำงานของ RTC Chipโดยใช้คอมพิวเตอร์ช่วยในการออกแบบ (Computer Aided Engineering) ทำให้การพัฒนาวงจร และการหาข้อผิดพลาดของวงจรทำได้รวดเร็วยิ่งขึ้น ซึ่งภาษา VHDL สามารถบรรยายได้ถึงทางพฤติกรรม (Behavioral Description) ผู้ออกแบบเพียงแต่รู้ว่าอินพุทเป็นอย่างไร และเอาท์พุทอะไรที่ต้องการโดยไม่จำเป็นต้องคำนึงถึงโครงสร้างภายในและการบรรยายเชิงโครงสร้าง (Structural Description) เป็นการบรรยายใกล้เคียงกับโครงสร้างของวงจรมากที่สุด เหมาะกับผู้ที่มีความรู้ทางด้านการออกแบบวงจรอยู่แล้ว ส่วนการบรรยายการไหลของข้อมูลทางอินพุทผ่านทาง Register หรือ Bus ของระบบหนึ่งไปยังอีกระบบหนึ่ง [1] ซึ่งการบรรยายทั้งหมดที่กล่าวมานี้ สามารถนำไปเลือกใช้งานได้ตามความเหมาะสม หรืออาจจะนำมาบรรยายร่วมกันก็ได้ ในบทความนี้ ได้แบ่งออกเป็น 5 หัวข้อ หัวข้อที่ 2 กล่าวถึงหลักการทำงาน และโครงสร้างภายในของฮาร์ดแวร์ หัวข้อที่ 3 กล่าวถึงขั้นตอนและการดำเนินงาน หัวข้อที่ 4 กล่าวถึงผลการดำเนินงาน และหัวข้อที่ 5 ได้กล่าวถึงสรุปผลการออกแบบ และรายละเอียดของการบรรยายเชิงพฤติกรรมของวงจรโดยใช้ภาษา VHDL อยู่ในภาคผนวก

2. หลักการทำงาน และโครงสร้างภายในของฮาร์ดแวร์

การออกแบบ และ การแบ่งโครงสร้างภายในของ RTC Chip ได้แสดงไว้ในรูปที่ 1 ซึ่งจะแบ่งออกเป็น 11 ส่วนรวมทั้งการต่อสายสัญญาณที่จำเป็นต่างๆ เข้าด้วยกัน และหน้าที่การทำงานของControl Register ตามตารางที่ 1 สามารถอธิบายได้ดังนี้

1. Seconds Register ทำหน้าที่เก็บข้อมูลในส่วนของนาฬิกาใช้แสดงค่าของ วินาที ดังแสดงไว้ในตารางที่ 1 ซึ่งข้อมูลที่เก็บลงไปจะเป็นรหัส BCD (Binary Code Decimal) มีค่าอยู่ในช่วง 00-59 ใช้งานใน Bit 0 ถึง Bit 6 ของ Seconds Register ได้ออกแบบให้ Seconds Register ทำงานที่สัญญาณนาฬิกา



รูปที่ 1 โครงสร้างภายในของ RTC Chip

ความถี่ 1 Hz และ Seconds Register ยังมี Bit Register ที่สำคัญอีกหนึ่งตัว คือ Bit 7 เรียกว่า Clock Halt Flag (CH) เมื่อ CH Bit เป็นลอจิก 1 จะทำให้ Register ทุกตัวหยุดเวลาต่าง ๆ ไว้ ณ. เวลานั้น (CH Bit จะเป็นลอจิก 0 ทุกครั้งหลังจากการได้รับสัญญาณ Power on reset)

2. Minutes Register ทำหน้าที่เก็บข้อมูลในส่วนของนาฬิกาใช้แสดงค่าของนาที ข้อมูลที่เก็บลงไปมีค่าอยู่ในช่วง 00-59 เช่นเดียวกับ Seconds Register และใช้งานใน Bit 0 ถึง Bit 6

3. Hrs Register ทำหน้าที่เก็บข้อมูลในส่วนของนาฬิกาใช้แสดงค่าของ ชั่วโมง และมี Bit Register ที่สำคัญคือ Bit 7 เป็นตัวกำหนดให้แสดงค่าแบบ 12 ชั่วโมง หรือ 24 ชั่วโมง โดยถ้า Bit นี้เป็นลอจิก 1 จะแสดงค่าเป็นแบบ 12 ชั่วโมง แต่ถ้าเป็นลอจิก 0 จะแสดงค่าเป็นแบบ 24 ชั่วโมง ส่วน Bit 5 จะบอกว่าเป็นการแสดงผล AM หรือ PM ซึ่งถ้า Bit 5 เป็นลอจิก 1 จะเป็น PM ถ้าเป็นลอจิก 0 จะเป็น AM

ตารางที่ 1 แสดงหน้าที่การทำงานของ Control Register

REGISTER	FUNCTION	RANGE DATA (BCD)	REGISTER DEFINITION							
			7	6	5	4	3	2	1	0
0	SECONDS	00-59	CH	10 SEC			SEC			
1	MINUTES	00-59		10 MIN			MIN			
2	12 HRS 24 HRS	01-12 00-23	12\ 24	0 0	AP 10	HR HR	HOUR			
3	DATE	01-31	0	0	10 DATE		DATE			
4	MONTH	01-12	0	0	0	10M	MONTH			
5	DAY	01-07	0	0	0	0	DAY			
6	YEAR	00-99	10 YEAR				YEAR			
7	WRITE PROTECT	00-80	WP	ALWAY ZERO						

4. Data Register ทำหน้าที่เก็บข้อมูลในส่วนของปฏิทินใช้แสดงค่าของวันที่ ใช้งาน Bit 0 ถึง Bit 5 มีค่าอยู่ในช่วง 01-31

5. Month Register ทำหน้าที่เก็บข้อมูลในส่วนของปฏิทิน ใช้แสดงค่าของเดือน ใช้งาน Bit 0 ถึง Bit 4 มีค่าอยู่ในช่วง 01-12

6. Day Register ทำหน้าที่เก็บข้อมูลในส่วนของปฏิทินใช้แสดงค่าของวัน มีค่าของข้อมูลอยู่ในช่วง 01-07 หรือ วันอาทิตย์ไปจนถึงวันเสาร์นั่นเอง

