Washington University in St.Louis

McKelvey School of Engineering

Spring Semester 2023

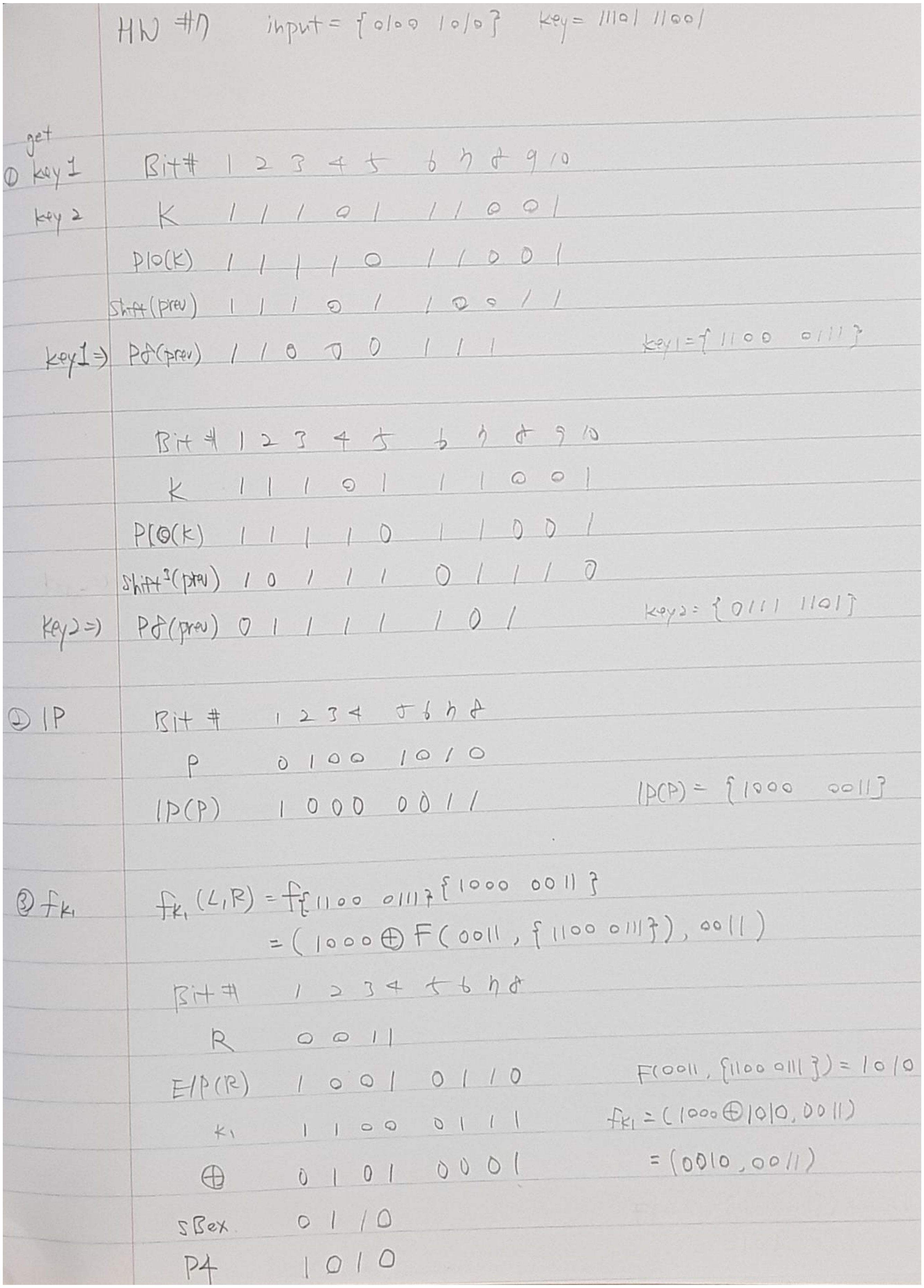
CSE463M: Digital Integrated Circuit Design and Architecture

Homework #7

1. **Using the simplified version of DES encryption algorithm for the 8-bit input sequence Input = 01001010 and 10-bit Input key k = 1110111001:**

   a. **Determine by hand an encrypted 8-bit output sequence Output. Show all your work steps in your report.**
   b. **Write a piece of code using preferred language (C, Python, etc.) and confirm your result from the part a. Submit your piece of code and show your simulation result screen shoot in your report.**

