Efficient interrupt-driven UART FIFO Workload

2023 Fall SOC Design – Final Project Proposal

311551095 資工科碩二 林聖博

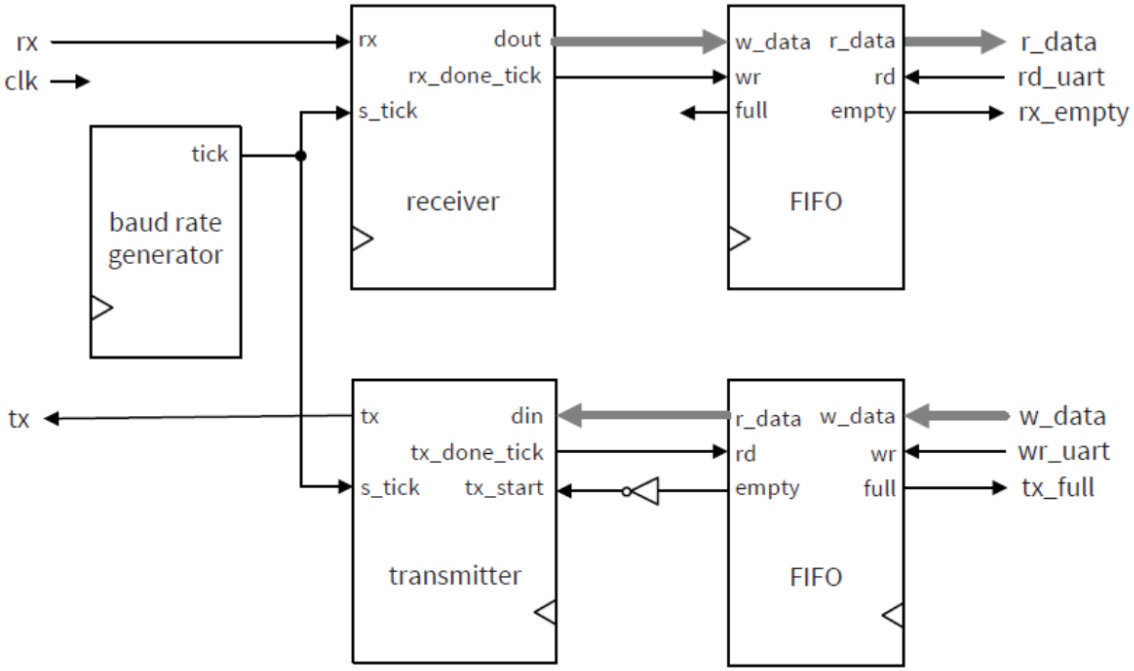
312551174 資工科碩一 張祐誠

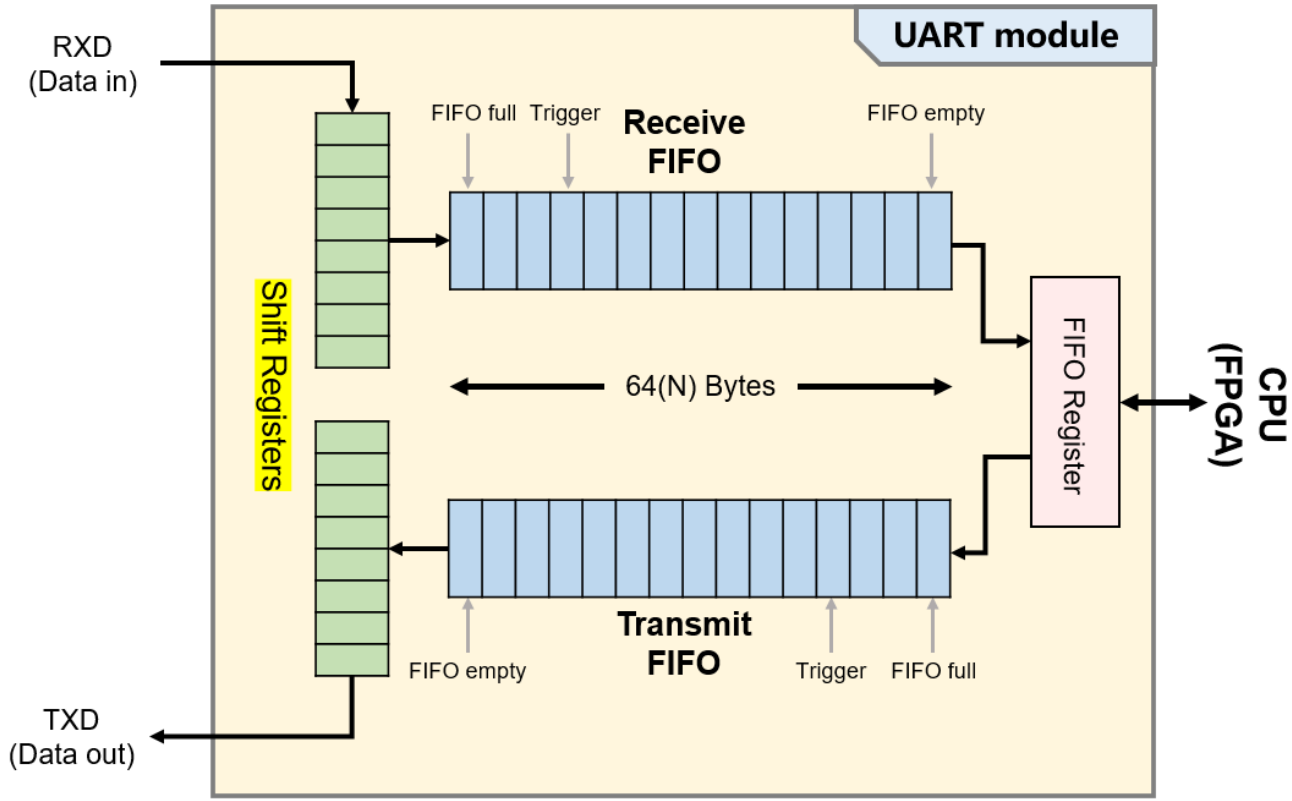
Team 13