7. Year Register ทำหน้าที่เก็บข้อมูลในส่วนของปฏิทิน ใช้แสดงค่าของปี มีค่าอยู่ในช่วง 00-99

8. Write Protect Register (WP) ซึ่งจะใช้งานเพียง Bit เดียวคือ Bit 7 ซึ่งก่อนจะเขียนข้อมูลลงไปยัง Register ใดก็ตามจะต้องกำหนดให้ Bit นี้เป็น 0 ก่อนทุกครั้ง

ซึ่งปกติแล้ว หลังจาก Power on reset แล้ว แต่ถ้ากำหนดให้ Bit นี้เป็นลอจิก 1 ก็จะเป็นการป้องกันการเขียนข้อมูลลงไปยัง Register ต่างๆ

9. Control Write ทำหน้าที่ควบคุมสัญญาณการอ่านเขียน Register ต่างๆ และ RAM โดยจะตรวจสอบสัญญาณจาก WP Register ซึ่งถึงแม้ว่าจะมีสัญญาณเขียน RAM หรือ Register ต่างๆ เกิดขึ้น แต่ถ้าหาก WP มีลอจิกเป็น 1 สัญญาณเขียนข้อมูลทางด้านเอาท์พุทของ Control Write ก็จะไม่เกิดขึ้นนั่น หมายถึง จะไม่มีการเขียนข้อมูลทับลงไปยัง RTC Chip ได้เลย ซึ่งสัญญาณเขียนข้อมูลจะเกิดขึ้นได้เมื่อ WP มีลอจิกเป็น 0 เท่านั้น

10. RAM 24 Byte เป็นหน่วยความจำสำหรับใช้งานทั่วๆ ไปเป็น Static RAM ซึ่งไม่จำเป็นต้องออกแบบ สามารถนำมาใช้งานจาก Library X4000 ได้เลย

11. MUX (Multiplexers) ทำหน้าที่เลือกเอาข้อมูลจาก Output แต่ละตัวของ Register หรือ RAM ออกมายังขาข้อมูลของ RTC Chip โดยขึ้นอยู่กับสัญญาณควบคุม Enout จาก Register ตัวใดตัวหนึ่งที่ต้องการส่งข้อมูลของตัวมันเองออกไปยังขาข้อมูลภายนอก

จากที่โครงสร้างภายในของ RTC Chip ที่ได้กล่าวมาแล้วทั้งหมดยังมีขาสัญญาณและขาข้อมูลต่างๆ อีก 18 ขาของ RTC Chip ที่ใช้สำหรับติดต่อหรือต่อพ่วงกับอุปกรณ์ภายนอกต่างๆ ซึ่งมีหน้าที่การทำงานดังนี้

- ADDR เป็นขาที่ใช้ชี้ตำแหน่งของ Register ต่างๆ หรือ RAM ภายในตัว RTC Chip มีทั้งหมด 5 ขาคือ ขา A0-A4 สามารถชี้ตำแหน่งได้ 32 ตำแหน่ง โดยตำแหน่งที่ 0-7 เป็นตำแหน่งของ Register ต่างๆ ภายในตัว RTC Chip ส่วนตำแหน่งที่ 8-31 เป็นตำแหน่งของ RAM ขนาด 24 Byte

- DATA เป็นขาข้อมูลแบบสองทิศทาง (Bidirectional Bus) มีทั้งหมด 8 ขาคือ ขา D0-D7 ทำหน้าที่เป็นขาข้อมูลเข้า และออกของ Register ทุกตัวรวมทั้ง RAM ภายใน RTC Chip ด้วย

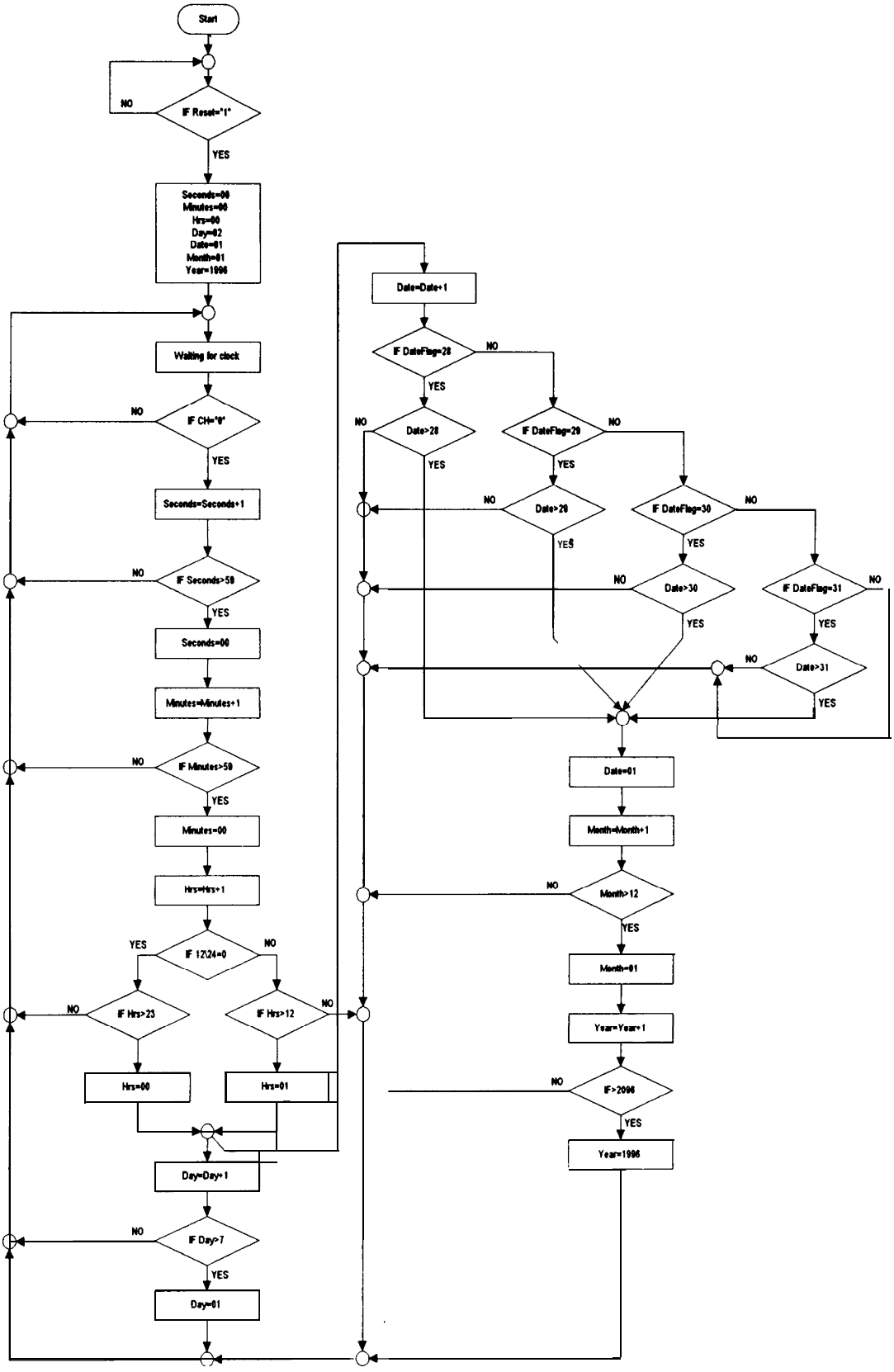
- RD เป็นขาสัญญาณควบคุมการอ่านข้อมูลจาก RTC Chip โดยเมื่อต้องการอ่านข้อมูล จะต้องให้ขา RD มีลอจิกเป็น 0 เสมอ

