Analisi Tecnica del Firmware

Progetto *Vera-Module-RBL-XE*

Analisi del progetto a cura di RetroBit Lab

6 ottobre 2025

Indice

1 Introduzione

Questo documento presenta un'analisi tecnica del firmware per il progetto *Vera-Module-RBL-XE*. L'obiettivo del firmware, eseguito su un microcontrollore ESP32, è di interfacciarsi con il bus di un processore 6502 per emulare un dispositivo PBI (Parallel Bus Interface) per computer ATARI.

L'analisi si concentra sulla fattibilità temporale (timing) dell'interazione tra la MCU moderna e la CPU vintage, illustrando la logica di funzionamento tramite diagrammi di flusso e temporali.

2 Architettura del Sistema

Il sistema si basa sull'interazione di due componenti principali.

2.1 Componenti Chiave

- **CPU 6502:** Il processore centrale dell'Atari, operante a una frequenza di clock di **1.79 MHz**.
- **MCU ESP32-S3:** Un moderno microcontrollore dual-core con una frequenza di clock di **240 MHz**, responsabile del monitoraggio del bus e dell'emulazione della periferica.

2.2 Interfacciamento Bus

L'ESP32 è collegato direttamente alle seguenti linee del bus del 6502:

- **Bus Indirizzi (A0-A15):** Per determinare a quale locazione di memoria la CPU vuole accedere.
- **Bus Dati (D0-D7):** Per leggere o scrivere dati.
- **Segnali di Controllo:** Principalmente `PHI2` (il clock principale della CPU) e `R/W` (indica un'operazione di lettura o scrittura).

3 Analisi delle Tempistiche

Una delle principali preoccupazioni in un progetto di questo tipo è la capacità della MCU di rispondere entro la finestra temporale definita dal ciclo di clock della CPU 6502.

- **Ciclo Clock 6502 (1.79 MHz):** $T_{6502} = 1/(1.79 \times 10^6) \approx 558$ ns
- **Ciclo Clock ESP32 (240 MHz):** $T_{ESP32} = 1/(240 \times 10^6) \approx 4.17$ ns

Il rapporto tra i due periodi di clock è di circa **134:1**. Questo significa che l'ESP32 ha a disposizione oltre 130 dei suoi cicli di clock per ogni singolo ciclo del 6502. Questo margine è eccezionalmente ampio e garantisce che tutte le operazioni di lettura dei pin, elaborazione logica e scrittura sul bus possano essere completate con largo anticipo.

3.1 Diagramma Temporale di un Ciclo di Lettura

Il diagramma seguente illustra un tipico ciclo di lettura gestito dall'ESP32. L'asse X rappresenta il tempo, mentre l'asse Y mostra lo stato logico dei segnali.

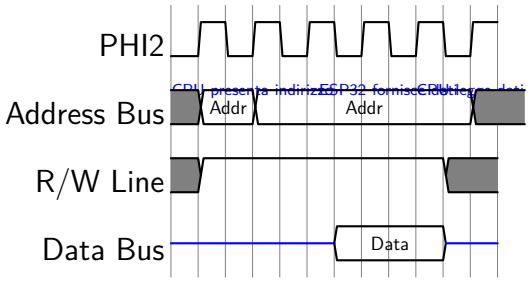

Figura 1: Diagramma temporale di un ciclo di lettura del 6502 gestito dall'ESP32.

4 Frammento di Codice Sorgente

Di seguito è riportato un estratto significativo del file `main.cpp`, che implementa la logica descritta nel diagramma di flusso. L'uso di accesso diretto ai registri (`read_gpio_level`, `read_address_bus`) e di un task dedicato (`MonitorTask`) è fondamentale per le prestazioni.

```
void MonitorTask(void *pvParameters)
{
    serialPrintQueue(ANSI_BLUE "6502 Bus Monitor Ready on Core 1\n"
   ANSI_RESET);

    for (;;)
    {
        // Attesa del fronte di salita di PHI2
        while (!read_gpio_level(PIN_PHI2))
            ;;

        uint16_t address = read_address_bus();
        bool rw = read_gpio_level(PIN_RW);

        // Esempio: gestione dell\'area di memoria CCTL ($D5xx)
        if (address >= 0xD500 && address <= 0xD5FF)
        {
            if (rw) // La CPU sta leggendo
            {
                uint8_t data = d500[address & 0x00FF];
                set_data_bus_direction(GPIO_MODE_OUTPUT);
                write_data_bus(data);

                // Attendi che il ciclo di clock finisca
                while (read_gpio_level(PIN_PHI2))
                    ;;

                set_data_bus_direction(GPIO_MODE_INPUT);
            }
```

```
            else // La CPU sta scrivendo
            {
                set_data_bus_direction(GPIO_MODE_INPUT);
                uint8_t data = read_data_bus();
                d500[address & 0x00FF] = data;
            }
        }
        // ... altre condizioni per PBI I/O, ROM, etc.
    }
}
```

Listing 1: Estratto dal MonitorTask in main.cpp

5 Diagramma di Flusso del Firmware

La logica principale del firmware è contenuta nel `MonitorTask`, che viene eseguito sul secondo core dell'ESP32 per garantire prestazioni in tempo reale.

