Cika Electrónica

Comentario Técnico: CTC-008
Título: **Rabbit 3000A**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	2/09/03	

Comentamos acerca de la nueva revisión de Rabbit 3000, conocido como Rabbit 3000A. El documento cubre las principales diferencias y mejoras de estos procesadores con respecto al Rabbit 3000 original, por lo que se recomienda la lectura del tutorial sobre Rabbit 3000 (CTU-003), y tal vez Rabbit 2000 (CTU-002), dependiendo del grado de conocimientos previos sobre la familia Rabbit.

Índice de contenido

Introducción

El Rabbit 3000A es el sucesor del Rabbit 3000. No sólo corrige anomalías en el R3000 sino que además agrega muchas prestaciones nuevas, como por ejemplo:

- Frecuencia de clock más alta
- Soporte mejorado para el bus auxiliar de I/O
- Mayor control de la generación de PWM, facilidad de interrupción
- Contador de 10 bits para los decodificadores de cuadratura
- Nuevas instrucciones para acelerar encripción
- Nuevas instrucciones en la familia de “cargar y repetir”
- Protección de memoria
- Protección de desborde de stack
- Modo usuario y modo sistema

El R3000A es completamente compatible con el R3000, las características nuevas deben habilitarse expresamente.

Mejoras y diferencias

Corrección de anomalías (bug fixes)

Se corrigen las siguientes anomalías:

- Se agrega un disparador Schmitt a la entrada de reloj de 32KHz, evitando que el ruido presente en la entrada del oscilador pudiera ocasionar que el RTC se incrementara demasiado rápido.
- Port paralelo F: Un error en la decodificación de las direcciones de los registros ocasionaba interferencia entre los ports A y F (ver TN228). Ha sido corregido.
- Short chip selects e interrupciones: en raras ocasiones podían ocasionar una falla en la lectura de memoria. Ha sido corregido.
- Cambios en el bus auxiliar de I/O durante las interrupciones. Corregido
- *LDDR*/*LDIR* y wait-states: El primer acceso en una instrucción de éstas era correcto, pero los restantes empleaban 0 wait-states. Corregido.
- Byte superior de direcciones de I/O. Si bien esto no es una anomalía, todos los procesadores anteriores sólo tomaban en cuenta el byte menos significativo para decodificar una dirección de I/O interna. Para soportar las prestaciones adicionales del R3000A fue necesario decodificar la palabra entera. Por este motivo, debe observarse el valor del registro conteniendo la parte alta de la dirección al efectuar una operación de I/O en registros internos, por ejemplo:

```
ld a, 0x55        ; dato a preservar
ld h, a           ; se guarda en H
ld l, 0x24        ; L contiene la dirección de SPCR
ld a, 0x84        ; valor a escribir en SPCR
ioi ld (hl), a    ; escribe 0x84 en SPCR
ld a, h           ; recupera el dato guardado en H
```

Si bien esto funciona en cualquier procesador anterior, no funciona en el R3000A, debido a que el mismo escribirá en *0x5524* en vez de *0x24*.

Hardware

Se realizaron las siguientes mejoras de hardware:

- Se mejoraron los retardos internos en los módulos de clock doubler y spectrum spreader, logrando frecuencias de clock más altas.
- Puede habilitarse la extensión del tiempo de hold del ciclo de lectura de I/O externos por un período de clock adicional.
- Cada uno de los bancos de I/O asociados al port E (I/O strobes) puede asignarse al bus de memoria o al bus auxiliar de I/O.
- Además de la opción de short chip select que existía para ciclos de lectura, se agrega una para ciclos de escritura.
- PWM: se ha agregado la posibilidad de generar una interrupción y de anular la salida. La interrupción puede configurarse para ocurrir cada 1, 2, 4 ú 8 ciclos; la intención es mejorar la generación de audio y el control de servos.
- Se extendió a 10 bits la resolución del contador de los decodificadores de cuadratura, de modo de reducir la carga de software al trabajar con encoders de alta resolución.
- Se agregó un timer de 8 bits cuya fuente de reloj es el oscilador de 32KHz. Este timer cuenta de forma descendente desde un valor seteado, y al llegar a cero genera una interrupción de prioridad 3: *Watchdog secundario*.

CPU

Modo sistema y modo usuario

Se trata de un sistema de dos niveles de control de la CPU. Un nivel (*sistema*) permite acceso total a todos los recursos, mientras que el otro (*usuario*) es más limitado. El modo *sistema* es equivalente al modo normal de funcionamiento del R3000 y R2000. Si no se habilita esta modalidad, la CPU se comporta como sus predecesores, las nuevas instrucciones que se describen a continuación funcionan como se indica, pero se ignora el modo de operación.

Al habilitar el modo usuario/sistema:

- El modo de operación de la CPU se mantiene en un nuevo registro de la CPU: *SU*, similar a *IP*.

- El reconocimiento de una interrupción, además de las tareas regulares asociadas, ocasiona que la CPU opere en modo *sistema*.

Si el procesador está en modo *usuario*:

- Cualquier escritura en un registro de I/O es ignorada, a menos que el *bit de autorización* (*permission bit*) haya sido seteado en el *registro de autorización* (*user enable register*) correspondiente a ese periférico.
- No se permite ejecución a prioridad 3. Si el usuario la setea explícitamente, el procesador operará con prioridad 2.
- Ante la ejecución de la nueva instrucción *IDET*, ocurrirá una interrupción de prioridad 3: *System Violation*

Si el procesador está en modo *sistema*, opera como lo hacía tradicionalmente, y la instrucción *IDET* carece de efecto (ver más adelante).