- WR เป็นขาสัญญาณควบคุมการเขียนข้อมูลลงไปในตัวของ RTC Chip โดยเมื่อต้องการเขียนข้อมูล จะต้องให้ขา WR มีลอจิกเป็น 0 เสมอ

- RST เป็นขาสัญญาณควบคุมการ Reset ทำงานที่ลอจิก 1 ซึ่งต้องได้รับการควบคุมจากอุปกรณ์ภายนอกให้เป็นลอจิก 1 ในขณะ Power on reset หรือเมื่อต้องการที่จะ Reset ตัว RTC Chip ณ. เวลาใดๆ ก็ตาม

- CLK เป็นขาสัญญาณนาฬิกาจากอุปกรณ์ภายนอกโดยต้องมีความถี่ 1 Hz เพื่อเป็นฐานเวลาให้กับ Register ต่างๆ ภายในตัว RTC Chip

- CS เป็นขาสัญญาณที่ต้องให้ลอจิกเป็น 0 ก่อนเสมอ เมื่อต้องการเขียน และอ่าน RTC Chip



รูปที่ 2 แสดงขั้นตอนการทำงานของ RTC Chip

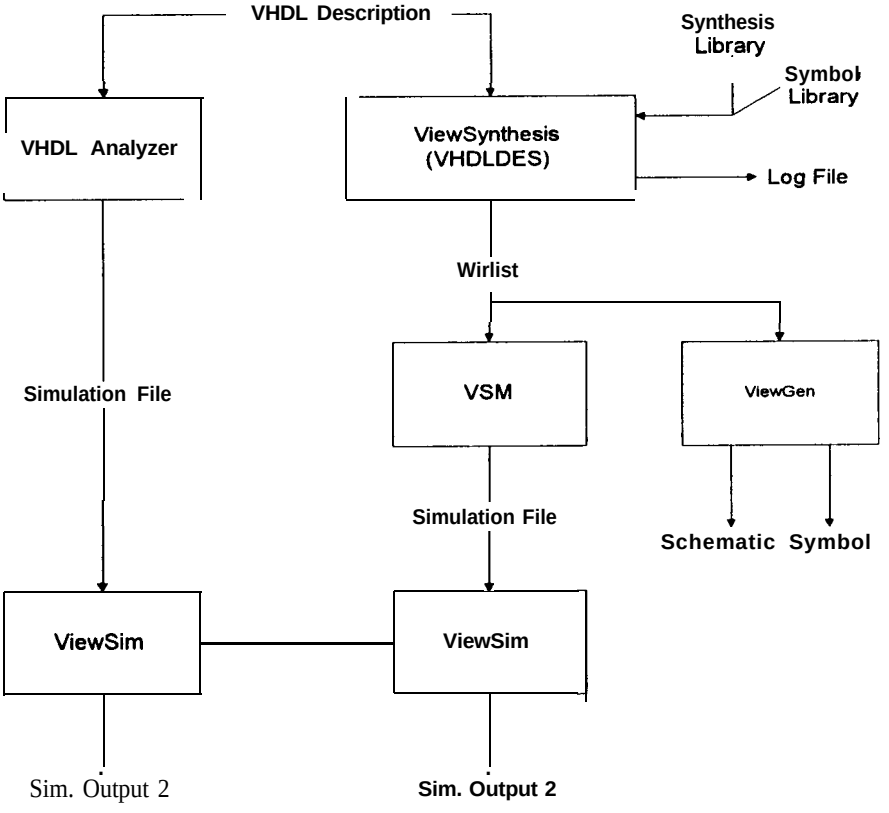
ดังที่ได้กล่าวถึงหน้าที่การทำงานแต่ละส่วนภายในตัว RTC Chip เมื่อนำส่วนต่างๆ มารวมกัน สามารถอธิบายถึงขั้นตอนในการทำงานต่างๆ ได้ดังนี้ คือ จากรูปที่ 2 เมื่อ RTC Chip ได้รับลอจิก 1 ที่ขา Reset ในขณะที่เริ่มจ่ายไฟเลี้ยงให้กับ RTC Chip หรือ ที่เรียกว่า Power on reset นั่นเอง Register ต่างๆ ที่อยู่ภายในแต่ละตัวจะกำหนดค่าเริ่มต้นของมันเอง ดังรูปที่ 3 หลังจากนั้น จะหยุดการทำงาน เพื่อรอสัญญาณนาฬิกาที่ขอบขาขึ้น เมื่อมีสัญญาณนาฬิกาเข้ามา Seconds Register จะทำงานเป็นส่วนแรก โดยจะทำการตรวจสอบ CH Bit ก่อนเป็นอันดับแรก ถ้าหากเป็นลอจิก 1 ก็จะทำให้ Seconds Register กลับไปรอสัญญาณนาฬิกาอีกครั้งหนึ่ง แต่ถ้าหากว่า CH Bit มีลอจิกเป็น 0 ส่วนของ Clock/Reset Sequential Statement ก็จะเริ่มทำงาน ซึ่ง Seconds Register ก็จะเริ่มนับขึ้นจาก 00 ในรหัส BCD ขึ้นไปเรื่อยๆ แต่ถ้ามีการเขียน หรือ อ่านข้อมูลในขณะใดขณะหนึ่ง เราต้องอาศัยความสามารถของภาษา VHDL ในส่วนของ Concurrent Signal Assignment [1] เข้าช่วยจะทำให้อ่าน และเขียนข้อมูลลงไปใน Register ต่างๆ ในเวลาใดก็ได้ ดังโปรแกรมที่แสดงไว้ในภาคผนวก ซึ่งทำการรวมทั้งส่วนของ Clock/Reset Sequential Statement และ Concurrent Signal เข้าด้วยกัน และเมื่อ Seconds Register นับขึ้นจนมากกว่า 59 มันก็จะทำการปรับข้อมูลในตัวมันเองให้เป็น 00 อีกครั้ง และสร้างสัญญาณทด หรือสัญญาณนาฬิกาไปให้ Minutes Register หลังจากนั้น Minutes Register ก็จะทำการนับขึ้น และมีหลักการเหมือนกับ Seconds Register นั่นเอง ส่วน Hrs Register เมื่อได้รับสัญญาณนาฬิกา ซึ่งเป็นสัญญาณทดจาก Minutes Register ในส่วนของ Hrs Register ก็จะทำการนับขึ้น หลังจากนั้นจะทำการตรวจสอบว่า Bit 7 (12/24) ถ้าเป็นลอจิก 1 ก็แสดงว่าตอนนี้กำลังแสดงผลอยู่ในโหมดของ 12 ชั่วโมง และหลังจากนั้นก็จะทำการปรับค่าของ Hrs Register อย่างเหมาะสม ถ้าหากว่ากำลังแสดงผลของชั่วโมงในโหมด 24 ชั่วโมงอยู่ มันก็จะทำการปรับค่าให้ตัวเองเป็น 00 เมื่อทำการตรวจสอบแล้วว่า ตอนนี้ค่าที่ทำการนับมากกว่า 23 แล้ว ซึ่งในโหมด 12 ชั่วโมงก็เช่นเดียวกัน ถ้าหากนับเกิน 12 แล้ว ก็จะทำการปรับค่าของ Hrs Register เป็น 01 พร้อมกับสร้างสัญญาณทดเพื่อเป็นสัญญาณนาฬิกาให้ Day Register และ Date Register ซึ่งในส่วนของ Date Register ก็จะมีการทำงานที่แตกต่างออกไป คือมันจะทำการตรวจสอบ DateFlag ก่อนเสมอ ซึ่ง DateFlag จะเก็บค่าของจำนวนวันที่สูงสุด ที่สามารถแสดงได้ ในแต่ละเดือนจากสายสัญญาณที่มาจาก Month Register ซึ่งแสดงการต่อสายสัญญาณไว้ดังรูปที่ 1 หลังจากนั้น Data Register ก็จะทำการปรับค่าตัวมันเองเป็น 01 เมื่อนับขึ้นจนมากกว่าค่าใน DateFlag และสร้างสัญญาณทด เพื่อเป็นสัญญาณนาฬิกาให้ Month Register ต่อไปในส่วนของ Month Register จะทำการปรับค่าของตัวมันเองเป็น 01 เมื่อตัวมันเองนับขึ้นจนมากกว่า 12 และ จะสร้างสัญญาณทด เพื่อเป็นสัญญาณนาฬิกาให้ Year Register ซึ่งในส่วนของ Year Register ได้ทำการออกแบบ ให้สามารถใช้เป็นปฏิทิน 100 ปี จาก ค.ศ. 1996-2096 และวันสุดท้ายของเดือนที่น้อยกว่าจะถูกปรับอย่างอัตโนมัติ รวมทั้งปีอธิกสุรทินที่ 4 ปีจะมี หนึ่งครั้ง โดย Platable [2] ของภาษา VHDL ซึ่งจะสะดวก และง่ายต่อการออกแบบ และการ Synthesis