2023/12/13

Function Specification

- Block Diagram





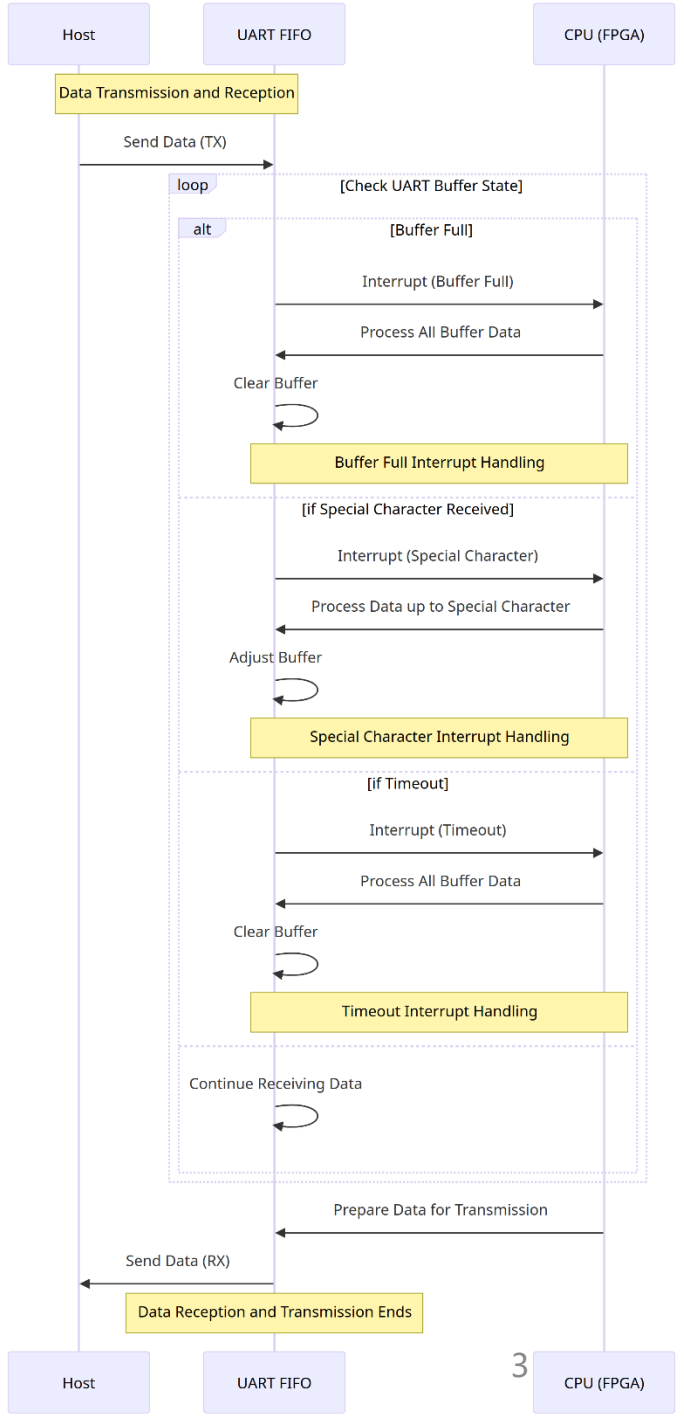
Function Specification

- Handshake

1. **Buffer Full：** 當 Buffer 填滿時，UART FIFO 會通知 CPU 處理所有 Buffer (緩衝區)中的資料，然後清空 Buffer，以便接收後續的資料