1.a



HW #7 input = {0100 1010} key = 11101 11001

get

① key1

key2

	Bit #	1	2	3	4	5	6	7	8	9	10
	K	1	1	1	0	1	1	1	0	0	1
	P10(K)	1	1	1	1	0	1	1	0	0	1
	Shift(prev)	1	1	1	0	1	1	0	0	1	1
key1 =>	P8(prev)	1	1	0	0	0	1	1	1		

key1 = {1100 0111}

	Bit #	1	2	3	4	5	6	7	8	9	10
	K	1	1	1	0	1	1	1	0	0	1
	P10(K)	1	1	1	1	0	1	1	0	0	1
	Shift^3(prev)	1	0	1	1	1	0	1	1	1	0
key2 =>	P8(prev)	0	1	1	1	1	1	0	1		

key2 = {0111 1101}

② IP

Bit #	1 2 3 4 5 6 7 8
P	0 1 0 0 1 0 1 0
IP(P)	1 0 0 0 0 0 1 1

IP(P) = {1000 0011}

③ f_{K_1}

$f_{K_1}(L,R) = f_{\{1100\ 0111\}}\{1000\ 0011\}$

$= (1000 \oplus F(0011, \{1100\ 0111\}), 0011)$

Bit #	1 2 3 4 5 6 7 8
R	0 0 1 1
E/P(R)	1 0 0 1 0 1 1 0
k_1	1 1 0 0 0 1 1 1
$\oplus$	0 1 0 1 0 0 0 1
SBox	0 1 1 0
P4	1 0 1 0

F(0011, {1100 0111}) = 1010

$f_{K_1} = (1000 \oplus 1010, 0011)$

$= (0010, 0011)$

④ SN

$$\begin{tikzcd} (0010,\ 0011) \arrow[d] \\ (0011,\ 0010) \end{tikzcd}$$

⑤ f_{K2}

$F\{0111\ 1101\}\{0011\ 0010\}$

$= (0011 \oplus F(0010, \{0111\ 1101\}), 0010)$

Bit #	1	2	3	4	5	6	7	8
R	0	0	1	0				
E/P(R)	0	0	0	1	0	1	0	0
K2	0	1	1	1	1	1	0	1
$\oplus$	0	1	1	0	1	0	0	1
SBox	1	0	1	0				
P4	0	0	1	1				