3. ขั้นตอนและอุปกรณ์ในการดำเนินงาน

1. ทำการศึกษาการทำงานของ DS 1202 RTC Chip โดยเริ่มจาก Block Diagram ของ Register ต่างๆ ที่อยู่ในตัว DS 1202 หลังจากนั้นนำ DS 1202 มาใช้งานจริงโดยต่อพ่วงเข้ากับ ไมโครคอนโทรเลอร์ เบอร์ 8031

2. ทำการออกแบบ Module ในส่วนต่างๆ ของ RTC Chip โดยเปลี่ยนส่วนที่ใช้ต่อพ่วงกับอุปกรณ์ภายนอก แบบอนุกรมมาเป็นแบบขนาน หลังจากนั้นทำการแยก Module ของ Register ต่างๆ ออกจากกัน และกำหนดขาสัญญาณของ Register แต่ละตัว เพื่อที่สามารถต่อพ่วง Register ทั้งหมดเข้าด้วยกัน และเลือกการบรรยายของภาษา VHDL ชนิดต่างๆ ที่เหมาะสมกับการทำงานของ Register แต่ละตัวโดยใช้ Software Tools ที่เรียกว่า Workview Plus for Windows

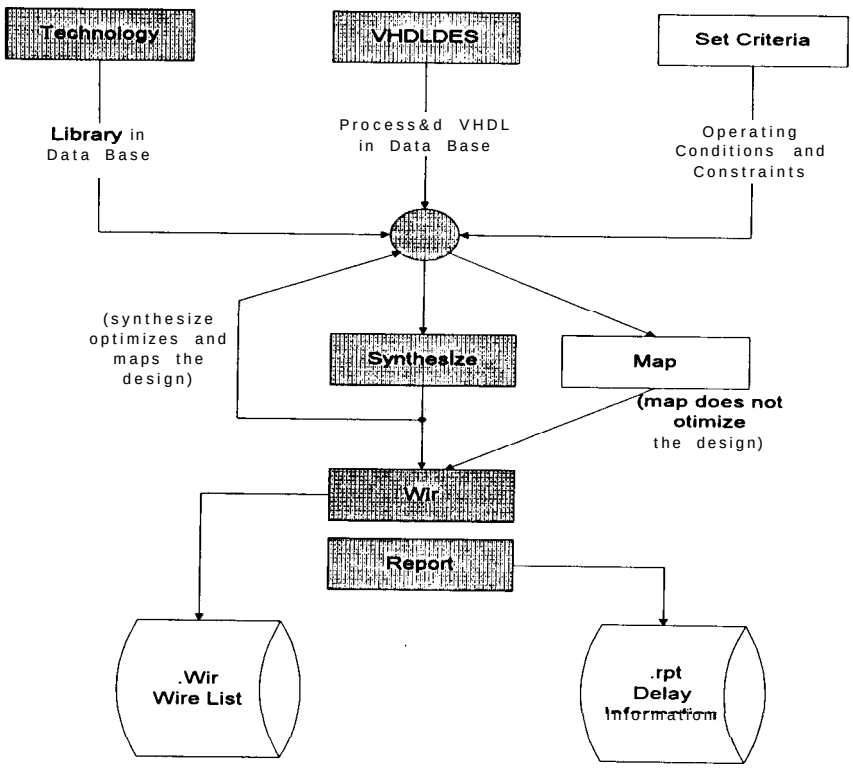
3. ทำการทดสอบ Source Code ของภาษา VHDL ของ Register แต่ละตัว โดยการ Simulate ดังแสดงขั้นตอนต่างๆ ไว้ดังรูปที่ 3 โดยจะมีทางเลือกสองทาง ทางเลือกที่หนึ่ง นำเอา Source Code ของภาษา VHDL มาทำการ Compile เพื่อตรวจสอบความถูกต้องของรูปแบบ (format) เพื่อสร้าง Simulate File โดยใช้ VHDL Analyzer เมื่อผ่านขั้นตอนนี้แล้ว ก็จะนำ Simulate File ไปทำการ Simulate โดยใช้ ViewSim ซึ่งขั้นตอนแรกที่กล่าวมานี้ ไม่สามารถที่จะนำไปทำการสังเคราะห์ให้เป็น Schematic ได้ส่วนทางเลือกที่สองจะนำ Source Code ของภาษา VHDL ผ่านการ Compile และ Synthesis โดยใช้ VHDLDES ก็จะได้ Wirlist File



