UART

논리회로실습

부경대 컴퓨터·인공지능공학부 최필주

목차

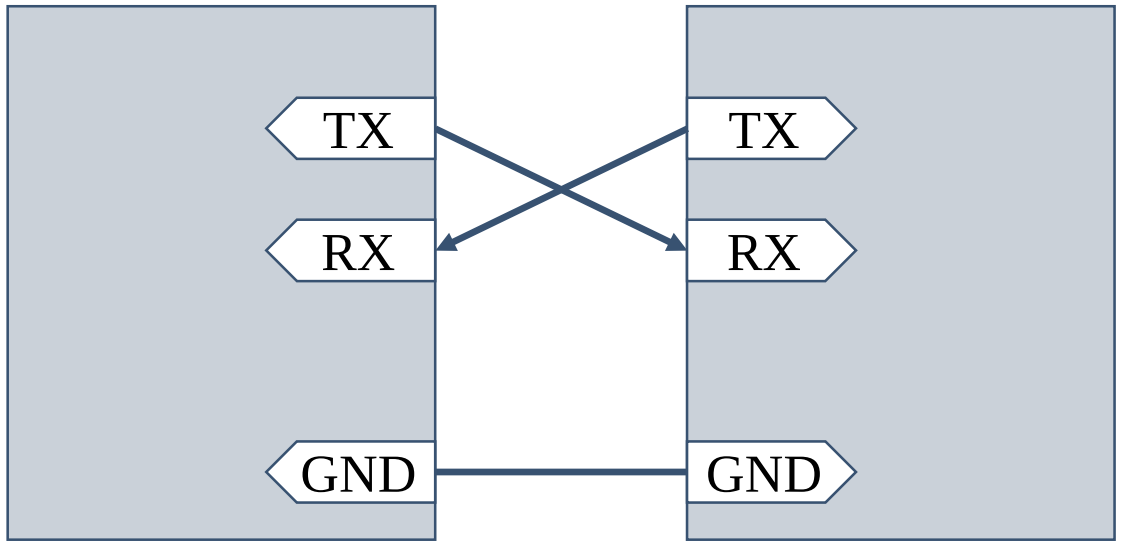
- UART 개요
- 하드웨어 설계
  - Rx 모듈
  - Tx 모듈

UART 개요

- UART란?
  - Universal Asynchronous Receiver / Transmitter의 약자
  - 두 장치 간 직렬 데이터 교환을 위한 프로토콜