Este esquema ha sido diseñado de modo de proveer un soporte simple para un concepto de sistemas operativos: las rutinas del kernel operan en modo *sistema*, y el código del usuario en un modo más restringido (*usuario*). Debido a las limitaciones de este último, las interrupciones del sistema (protección de memoria, desborde del stack, etc.) siempre serán atendidas. El usuario puede solicitar una acción al sistema operativo (OS) mediante la instrucción *SYSCALL*; la acción propiamente dicha puede detallarse en un registro, stack, etc. Los periféricos son, por defecto, controlados por el OS, el usuario puede solicitar acceso a un periférico (y registrar una interrupción, si es necesario) mediante peticiones al OS.

Instrucciones

Criptografía

Se agregaron dos instrucciones para acelerar los cálculos en operaciones criptográficas. Éstas desarrollan operaciones en variables largas (varios bytes), apuntadas por un registro índice. La variable apuntada por el registro índice es multiplicada por una constante de 8 bits y sumada o restada a otra variable similar; el resultado se almacena en memoria.

Instruction	Bytes	Clks	A	I	S	Z	V	C	Operation
UMA	2	8+8i		-	-	-	-	*	{D:(HL)} = (IX) + [(IY) * E + D]; BC = BC - 1; IX = IX + 1; IY = IY + 1; HL = HL + 1; repeat while BC != 0
UMS	2	8+8i		-	-	-	-	*	{D:(HL)} = (IX) - [(IY) * E + D]; BC = BC - 1; IX = IX + 1; IY = IY + 1; HL = HL + 1; repeat while BC != 0

Carga y repetición (Load and repeat)

Las instrucciones tradicionales *LDIR* y *LDDR* tienen limitaciones a la hora de acceder a registros de I/O; el prefijo sólo afecta al destino, y el hecho de que la dirección sea autoincrementada hace que generalmente no sean de utilidad en estos casos. A tal fin, se agregaron las siguientes instrucciones:

Instruction	Bytes	Clks	A	I	S	Z	V	C	Operation
LDDSR	2	6+7i		d	-	-	*	-	(DE) = (HL); BC = BC - 1; HL = HL - 1; repeat while BC != 0
LDISR	2	6+7i		d	-	-	*	-	(DE) = (HL); BC = BC - 1; HL = HL + 1; repeat while BC != 0
LSDR	2	6+7i		s	-	-	*	-	(DE) = (HL); BC = BC - 1; DE = DE - 1; HL = HL - 1; repeat while BC != 0
LSIR	2	6+7i		s	-	-	*	-	(DE) = (HL); BC = BC - 1; DE = DE + 1; HL = HL + 1; repeat while BC != 0
LSDDR	2	6+7i		s	-	-	*	-	(DE) = (HL); BC = BC - 1; DE = DE - 1; repeat while BC != 0
LSIDR	2	6+7i		s	-	-	*	-	(DE) = (HL); BC = BC - 1; DE = DE + 1; repeat while BC != 0

Modo usuario y modo sistema

Se agregaron siete instrucciones nuevas, a fin de soportar estos dos modos de operación. Todas excepto *IDET* corresponden a opcodes nuevos. *IDET* en particular comparte su opcode con la instrucción *LD E,E* y solamente funciona como *IDET* cuando el procesador opera en modo *usuario*.

Las nuevas instrucciones son las siguientes:

Instruction	Bytes	clk	A	I	S	Z	V	C	Operation	Priv?
SETUSR	2	4		-	-	-	-	-	SU = {SU[5:0], 0x01}	yes
PUSH SU	2	9		-	-	-	-	-	(SP-1) = SU; SP = SP - 1	yes
POP SU	2	7		-	-	-	-	-	SU = (SP); SP = SP + 1	yes
SURES	2	4		-	-	-	-	-	SU = { SU[1:0], SU[7:2]}	yes
IDET	1	2		-	-	-	-	-	Performs "LD E,E", but if (EDMF && SU[0]) then the System Violation interrupt flag is set	no
RDMODE	2	4		-	-	-	-	*	CF = SU[0]	yes
SYSCALL	2	10		-	-	-	-	-	SP = SP - 2; PC = {R,v} where v = SYSCALL offset	no

- *SETUSR* pone la CPU en modo *usuario*, modificando apropiadamente el registro *SU* (push)
- *PUSH SU* y *POP SU* guardan y recuperan, respectivamente, este registro en el stack
- *SURES* hace una operación de pop en el registro *SU*, retornando al modo anterior
- *IDET* ocasiona una interrupción si la CPU opera en modo *usuario*
- *RDMODE* devuelve el modo de operación en el flag de carry
- *SYSCALL* pone la CPU en modo sistema y salta a una posición en la tabla de interrupciones

Manejo de Memoria

Protección de Memoria

Los predecesores del R3000A podían arbitrar la protección contra escritura por bancos de 256K. Ahora esto puede hacerse con una resolución de 64K, y dos de estos segmentos en particular pueden resolverse cada 4K.

Si se intenta una operación de escritura en un segmento protegido, se produce una interrupción de prioridad 3: *Write Violation*.

Protección de desborde de stack

Pueden setearse un límite superior y uno inferior para el stack, con resolución de 256 bytes. Si se produce una escritura relativa al SP cercana (16 bytes) o más allá de estos límites, se produce una interrupción de prioridad 3: *Stack Violation*. Los dieciseis bytes mencionados proveen suficiente espacio en el stack como para que la interrupción generada pueda operar y resolver el problema.

Segmento de RAM en código en segmento root

Esta característica resuelve la dificultad para actualizar código al operar en el modo *espacios separados para I&D*. Al habilitarla, se crea una ventana de 1 a 4 K de RAM que puede contener código.