2. **Special Character Received：**若在資料中接收到特定的結束字元，如: \n，UART FIFO 會通知 CPU 處理到該特殊字元為止的所有資料，然後調整 Buffer 的剩餘部分

3. **Timeout：**若在超過設定的時間沒有接收到新資料，UART FIFO 會通知 CPU 處理所有 Buffer 的資料，並清空 Buffer



Function Specification

Register spec / Function

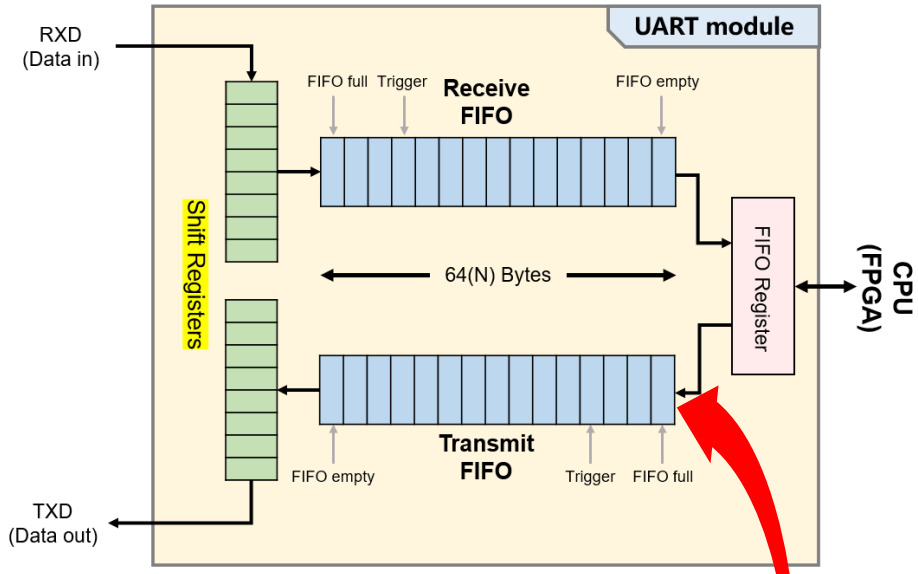
Register Name	Function	Offset	Bits
CR (Control Register)	Control (enables, resets, breaks)	0x00000000	9
IER (Interrupt Enable Reg.)	Enables possible interrupt sources	0x00000004	13
IDR (Interrupt Disable Reg.)	Disables possible interrupt sources	0x00000008	13
ISR (Interrupt Status Reg.)	Indicates which interrupt are currently asserted	0x0000000C	13
BBR (Clock divider / Baud Rate Reg.)	Baud rate clock divider for main sample clock	0x00000010	16
FCR (FIFO Control Reg.)	Control FIFO operations, such as setting the FIFO trigger level and resetting the FIFO	0x00000014	8
RXTOUT (Idle time set)	Sets receiver idle time before possible interrupt issued	0x00000018	8
RXWM (Receive fill level)	Sets receive FIFO fill level before possible interrupt issued	0x0000001C	6
FIFO (Read/write FIFO data port)	Transmit and receive FIFO data port	0x00000020	8
SR (FIFO status)	FIFO status register (full, empty, near full, etc.)	0x00000024	10
TX_Trigger (Transmit fill level)	Sets transmit FIFO fill level before possible interrupt issued	0x00000028	6
TXFIFO	Store data to be sent via UART	0x0000008C	512 (64 Bytes)
RXFIFO	Store data received from UART	0x000000F0	512 (64 Bytes)

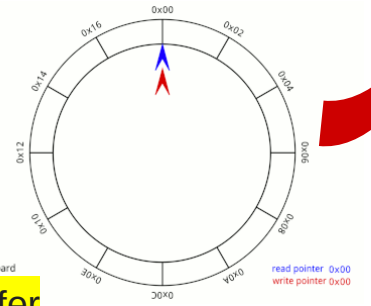
Function Specification

- Feature
  - 64-Byte Managed FIFOs + 8-Bits Data Port
  - Shift register
    - Transmission, reception and storage of serial data to ensure data accuracy and timing
  - Reduce CPU load and the number of interrupts