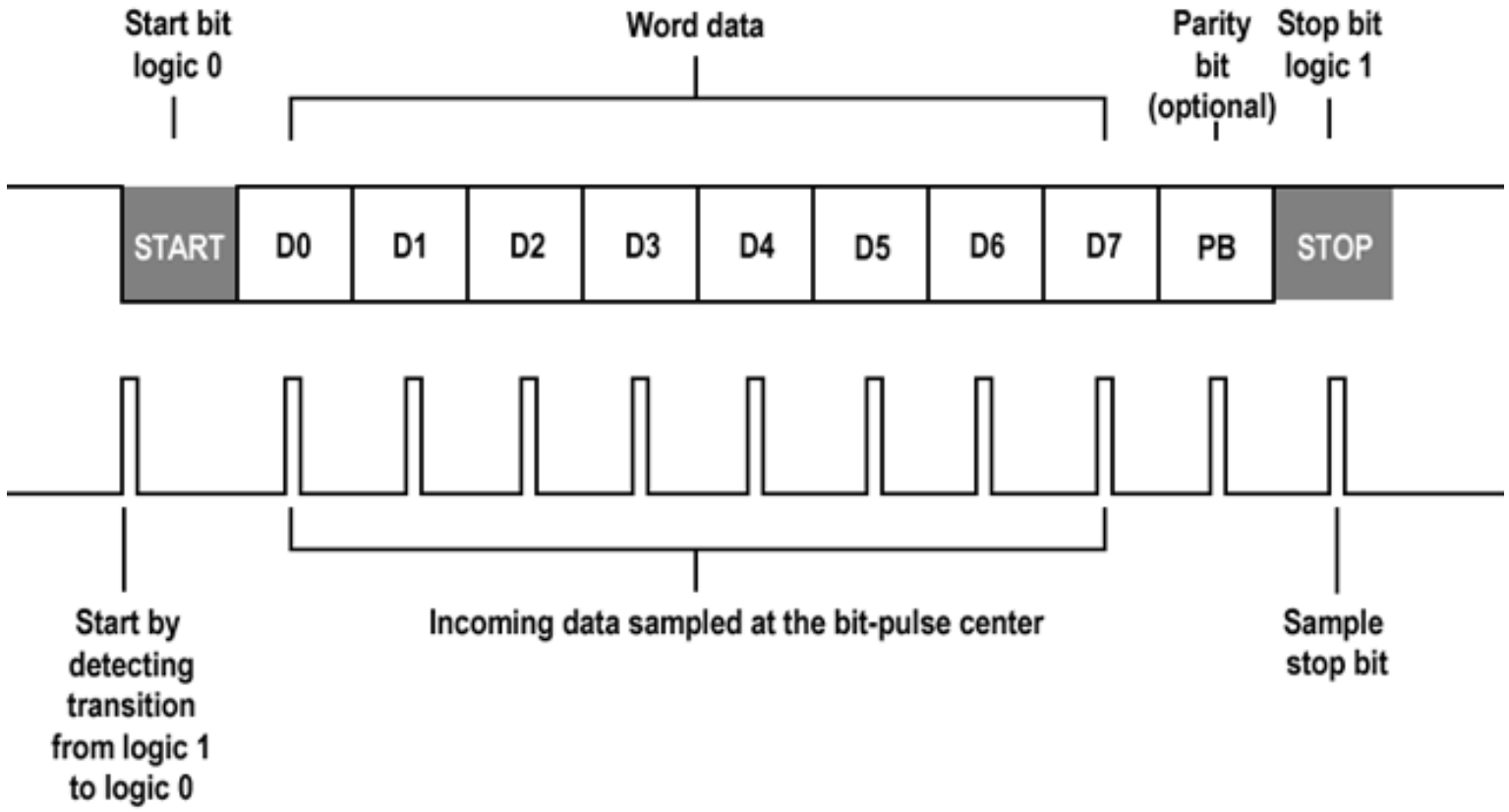
UART 개요

- 장치간 UART 연결



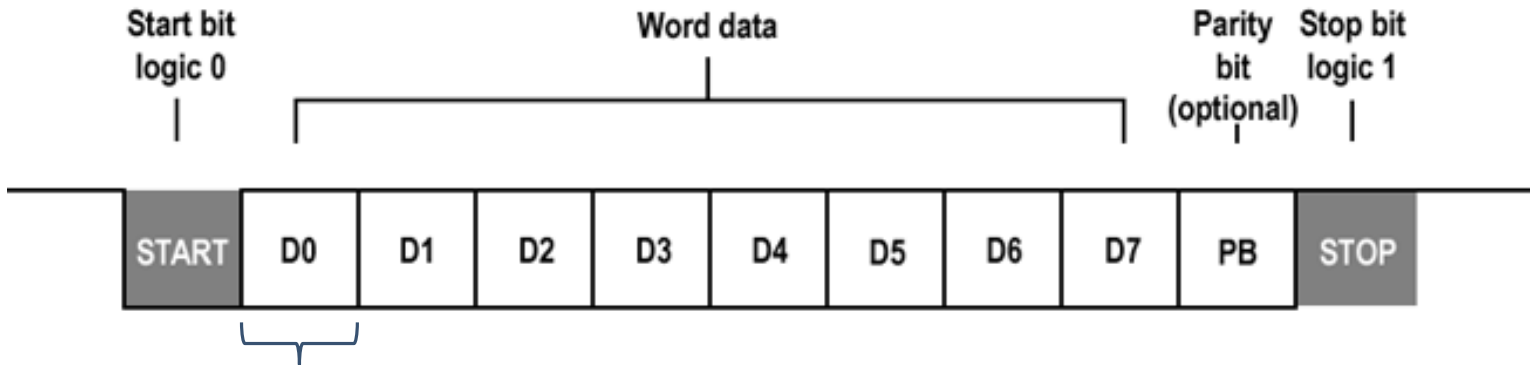
UART 개요

- UART 통신 방식



UART 개요

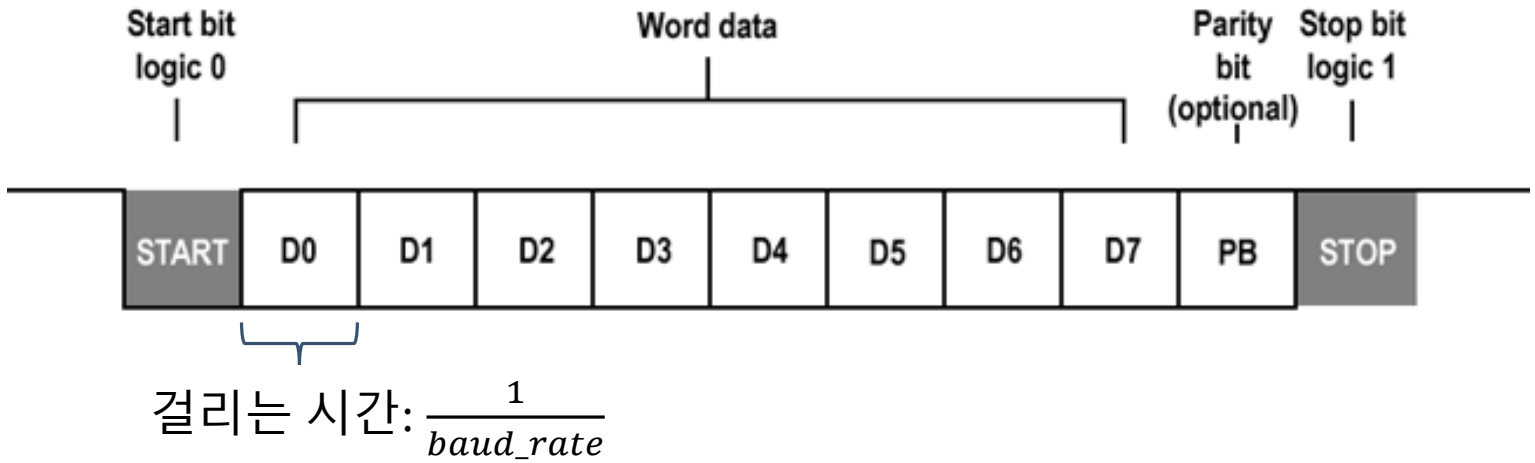
- 하드웨어 설계 시 고려 사항



폭의 길이는?

UART 개요

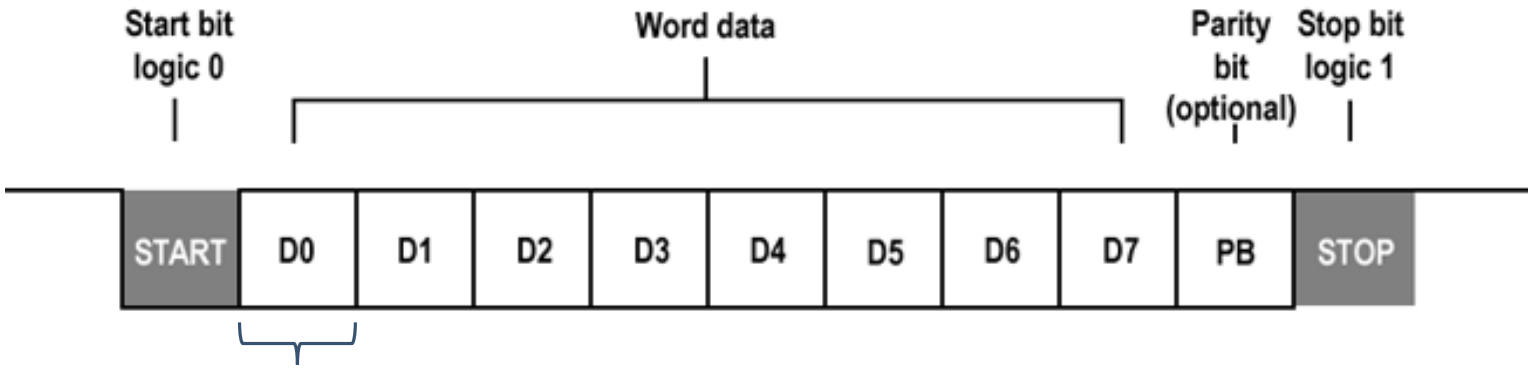
- 하드웨어 설계 시 고려 사항



  - Baud rate
    - 초당 전송하는 symbol의 수
    - 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, … 사용