รูปที่ 3 แสดงขั้นตอนของการ Simulate

ซึ่งจะเก็บตำแหน่งการเชื่อมต่อของขาอุปกรณ์ต่างๆ ไว้ และถ้าต้องการจะนำเอา Wirlist File ไปสร้างเป็น Schematic ก็จะต้องใช้ Viewgen ทำการสร้างอีกครั้งหนึ่ง แต่ถ้าต้องการจะนำเอา Wirlist ไปทำการ Simulate ก็ต้องใช้ VSM ทำการสร้าง Simulate File หลังจากนั้นก็จะทำการ Simulate โดยใช้ ViewSim เหมือนกับทางเลือกที่หนึ่ง ซึ่งหลังจากทำการ Simulate Register แต่ละตัวแล้ว ถ้าหากว่าไม่ได้ผล หรือ Timing ตามต้องการก็จะ นำเอา Source Code มาแก้ไขใหม่ หลังจากนั้นทำการ Compile แล้วจึง Simulate ใหม่อีกครั้ง ทำเช่นนี้ซ้ำไปเพื่อให้ได้ Timing ตามที่ต้องการ

4. ทำการสังเคราะห์ (Synthesis) ผังวงจรของ Register ต่างๆ ซึ่งได้แสดงขั้นตอนดังรูปที่ 4 เริ่มจากการเลือก Technology ที่ต้องการใช้ในการ Synthesis เช่น X3000, X4000 และ X7000 ของ Xilinx [3] เป็นต้น จากนั้นทำการเรียก Module ของ Register ที่ต้องการ Simulate ขึ้นมาเก็บในฐานข้อมูล (Data Base) แล้วทำการเลือกเงื่อนไขต่างๆ ที่ต้องการจะกำหนด เช่นพื้นที่ หรือความเร็วในการ Synthesis เป็นต้น จากนั้นก็ทำการ Synthesis ซึ่งจะมีให้สองทางเลือกคือการใช้คำสั่ง Map ซึ่งเป็นการ Synthesis ที่ไม่สามารถ ทำการ Optimize ได้ส่วนทางเลือกอีกทางคือการใช้คำสั่ง Synthesis ซึ่งเราสามารถทำการ Optimize โดยใช้คำสั่ง Resynthesis ซ้ำกี่ครั้งก็ได้ เพื่อให้ได้พื้นที่ และความเร็วที่เราต้องการ เมื่อได้ผลตามต้องการแล้วก็ทำการสร้าง Wirlist File โดยใช้คำสั่ง Wir และ หลังจากนั้น การใช้คำสั่ง Report เพื่อสร้าง Report File สำหรับดูผลที่ได้จากขบวนการ Synthesis ที่ผ่านมา



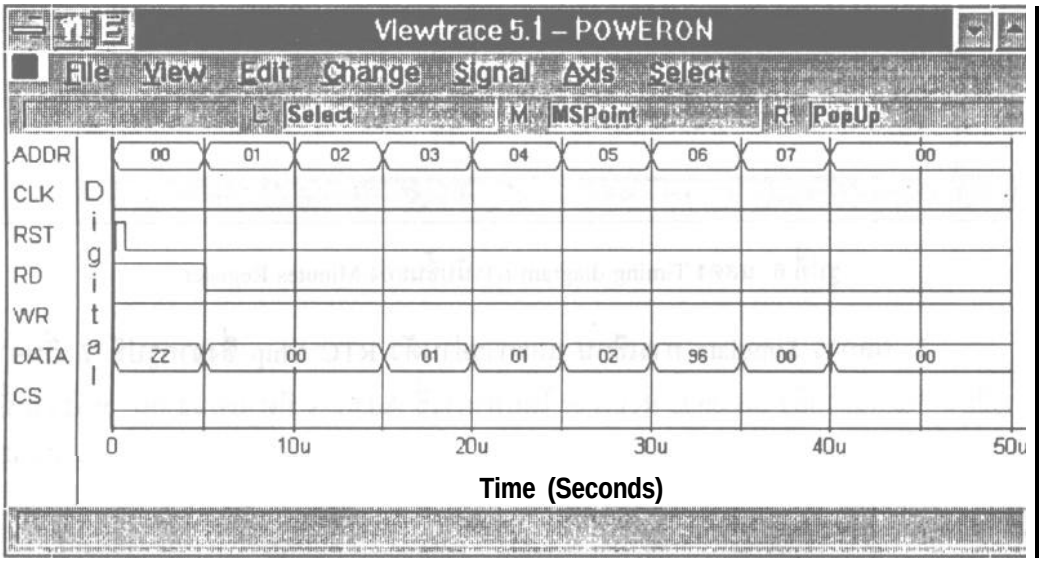
รูปที่ 4 แสดงขั้นตอนของการ Synthesis

5. ทำการรวม Module ของ Register ต่าง ๆ ที่ผ่านการ Simulate และการ Synthesis แล้วเข้าด้วยกัน โดยการใช้หลักการออกแบบที่เรียกว่า Hierarchical Design (Top-Down Design)