$F(1101, \{0111\ 1101\}) = 0011$

$f_{K_2} = (0011 \oplus 0011, 0010)$

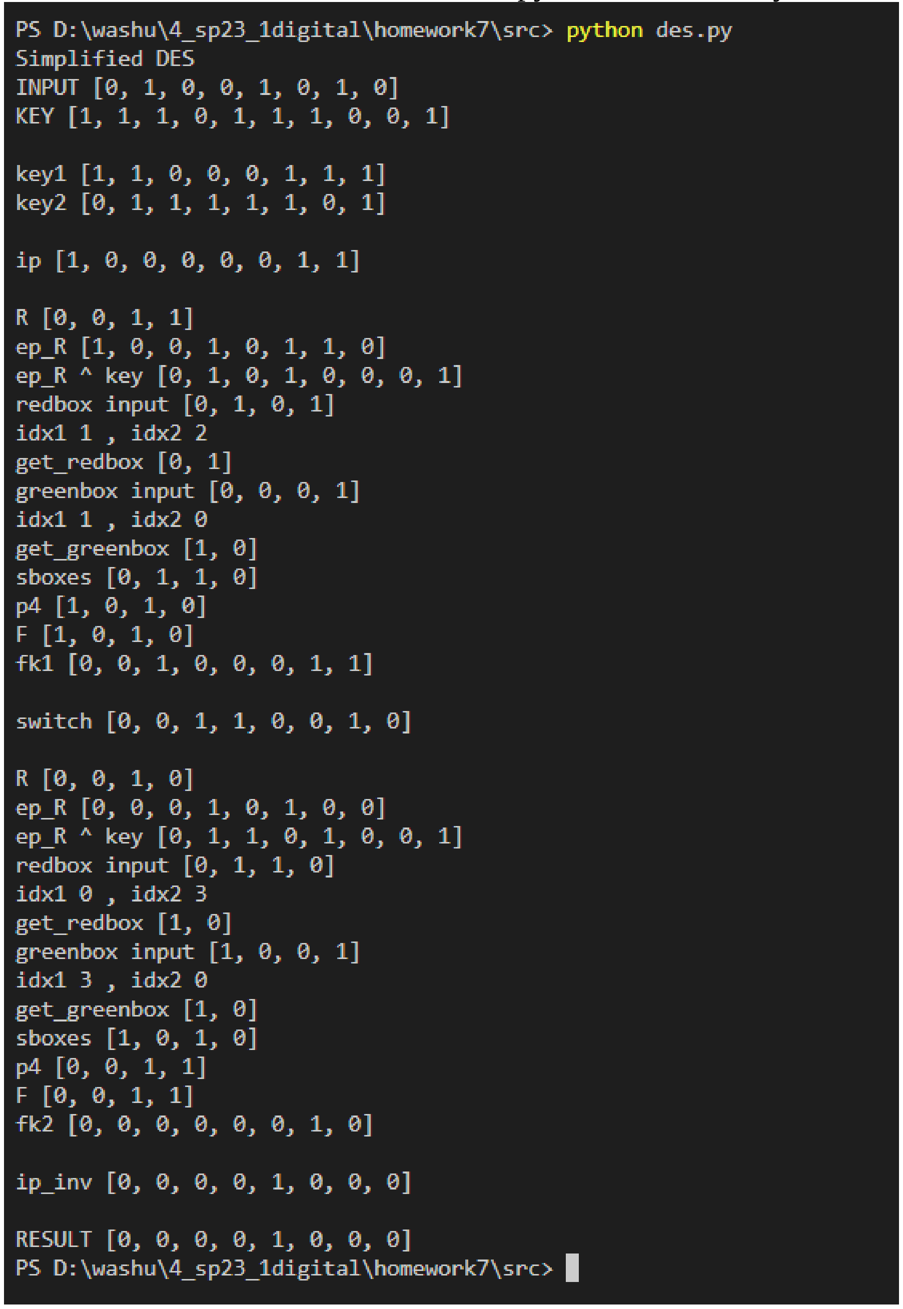
$= (0000, 0010)$

⑥ IP^{-1}

Bit #	1	2	3	4	5	6	7	8
R, L	0	0	0	0	0	0	1	0
IP^{-1}	0	0	0	0	1	0	0	0

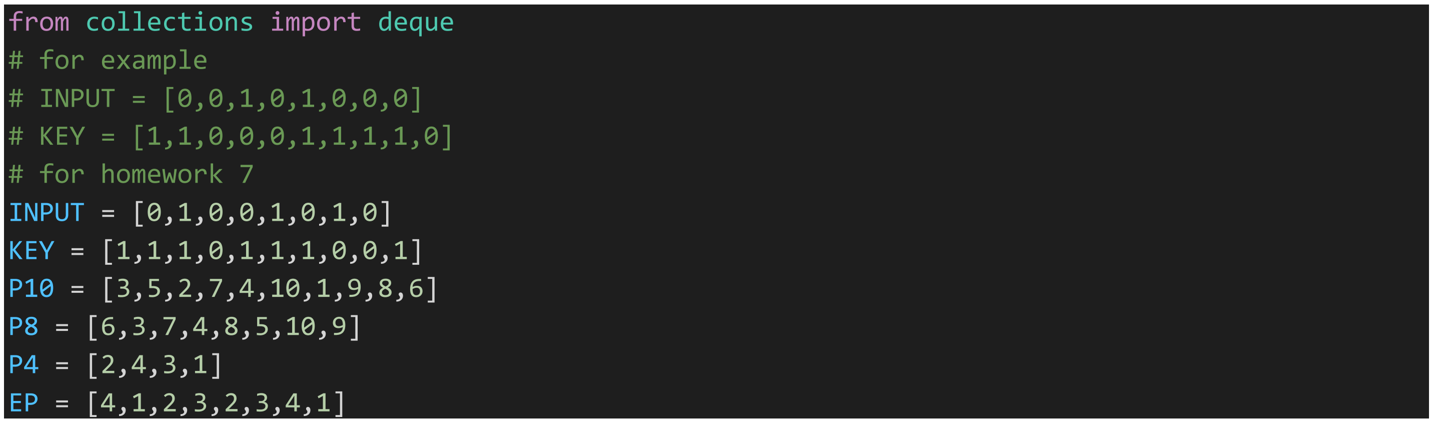
Result 0000 1000

1.b
+ Here is the result screenshot. And the python result and my hand solution have the same result value.



```
PS D:\washu\4_sp23_1digital\homework7\src> python des.py
Simplified DES
INPUT [0, 1, 0, 0, 1, 0, 1, 0]
KEY [1, 1, 1, 0, 1, 1, 1, 0, 0, 1]

key1 [1, 1, 0, 0, 0, 1, 1, 1]
key2 [0, 1, 1, 1, 1, 1, 0, 1]

ip [1, 0, 0, 0, 0, 0, 1, 1]

R [0, 0, 1, 1]
ep_R [1, 0, 0, 1, 0, 1, 1, 0]
ep_R ^ key [0, 1, 0, 1, 0, 0, 0, 1]
redbox input [0, 1, 0, 1]
idx1 1 , idx2 2
get_redbox [0, 1]
greenbox input [0, 0, 0, 1]
idx1 1 , idx2 0
get_greenbox [1, 0]
sboxes [0, 1, 1, 0]
p4 [1, 0, 1, 0]
F [1, 0, 1, 0]
fk1 [0, 0, 1, 0, 0, 0, 1, 1]

switch [0, 0, 1, 1, 0, 0, 1, 0]

R [0, 0, 1, 0]
ep_R [0, 0, 0, 1, 0, 1, 0, 0]
ep_R ^ key [0, 1, 1, 0, 1, 0, 0, 1]
redbox input [0, 1, 1, 0]
idx1 0 , idx2 3
get_redbox [1, 0]
greenbox input [1, 0, 0, 1]
idx1 3 , idx2 0
get_greenbox [1, 0]
sboxes [1, 0, 1, 0]
p4 [0, 0, 1, 1]
F [0, 0, 1, 1]
fk2 [0, 0, 0, 0, 0, 0, 1, 0]

ip_inv [0, 0, 0, 0, 1, 0, 0, 0]

RESULT [0, 0, 0, 0, 1, 0, 0, 0]