    - UART에서는 1 baud = 1 bit이므로 baud rate = bps(bits per second)

UART 개요

- 하드웨어 설계 시 고려 사항



소모 클럭 사이클 수: $\frac{1}{baud\_rate} \div clock\_period = \frac{clock\_frequency}{baud\_rate}$

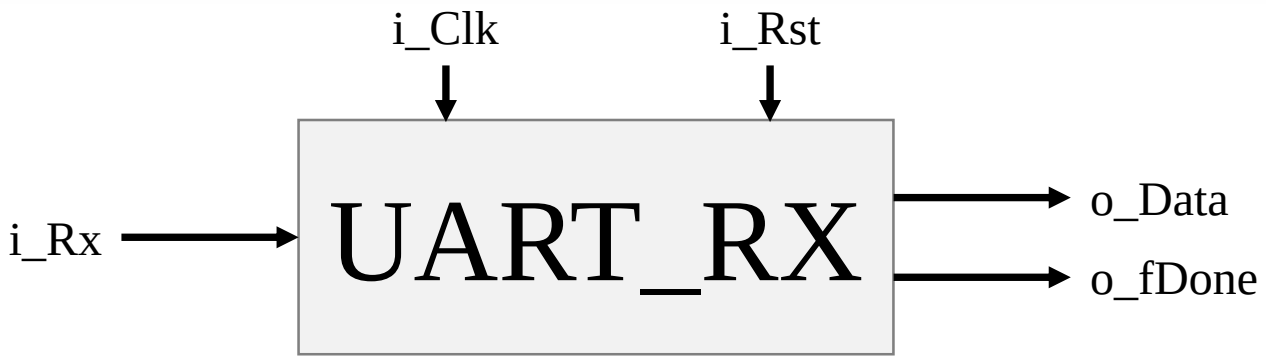
UART 개요

- 1비트 전송에 필요한 clock cycle 수
  - 계산: $\left\lceil \frac{clock\_frequency}{baud\_rate} \right\rceil$