6. ทำการทดสอบการทำงานของ Hierarchical Design โดยการ Simulate อีกครั้งหนึ่งว่าได้Timing ตามที่ต้องการหรือไม่

4. ผลการดำเนินงาน

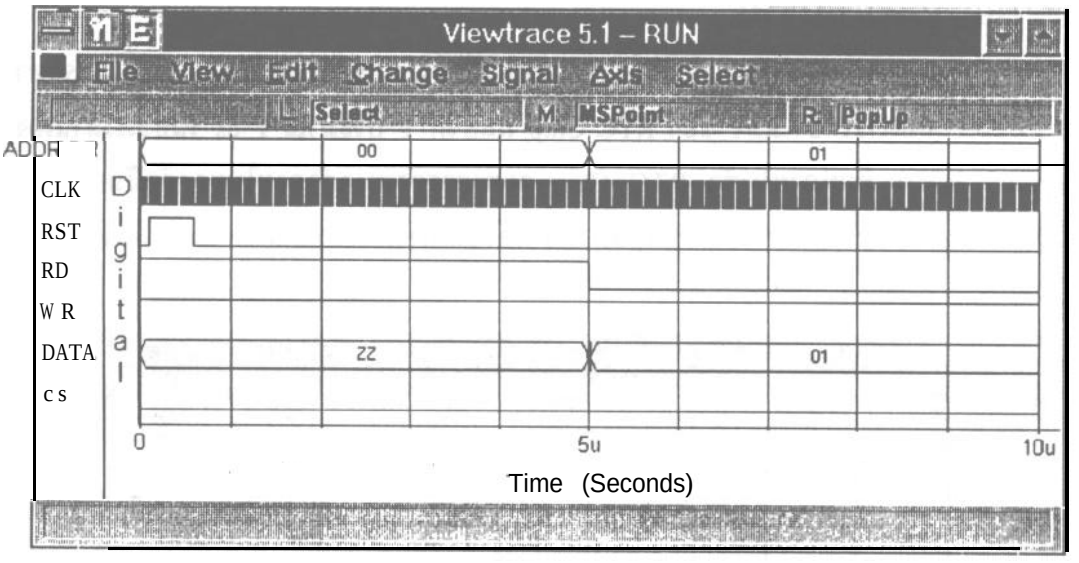
1. ผลจากการ Simulate หลังจากการเกิด Power on reset ซึ่งจากรูปที่ 5 จะทำการ Reset โดยการให้ขา RST ของ RTC Chip มีลอจิก 1 ประมาณ 1 us ในช่วงเวลานี้เอง Register ต่าง ๆ ที่อยู่ภายในตัว RTC Chip จะทำการกำหนดค่าภายในของตัวมันเอง ดังแสดงไว้ในรูปที่ 6 ซึ่งทำการอ่านค่าข้อมูลต่าง ๆ หลังจากการเกิด Power on reset ได้ดังนี้คือ



รูปที่ 5 แสดง Timing diagram หลังจาก Power on reset

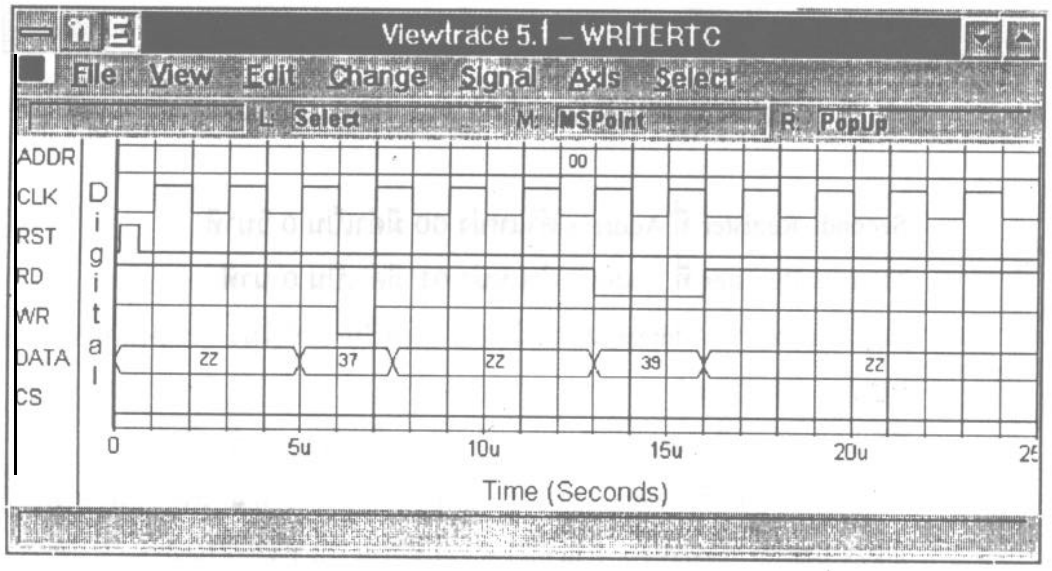
- Seconds Register ที่ Address ตำแหน่ง 00 มีค่าเป็น 0 วินาที
- Minutes Register ที่ Address ตำแหน่ง 01 มีค่าเป็น 0 นาที
- Hrs Register ที่ Address_ ตำแหน่ง 02 มีค่าเป็น 0 ชั่วโมง (ในโหมดการแสดงผลแบบ 24 ชั่วโมง)
- Date Register ที่ Address ตำแหน่ง 03 มีค่าเป็น 01 หรือวันที่ 1
- Month Register ที่ Address ตำแหน่ง **04** มีค่าเป็น 01 หรือเดือนมกราคม
- Day Register ที่ Address ตำแหน่ง 05 มีค่าเป็น 02 หรือวันจันทร์
- Year Register ที่ Address ตำแหน่ง 06 มีค่าเป็น 96 หรือ ปี 1996

2. ผลการ Simulate การนับขึ้นของ Minutes Register ซึ่งโดยปกติแล้วการนับขึ้นของ Seconds Register จะนับขึ้นในขณะขอบขาขึ้นของสัญญาณนาฬิกาที่ได้จากภายนอก แต่สำหรับ Register ตัวอื่น ๆ จะนับขึ้นได้ ก็ต่อเมื่อมีสัญญาณทดจาก Register ตัวใดตัวหนึ่ง จากรูปที่ 6 ยกตัวอย่าง เช่น Minutes Register จะนับขึ้นได้ก็ต่อเมื่อมีสัญญาณนาฬิกาเข้าไป ที่ขา CLK จนครบ 60 ลูก จะทำให้มีสัญญาณทดเกิดขึ้นจาก Seconds Register เพื่อเป็นสัญญาณนาฬิกาให้ Minutes Register ทำการนับขึ้นจากค่าเดิมที่มีอยู่คือ 00 เป็น 01 หรือเป็น 1 นาทีนั่นเอง



