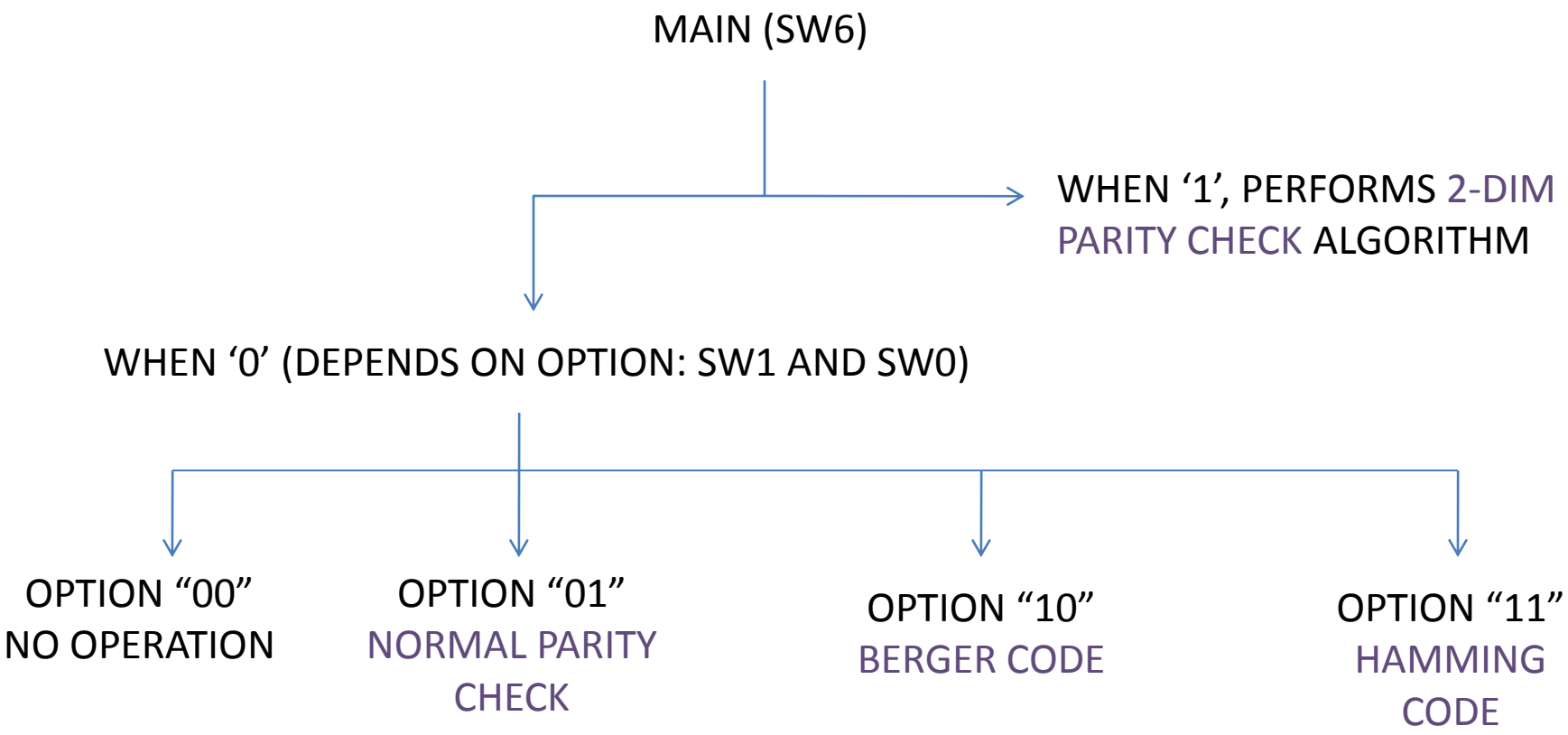
ERROR DETECTION AND CORRECTION TECHNIQUES - IMPLEMENTATION IN FPGA

By

AARON T IMMANUEL
217-74-907

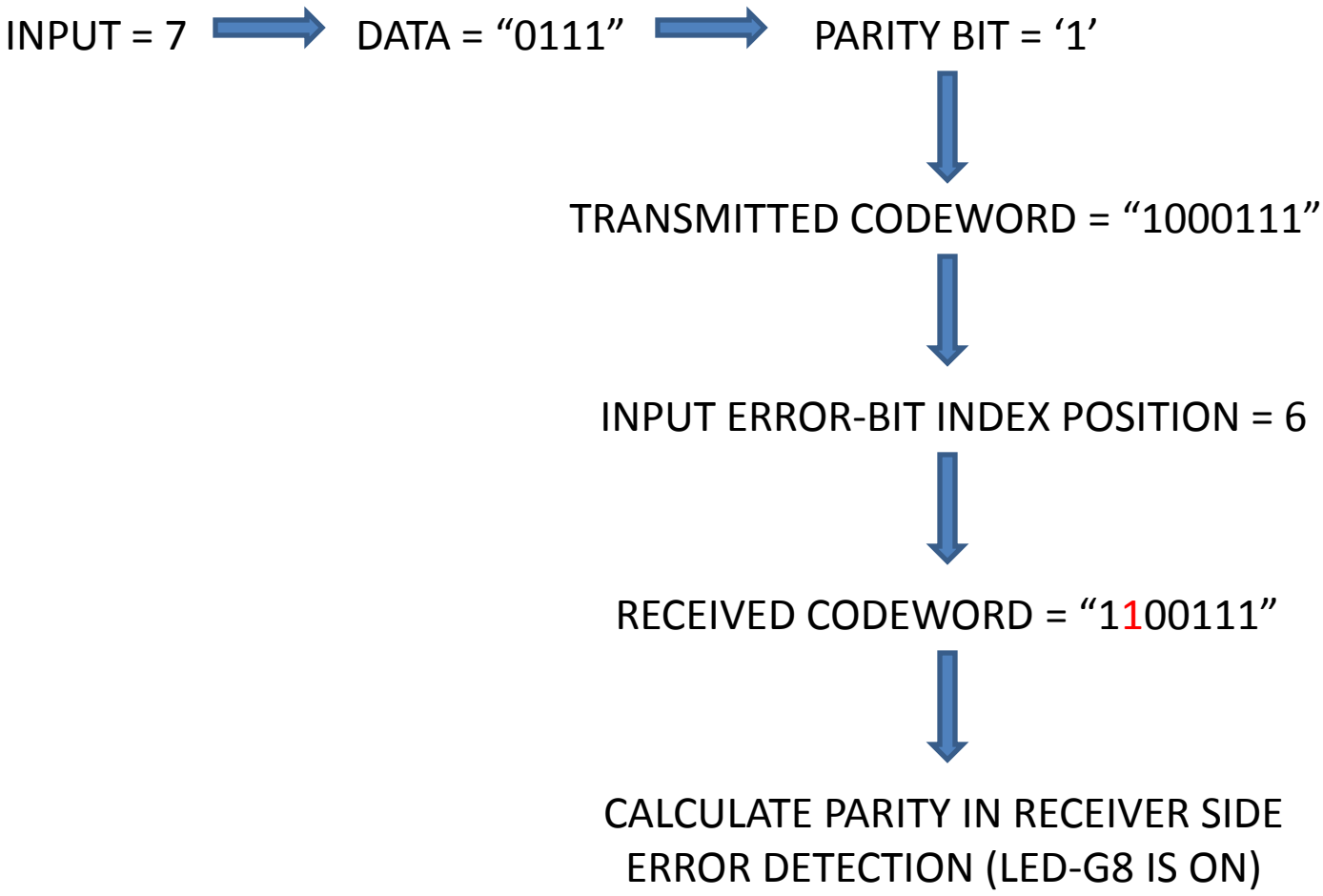
Review: Prof. Mustafa Torun

PROJECT OUTLINE



Another project: 4-DIMENSIONAL PARITY CHECK algorithm using RAM implementation. User can input 2 error-bit index position using (SW4, SW3, SW2, SW1, SW0) and (SW14, SW13, SW12, SW11, SW10).