	1200	2400	4800	9600	14400	19200	38400	56000	57600	115200
100 MHz	83334	41667	20834	10417	6945	5209	2605	1786	1737	869
80 MHz	66667	33334	16667	8334	5556	4167	2084	1429	1389	695
50 MHz	41667	20834	10417	5209	3473	2605	1303	893	869	435
40 MHz	33334	16667	8334	4167	2778	2084	1042	715	695	348
25 MHz	20834	10417	5209	2605	1737	1303	652	447	435	218
20 MHz	16667	8334	4167	2084	1389	1042	521	358	348	174
16 MHz	13334	6667	3334	1667	1112	834	417	286	278	139
8 MHz	6667	3334	1667	834	556	417	209	143	139	70
5 MHz	4167	2084	1042	521	348	261	131	90	87	44
2 MHz	1667	834	417	209	139	105	53	36	35	18
1 MHz	834	417	209	105	70	53	27	18	18	9

하드웨어 설계 – Rx 모듈

- 입력과 출력

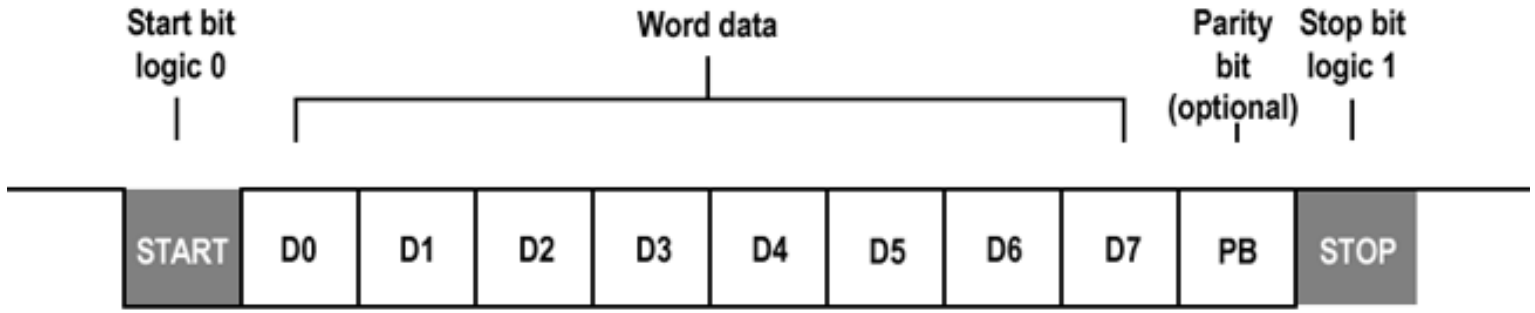


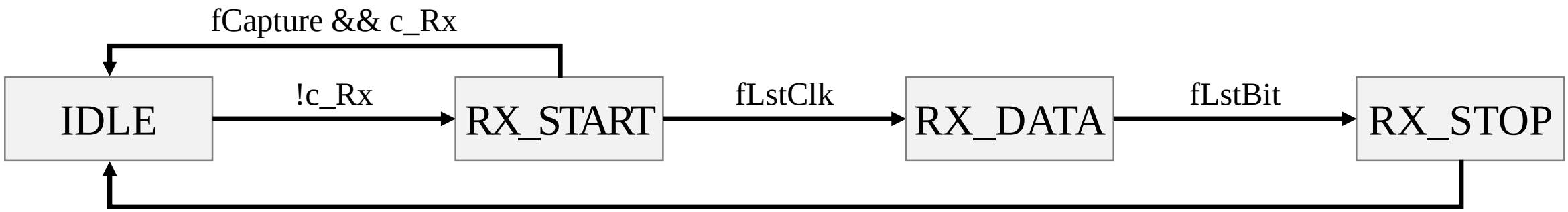
이름	구분		bits	설명
i_Clk	입력	기본	1	positive clock 신호
i_Rst			1	negative reset 신호(0: reset, 1: 동작)
i_Rx		데이터	1	UART Rx 신호
o_fDone	출력	상태	1	수신 완료
o_Data		데이터	8	수신한 데이터

하드웨어 설계 – Rx모듈

- CYCLES_PER_BIT = CLK_FREQ / BAUD
- fLstClk = c_ClkCnt == CYCLES_PER_BIT
- fCapture = c_ClkCnt == CYCLES_PER_BIT/2
- fLstBit = fLstClk && &c_BitCnt