รูปที่ 6 แสดง Timing diagram การนับขึ้นของ Minutes Register

3. ผลการ Simulate การเขียน และการอ่านตัว RTC Chip ซึ่งจากรูปที่ 7 เริ่มจากการเขียนข้อมูลลงไปยัง Seconds Register โดยทำการชี้ Address ที่ตำแหน่ง 00 หลังจากนั้นให้ขา WR เป็นลอจิก 0 ในขณะที่สัญญาณที่ขา Address ต้องคงที่ และขา RD ต้องมีลอจิก



รูปที่ 7 แสดง Timing diagram ของการเขียนและการอ่านข้อมูล

เป็น 1 แล้วทำการเขียนข้อมูล 37 วินาทีลงไป เมื่อเขียนข้อมูลเสร็จเรียบร้อย ก็ให้ขา WR เป็นลอจิก 1 เหมือนเดิมจากรูปที่ 8 ที่เวลา 13 us เป็นการอ่านข้อมูลที่ทำการเขียนไปแล้วออกมาจาก RTC Chip โดยให้ขา RD มีลอจิกเป็น 0 ในขณะขาWR มีลอจิกเป็น 1 และสัญญาณชี้ตำแหน่งของ Address คงที่ ซึ่งข้อมูลที่ได้ออกมาคือ 39 วินาที เนื่องจากมีสัญญาณนาฬิกาขอบขาขึ้นได้ผ่านไปสองครั้งแล้ว

4. ผลที่ได้จากการ Synthesis ซึ่งในบทความนี้ขอยกตัวอย่างผลการ Synthesis เฉพาะ Day Register เท่านั้นซึ่งในส่วนของ Source Code ที่บรรยายถึงพฤติกรรมของ Day Register ได้แสดงไว้ในภาคผนวก และได้แสดงส่วนที่สำคัญสองส่วนจากผลของการ Synthesis คือส่วนที่แสดงถึงสภาวะแวดล้อมที่เราได้กำหนดทางเลือกต่างๆ ก่อนการ Synthesis เพื่อจะให้ได้ฮาร์ดแวร์ที่เหมาะสมตามความต้องการ ในส่วนที่สองคือ Gate Usage Summary เป็นส่วนที่แสดงถึงรายละเอียด และจำนวนของอุปกรณ์จาก Library ที่ได้เลือกมา Synthesis และได้แสดง Schematic ของ Day Register ไว้ดังรูปที่ 8

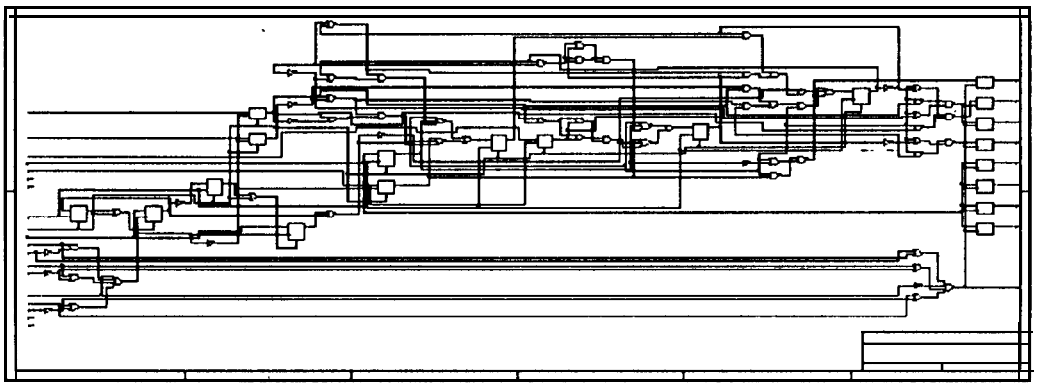
```
----- Module Name          : DAY
----- Library Name         : x4000
----- Operating Voltage    : 5.000000
----- Operating Temp       : 25
----- Process              : TYPICAL
----- Logic Type           : SMALL
----- area/speed Factor    : 1.000000
----- Package              : DEFAULT
```

****

Gate Usage Summary

****

Cell	Count	Area/Cell	Cell	Count	Area/Cell
MX4000:FD	8	0.00	MX4000:FDR	6	0.00
MX40000:FDS	2	0.00	MX4000:LDRD	4	3.00
X4000:AND2	10	0.25	X4000:INV	13	0.00
X4000:NAND2	12	0.25	X4000:NOR3	4	0.50
X4000:NOR4	1	0.75	X4000:OR2	23	0.25
X4000:OR3	1	0.50	X4000:OR4	1	0.75
Total Cells :	85		Total Area :	27.25	