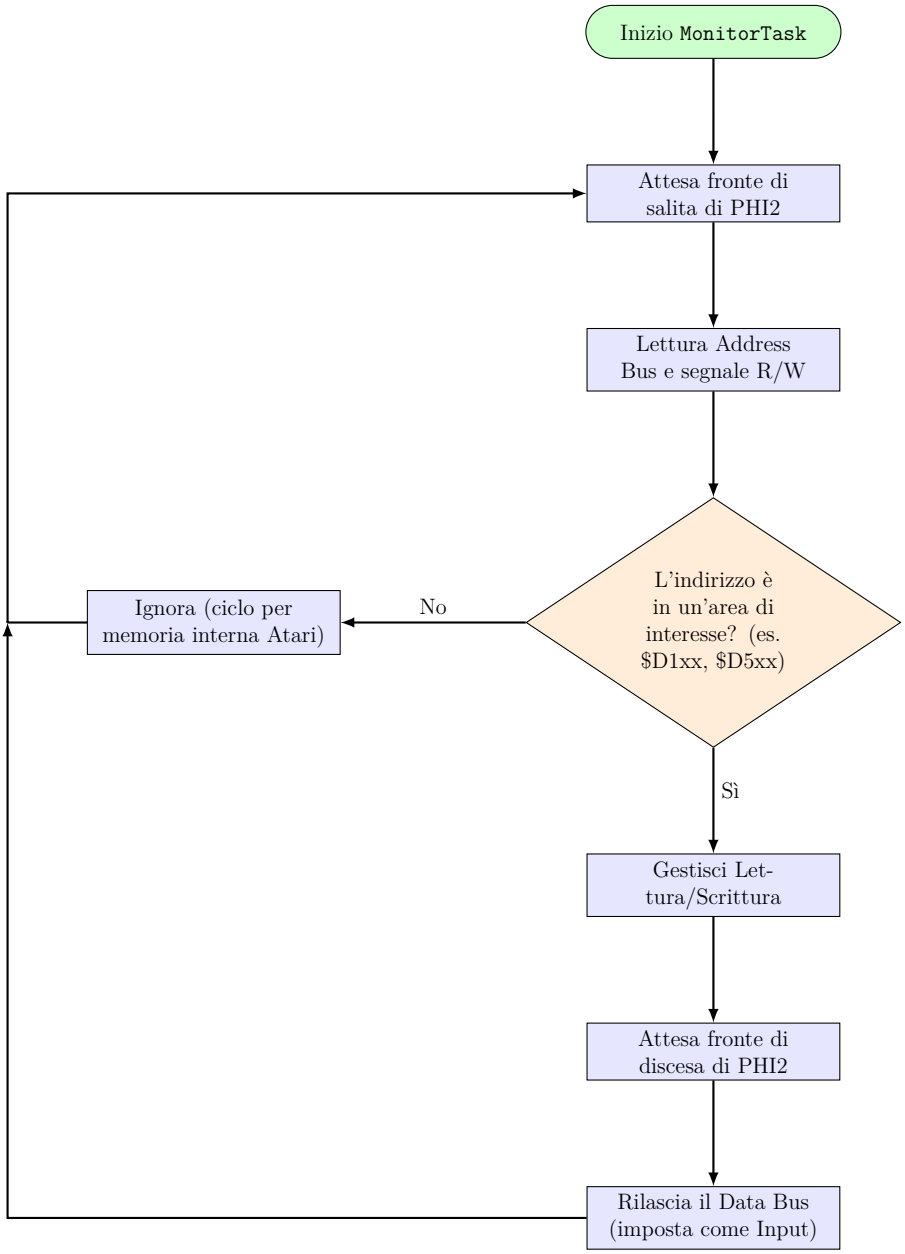

Figura 2: Diagramma di flusso semplificato del `MonitorTask`.

6 Conclusione

L'analisi conferma che l'architettura hardware e software del progetto è solida e performante. L'uso di un microcontrollore ESP32 a 240 MHz fornisce un margine di elaborazione più che sufficiente per interfacciarsi con un bus 6502 a 1.79 MHz (o anche 2 MHz) senza incorrere in alcun problema di tempistica. Le tecniche di programmazione a basso livello, come l'accesso diretto ai registri GPIO e l'uso di task dedicati, sono state implementate correttamente per garantire la massima efficienza.