- FSM

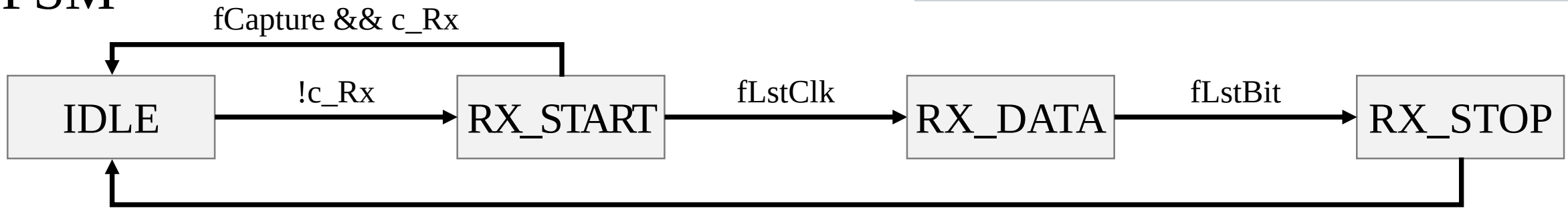




하드웨어 설계 – Rx모듈

- CYCLES_PER_BIT = CLK_FREQ / BAUD
- fLstClk = c_ClkCnt == CYCLES_PER_BIT
- fCapture = c_ClkCnt == CYCLES_PER_BIT/2
- fLstBit = fLstClk && &c_BitCnt

- FSM



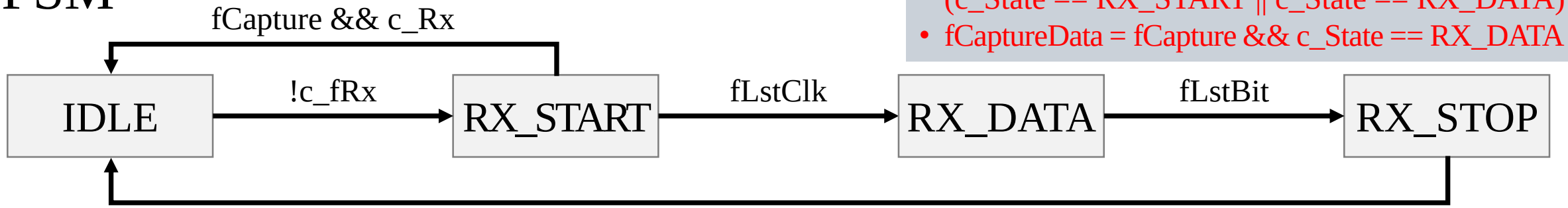
State	Regs.	bits	IDLE	RX_START	RX_DATA	RX_STOP
Regs.	n_Rx	1	i_Rx			
	n_ClkCnt	8	0	fLstClk ? 0 : c_ClkCnt + 1		X
	n_BitCnt	3	X	0	fLstClk ? c_BitCnt + 1 : c_BitCnt	X
	n_Data	8	X	X	fCapture ? {c_Rx, c_Data[7:1]} : c_Data	c_Data
Output	o_fDone	1	0	0	0	1
	o_Data	8	0	0	0	c_Data

- o_fDone = c_State == RX_STOP
- o_Data = o_fDone ? c_Data : 0

하드웨어 설계 – Rx모듈

- fLstClk = c_ClkCnt == CYCLES_PER_BIT
- fCapture = c_ClkCnt == CYCLES_PER_BIT/2
- fLstBit = fLstClk && &c_BitCnt
- fIncClktCnt = !fLstClk && (c_State == RX_START || c_State == RX_DATA)
- fCaptureData = fCapture && c_State == RX_DATA

- FSM



State	Regs.	bits	IDLE	RX_START	RX_DATA	RX_STOP
Regs.	n_Rx	1	i_Rx			
	n_ClkCnt	8	0	fLstClk ? 0 : c_ClkCnt + 1		X
	n_BitCnt	3	X	0	fLstClk ? c_BitCnt + 1 : c_BitCnt	X
	n_Data	8	X	X	fCapture ? {c_Rx, c_Data[7:1]} : c_Data	C_Data