รูปที่ 8 Schematic ของ Day Register ที่ได้จากการ Synthesis

5. สรุปและวิจารณ์

จากการออกแบบ RTC Chip ให้สามารถติดต่อสื่อสารข้อมูล กับอุปกรณ์ภายนอกแบบขนานแทนแบบเดิมที่เป็นแบบอนุกรม สิ่งหนึ่งที่แตกต่างกันอย่างเห็นได้ชัดคือ ความเร็วที่เหนือกว่า เนื่องจากหากเราต้องการจะติดต่อสื่อสารข้อมูลกับ RTC Chip เบอร์ DS 1202 แบบอนุกรม จะต้องใช้สัญญาณนาฬิกาถึง 16 ลูก ในการอ่านและเขียนข้อมูลแต่ละครั้ง แต่สำหรับ RTC Chip ที่ได้ออกแบบขึ้นใหม่ในบทความนี้ จะใช้เวลาเพียงชั่วขณะเดียวเท่านั้นในการอ่านเขียนข้อมูล เมื่อสัญญาณควบคุมการอ่านเขียนเกิดขึ้น รวมไปจนถึงข้อดีของการเขียนโปรแกรมที่ใช้ควบคุมการอ่านเขียนจากอุปกรณ์ภายนอก ก็ทำได้ง่ายกว่าอีกด้วย บทความนี้ยังได้นำความสามารถของภาษา VHDL มาช่วยในการออกแบบสถาปัตยกรรมของ RTC Chip ซึ่งมีความแตกต่างเป็นอย่างมาก ในการออกแบบวงจรดิจิตอลขนาดเล็กไปจนถึงขนาดใหญ่ เมื่อเปรียบเทียบกับอดีตที่ผ่านมา คือผู้ออกแบบต้องเริ่มจากการออกแบบด้วยมือ หลังจากนั้นเอาอุปกรณ์ตัวแต่ละตัวที่ทำการออกแบบมาทำการต่อทดลองในแผงวงจรจริง และทำการทดสอบวงจรเพื่อหาข้อผิดพลาด ซึ่งต้องใช้เวลานานกับการแก้ปัญหาแต่ละอย่างที่เกิดขึ้น แต่ในปัจจุบันด้วยความสามารถของคอมพิวเตอร์ช่วยในการออกแบบด้วยภาษา VHDL ผู้ออกแบบเพียงแต่เขียน Source Code บรรยายการทำงานของฮาร์ดแวร์ หลังจากนั้นก็ทำการ Compile แล้วทำการ Simulate ดูว่าได้ Timing ตามที่ต้องการหรือไม่ ถ้ายังไม่ได้ผลตามที่ต้องการก็ทำขั้นตอนเดิมที่กล่าวมาแล้วซ้ำอีกครั้ง ซึ่งหลังจากได้ผลที่ต้องการแล้วก็จะนำเอา Source Code ที่ได้ไปทำการ Synthesis เพื่อให้ได้ Schematic แล้วทำการ Simulate อีกครั้ง เพื่อเปรียบเทียบกับผลการ Simulate จาก Source Code ที่ได้ในครั้งแรกเมื่อได้ผลตามต้องการก็จะนำเอา Sehematic ที่ได้ไปทำการ Map ลงไปยัง FPGA (Field Programmable Gate Array) เพื่อเป็น Chip ต้นแบบสำหรับใช้งาน ถึงอย่างไรก็ตามการใช้งานภาษา VHDL ถ้าผู้ใช้ทำการออกแบบฮาร์ดแวร์โดยเลือกใช้การบรรยายเชิงพฤติกรรมจะทำให้ได้ Schematic ที่มีลอจิกเกตเป็นจำนวนมาก ซึ่งเหมาะสมกับงานที่ต้องการ

ความรวดเร็วในการออกแบบในขณะที่ผู้ออกแบบไม่จำเป็นต้องมีความรู้ทางด้านการออกแบบวงจรมากนัก แต่ถ้าหากผู้ออกแบบเลือกการบรรยายของฮาร์ดแวร์ในระดับโครงสร้าง จนถึงระดับ Gate Level จะทำให้ Schematic ที่ได้มีลอจิกเกตน้อยกว่าการบรรยายเชิงพฤติกรรม ซึ่งเหมาะสำหรับการ Optimize วงจรให้มีประสิทธิภาพ และต้องการให้ Schematic ที่ได้มีลอจิกเกตที่น้อยที่สุด

เอกสารอ้างอิง

1. Zainalabedin Navabi, 1993, VHDL : Analysis and Modeling of Digital System, McGraw-Hill Inc., pp 1-13.
2. Viewlogic, 1992, VHDL Reference Manual for Synthesis, Viewlogic System Inc., pp l-17.
3. Xilinx, 1994, The Programmable Logic Data Book, Xilinx.

กิตติกรรมประกาศ

โครงการวิจัยนี้ได้รับความช่วยเหลือในเรื่องทุนวิจัย และเครื่องมือต่างๆ ภายใต้การสนับสนุนของ "โครงการความร่วมมือด้านการวิจัย และพัฒนาทางวิศวกรรม" ระหว่างสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กับ บริษัท อาร์แอนด์ดี คอมพิวเตอร์ซิสเท็ม จำกัด

ภาคผนวก

```
library synth;
use synth.stdsynth.all;
entity Day is
port (signal CLK, RD, WR, RST :in vlbit;
    signal ADDR :in vlbit_1d (4 downto 0);
    signal En_out: out vlbit;
    signal DIN : in vlbit_1d (7 downto 0);
    signal DOUT : out vlbit_1d (7 downto 0));
end Day;
architecture behavioral of Day is
signal statel, Data:vlbit_ld (7 downto 0);
signal state2, nextstatel, nextstate :vlbit_1d (3 downto 0);
signal flagl, resetflagl , resetflag2, clkdffc, clkdlatchc, datainflagl , datainflag2, resetdffc1,
resetdffc2 : vlbit; constant day_table:vlbit_2d (0 to 7,7 downto O):=(
```

```
    B"0001_0010",
    B"0010_0011",
    B"0011_0100",
    B"0100_0101",
    B"0101_0110",
    B"0110_0111",
    B"0111_0001",
    B"1XXX_0001");
begin
    flag1 <='1';
    clkdlatchc< = '1' when ((ADDR = B "00101") and (RD = '1') and (WR = '0'))
    else '0';
    dlatchc_v (DIN, RST, clkdlatchc, statel);
    clkdffc< = '0' when ((ADDR = B "00101") and (RD = '1') and (WR = '0'))
    else '1';
    dffc( flagl, resetdffcl, clkdffc, datainflagl);
    resetdffc1< = '1' when ((RST = '1') or (resetflag1 = '1')) else '0';
    dffc (flagl, resetdffc2, clkdffc, datainflag2);
    resetdffc2< = '1' when ((RST='1') or (resetflag = '1')) else '0';
    Data< =B "0000" & state1 (3 downto 0) when ((datainflagl = '1') or (datainflag2
    = '1')) else B "0000" & state2;
    En_out< = '1' when ((ADDR = B"OO1O1") and (RD = '0') and (WR = '1')) else '0';
    dff_v (Data, En-out, DOUT);
    pla_table (state1 (3 downto 0), nextstatel, day-table);
    pla_table (state2, nextstate2, day-table);
-------------------------------
  process
  begin
  wait until ((prising(CLK)) or (RST='1'));
    if (RST='1') then
    state2< = B "0010";
    resetflag1< = '0';
    resetflag2< = '1';
```

```
    else
        if (datainflagl = ' 1') then
            state2< = nextstatel;
            resetflag1< = '1';
            resetflag2< = '0';
        elsif (datainflag2 = '1') then
            state2< = nextstatel;
            resetflag1< = '0';
            resetflag2< = '1';
        else
            state2< = nextstate2;
            resetflag1< = '0';
            resetflag2< = '1';
        end if;
    end if;
  end process;
-----------------------------
end behavioral
```