➢ Optional

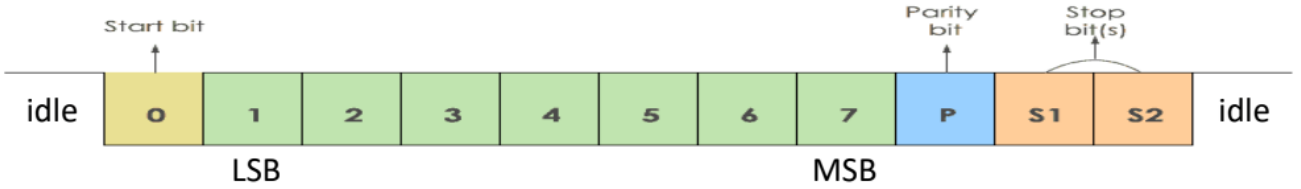
  - Circular Buffers
    - **Hardware FIFOs** have a **fixed size**, and if data exceeds their capacity, a **software circular buffer** in SRAM is used to prevent busy waiting.
  - FIFO_sync / FIFO_async





Circular Buffer

Function Specification



- Caravel **9 bit/byte**
  - Data bits: 8
  - Parity bit: 0
  - Stop bit: 1

- Estimated performance

1. Buffer capacity and Data overflow

➢ **64-Bytes FIFO**：提供 64-Byte Buffer Size，CPU 讀取之前可存儲 64 bytes 的資料。減少高速資料傳輸時發生 Data Overflow 的風險

➢ **預估 Buffer time**：相比無 FIFO 的設計，使用 64-Byte Buffer Size，假設 UART Baud Rate 為 115200，FIFO 提供約 5 毫秒(ms) 的緩衝時間

- Total Bits = FIFO Size × Bits per Byte = 64 bytes × **9 bit/byte** = 576 bits
- Baud Rate = 115200 bits/sec

- $Buffer\ Time = \frac{Total\ Bits}{Baud\ Rate} = \frac{576\ bits}{115200\ bits/sec} \approx 0.005\ s\ \ or\ 5\ ms$

Function Specification

- Estimated performance

2. Interrupt management

➢ **減少中斷次數：不使用 FIFO 的情況下，每接收到一個字元(Byte)可能產生一次中斷**。使用 64-Byte FIFO，可以設定中斷觸發條件

- 例如: 當 FIFO 達到一定 fill level（如 32 Bytes (half full)）時才產生中斷

➢ **預估中斷頻率(Interrupt Frequency)**：假設每秒接收 1000 Bytes 下，使用 FIFO 的中斷頻率大約降至每秒約 31 次，相比原來的 1000 次中斷，減少約 97%

- 如果每 32 bytes (Half full) 觸發一次中斷，則每秒中斷的次數為:

$$Interrupts\ per\ second = \frac{Number\ of\ bytes\ per\ interrupt}{bytes\ per\ second} = \frac{32_{bytes/interrupt}}{1000_{bytes/sec}} \approx 31.25\ times/sec$$

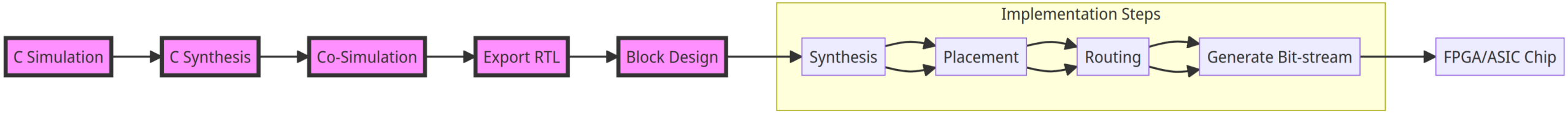
Work Partition

組員一： 林聖博

- 設計 Testbench
- Fireware Code 撰寫
- 整合 User Project Wrapper
- 優化與調整 UART FIFO
- FPGA 測試
- 撰寫報告與統整實驗流程與結果評估

組員二： 張祐誠

- 設計 UART FIFO module
- 設計 FIFO FSM
- 驗證 UART FIFO 邏輯與行為
- 優化與調整 UART FIFO
- FPGA 測試
- 撰寫報告與統整實驗流程與結果評估



Schedule Plan

Week 1
(2023/12/18)

基本 FIFO 設計針對 UART

- Protocol 制定與實現
- 基於 Reg. Spec 設計 UART FIFO
- User Proj. Wrapper 整合
- 初步 Testbench 撰寫

Week 2
(2023/12/25)

調整與優化 UART FIFO

- 邏輯優化與調整 (FSM)
- Fireware Code 撰寫
- 反覆微調(tuning)設計或架構，如: Reg.使用, FIFO size
- 研究與探索現今已有的 UART FIFO 設計與參考

Week 3
(2024/01/01)

驗證與測試

- 鑑測設計與觀察行為是否符合預期
- 合成並於 FPGA 上進行測試
- 收集測試結果與資源使用率等

Week 4
(2024/01/08)

整合成果與報告製作

- 彙整設計實驗內容
- 提交程式並完善 Github Repo
- 撰寫設計觀察與數據結果分析
- 製作報告與簡報

Deliverables

完成 UART FIFO Advanced Features
並與原始 UART 比較能有明顯效能提升的成果