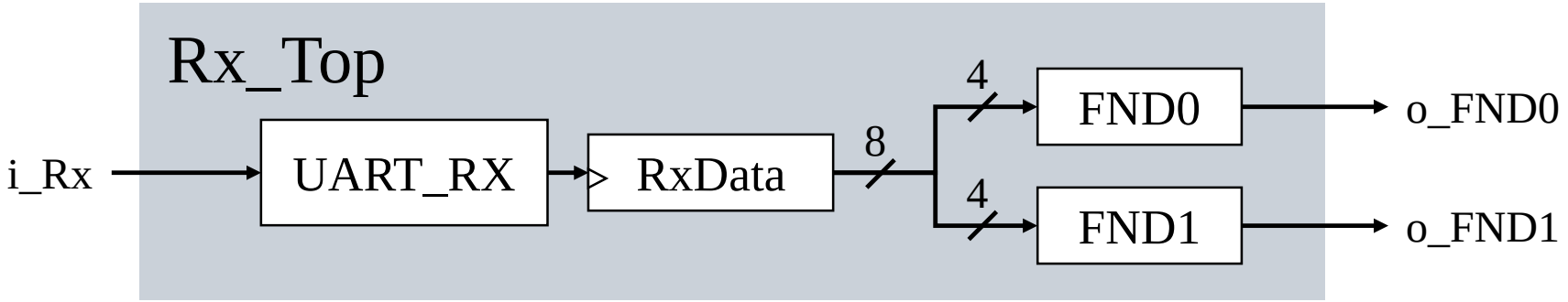
- n_Rx = i_Rx
- n_ClkCnt = fIncClkCnt ? C_ClkCnt + 1 : 0
- n_BitCnt = fRxData ? fLstClk ? c_BitCnt + 1 : c_BitCnt : 0
- n_Data = fCaptureData ? {c_Rx, c_Data[7:1]} : c_Data

하드웨어 설계 – Rx모듈

- 실습 - 명세
  - 입력: i_Rx(1)
  - 출력: o_FND0(7), o_FND1(7)
  - 모듈 이름: Rx_Top
  - 동작
    - 1바이트를 전송하면 4비트씩의 값을 FND에 표시함
      - 예) 0x39을 전송하면 3, 9가 FND에 표시됨
  - 사용할 파일
    - 첫 번째 시뮬레이션: tb_Rx, UART_RX
    - 두 번째 시뮬레이션: tb_Rx_Top, UART_RX, Rx_Top, FND
    - FPGA: UART_RX, Rx_Top, FND

하드웨어 설계 – Rx모듈

- 실습 - 구조 설계



하드웨어 설계 – Rx모듈

- 실습 – FPGA 연결

GPIO0

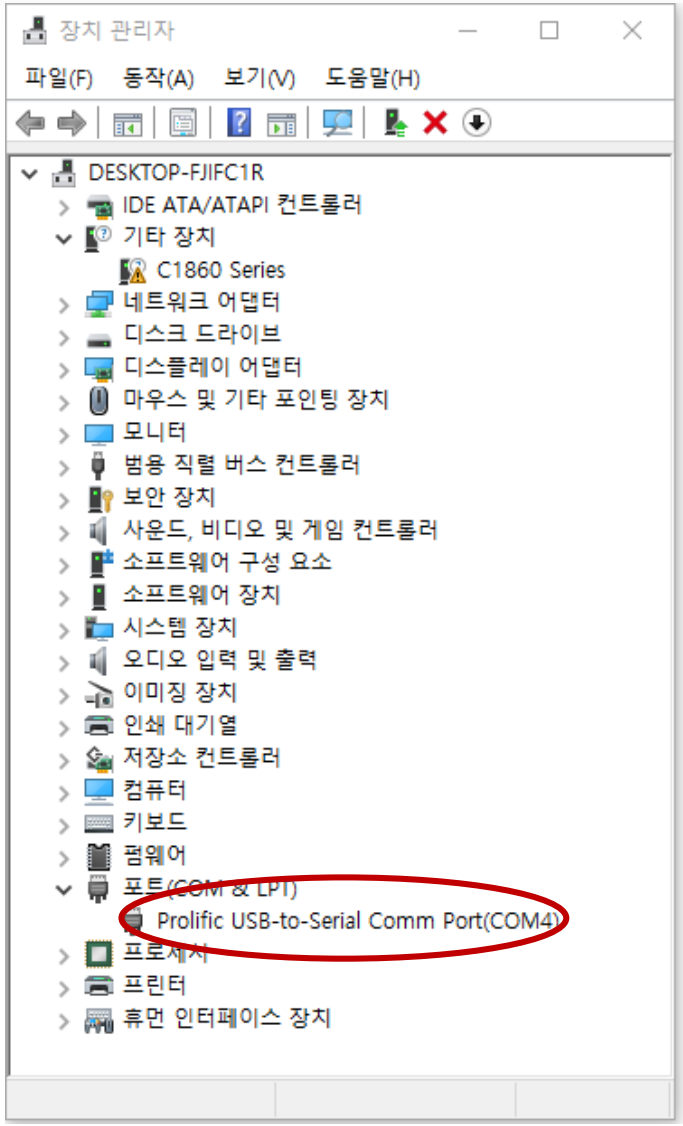
0	1
2	3
4	5
6	7
8	9
VCC	GND
10	11
12	13
14	15
16	17
18	19
20	21
22	23
24	25
VCC	GND
26	27
28	29
30	31
32	33
34	35