NORMAL PARITY CHECK EXAMPLE



BERGER CODE EXAMPLE

INPUT = 9 → DATA = “1001” → FIND No. OF ZEROES IN DATA = “010”

↓

TRANSMITTED CODEWORD = “0101001”

↓

INPUT ERROR-BIT INDEX POSITION = 7

↓

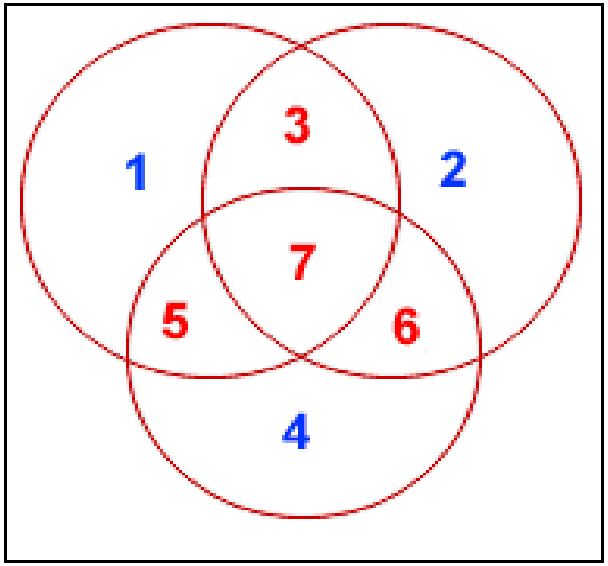
RECEIVED CODEWORD = “1101001”

↓

CALCULATE No. OF ZEROES IN RECEIVED CODEWORD’S DATA
ERROR DETECTION (LED-G8 IS ON)

HAMMING CODE EXAMPLE

INPUT = A → DATA = “1010” → PARITY IN (1,2,4) AND DATA IN (3, 5, 6, 7)



PARITY 1 = ‘0’ with PARITY 2 = ‘1’ and PARITY 4 = ‘0’

TRANSMITTED CODEWORD = “1010010”

INPUT ERROR-BIT INDEX POSITION = 1

RECEIVED CODEWORD = “1010011”

CALCULATE PARITY BITS IN RECEIVER SIDE AND ERROR-BIT INDEX POSITION (USING LEDR4 – LEDR2) IS “001”

Courtesy: http://www.ee.unb.ca/tervo/ee4253/hamming.shtml

2-DIMENSIONAL PARITY CHECK

INPUT DATA (1, 2, 3, 4) → FIND PARITY BY ADDING EACH ROW AND COLUMN

↓

TRANSMITTED ARRAY <= "1-2-3-3-4-7-4-6"

↓

INPUT ERROR INDEX POSITION = 5

↓

RECEIVED ARRAY <= "1-2-3-3-100-7-4-6"

↓

MDPC_LED <= "0101" → ERROR DETECTION BY HEX3

1	2	3
3	4	7
4	6	

Courtesy: http://en.wikipedia.org/wiki/Multidimensional_parity-check_code

4-DIMENSIONAL PARITY CHECK

ACCESS DATA STORED IN RAM
(SHOWN IN RED)

45	12	60	2	119
47	99	14	25	185
1	87	40	33	161
72	63	91	3	229
165	261	205	63	

FIND PARITY BY ADDING EACH ROW AND COLUMN
(SHOWN IN BLUE)

TRANSMITTED ARRAY

INPUT 2 ERROR INDEX POSITIONS
ASSUMPTIONS: NO SAME POSITION IN ERROR
NO PARITY IN ERROR (5, 10, 15, 20, 21, 22, 23, 24)

RECEIVED ARRAY with 1000 and 2000 AT ERROR POSITIONS

ERROR_BIT_LED, (HEX7, HEX6) AND (HEX5, HEX4)
SHOW (ROW-COLUMN) ERROR DETECTION

CORRECT DATA VALUES IN
(HEX3, HEX2) AND (HEX1, HEX0)

Thank you