GPIO1

0	1
2	3
4	5
6	7
8	9
VCC	GND
10	11
12	13
14	15
16	17
18	19
20	21
22	23
24	25
VCC	GND
26	27
28	29
30	31
32	33
34	35

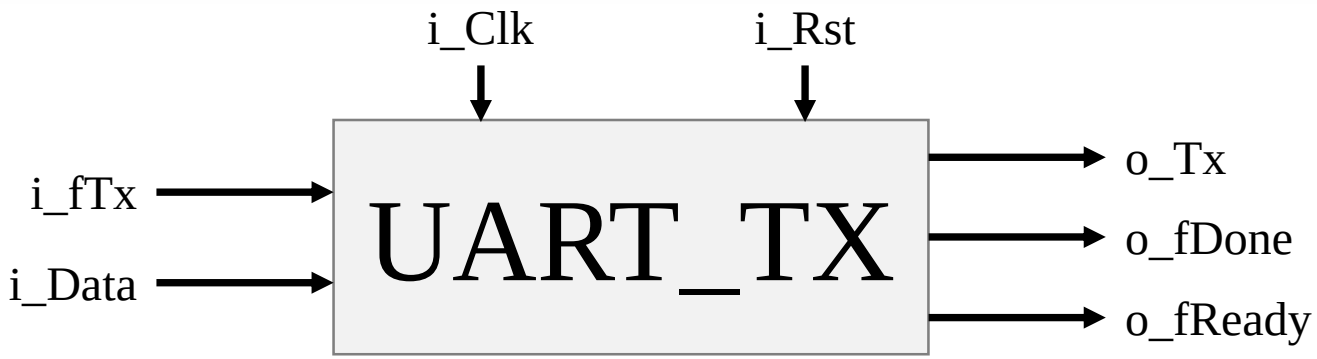
하드웨어 설계 – Rx모듈

- 실습 – UART 프로그램
  - 장치 관리자에서 COM 포트 번호 확인
  - 미인식 시 드라이버 설치



하드웨어 설계 – Tx 모듈

- 입력과 출력

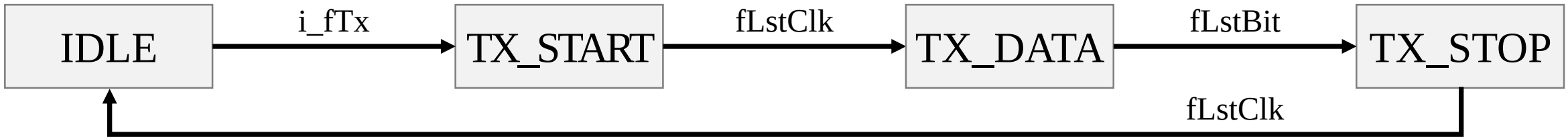


이름	구분		bits	설명
i_Clk	입력	기본	1	positive clock 신호
i_Rst			1	negative reset 신호(0: reset, 1: 동작)
i_fTx		제어	1	송신 시작 신호
i_Data		데이터	8	송신할 데이터
o_fDone	출력	상태	1	송신 완료
o_fReady			1	송신 가능 상태
o_Tx		데이터	1	UART Tx 신호

하드웨어 설계 – Tx 모듈

- CYCLES_PER_BIT = CLK_FREQ / BAUD
- fLstClk = c_ClkCnt == CYCLES_PER_BIT
- fLstBit = fLstClk && &c_BitCnt

FSM

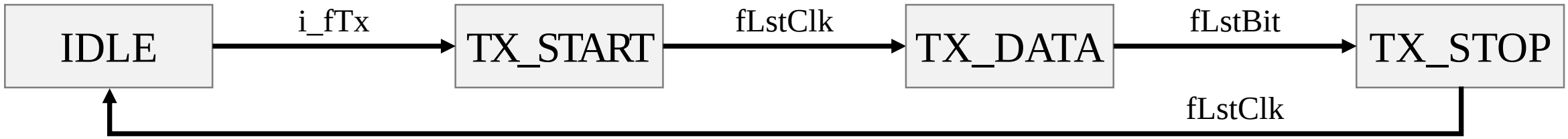


State	Regs.	bits	IDLE		TX_START	TX_DATA	TX_STOP
			i_fTx = 0	i_fTx = 1			
Regs.	n_ClkCnt	8	0		fLstClk ? 0 : c_ClkCnt + 1		
	n_BitCnt	3	X		0	fLstClk ? c_BitCnt + 1 : c_BitCnt	X
	n_Data	8	X	i_Data	c_Data	fLstClk ? c_Data[7:1] : c_Data	X
Output	o_fDone	1	0		0	0	fLstClk
	o_fReady	1	1		0	0	0
	o_Tx	1	1		0	c_Data[0]	1

하드웨어 설계 – Tx 모듈

- CYCLES_PER_BIT = CLK_FREQ / BAUD
- fLstClk = c_ClkCnt == CYCLES_PER_BIT
- fLstBit = fLstClk && &c_BitCnt

FSM

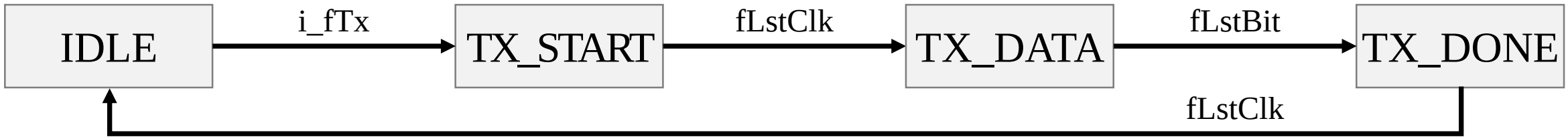


State	Regs.	bits	IDLE		TX_START	TX_DATA	TX_STOP
			i_fTx = 0	i_fTx = 1			
Regs.	n_ClkCnt	8	0		fLstClk ? 0 : c_ClkCnt + 1		
	n_BitCnt	3	X		0	fLstClk ? c_BitCnt + 1 : c_BitCnt	X
	n_Data	9	{8'hXX, 1'b1}	{i_Data, 1'b0}	fLstClk ? {1'b1, c_Data[8:1]} : c_Data		{8'hXX, 1'b1}
Output	o_fDone	1	0		0	0	fLstClk
	o_fReady	1	1		0	0	0
	o_Tx	1	c_Data[0]				

하드웨어 설계 – Tx 모듈

- CYCLES_PER_BIT = CLK_FREQ / BAUD
- fLstClk = c_ClkCnt == CYCLES_PER_BIT
- fLstBit = fLstClk && &c_BitCnt

- FSM



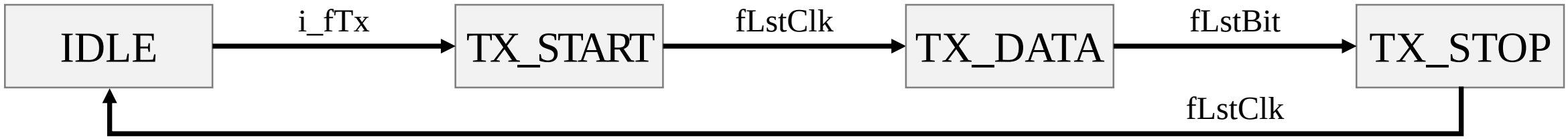
State	Regs.	bits	IDLE		TX_START	TX_DATA	TX_STOP
			i_fTx = 0	i_fTx = 1			
Output	o_fDone	1	0		0	0	fLstClk
	o_fReady	1	1		0	0	0
	o_Tx	1	c_Data[0]				

- o_fDone = c_State == TX_STOP && fLstClk
- o_fReady = c_State == IDLE
- o_Tx = c_Data[0]

하드웨어 설계 – Tx 모듈

- CYCLES_PER_BIT = CLK_FREQ / BAUD
- fLstClk = c_ClkCnt == CYCLES_PER_BIT
- fLstBit = fLstClk && &c_BitCnt
- fIncClktCnt = c_State != IDLE && !fLstClk

FSM



State	Regs.	bits	IDLE		TX_START	TX_DATA	TX_STOP
			i_fTx = 0	i_fTx = 1			
Regs.	n_ClkCnt	8	0		fLstClk ? 0 : c_ClkCnt + 1		
	n_BitCnt	3	0		0	fLstClk ? c_BitCnt + 1 : c_BitCnt	0
	n_Data	9	{8'hXX, 1'b1}	{i_Data, 1'b0}	fLstClk ? {1'b1, c_Data[8:1]} : c_Data		{8'hXX, 1'b1}

- n_ClkCnt = fIncClkCnt ? C_ClkCnt + 1 : 0
- n_BitCnt = fTxData ? fLstClk ? c_BitCnt + 1 : c_BitCnt : 0
- n_Data = fIdle && i_fTx ? {i_Data, 1'b0} : fLstClk ? {1'b1, c_Data[8:1]} : c_Data

하드웨어 설계 – Tx 모듈

- 실습 - 명세
  - 입력: i_Rx(1), i_Push(4)
  - 출력: o_Tx(1), o_FND0(7), o_FND1(7)
  - 모듈 이름: UART_Top
  - 동작
    - Push을 누르면 누른 위치(0~3)을 전송함
    - 1바이트를 전송하면 4비트씩의 값을 FND에 표시함
      - 예) 0x39을 전송하면 3, 9가 FND에 표시됨

하드웨어 설계 – Tx 모듈

- 실습 - 구조 설계