PS D:\washu\4_sp23_1digital\homework7\src>
```

+ Here is my python source code.



```
from collections import deque
# for example
# INPUT = [0,0,1,0,1,0,0,0]
# KEY = [1,1,0,0,0,1,1,1,1,0]
# for homework 7
INPUT = [0,1,0,0,1,0,1,0]
KEY = [1,1,1,0,1,1,1,0,0,1]
P10 = [3,5,2,7,4,10,1,9,8,6]
P8 = [6,3,7,4,8,5,10,9]
P4 = [2,4,3,1]
EP = [4,1,2,3,2,3,4,1]
```

```python
IP = [2,6,3,1,4,8,5,7]
IP_INV = [4,1,3,5,7,2,8,6]
RED_BOX = [
    [[0,1],[0,0],[1,1],[1,0]],
    [[1,1],[1,0],[0,1],[0,0]],
    [[0,0],[1,0],[0,1],[1,1]],
    [[1,1],[0,1],[1,1],[1,0]]
]
GREEN_BOX = [
    [[0,0],[0,1],[1,0],[1,1]],
    [[1,0],[0,0],[0,1],[1,1]],
    [[1,1],[0,0],[0,1],[0,0]],
    [[1,0],[0,1],[0,0],[1,1]]
 ]

def p10(input):
    return_value = []
    for i in P10:
        return_value.append(input[i-1])
    return return_value


def p8(input):
    return_value = []
    for i in P8:
        return_value.append(input[i-1])
    return return_value


def p4(input):
    return_value = []
    for i in P4:
        return_value.append(input[i-1])
    return return_value


def exp(input):
    return_value = []
    for i in EP:
        return_value.append(input[i-1])
    return return_value


# direction: 1 for forward, -1 for backward
def get_ip(input, direction):
    return_value = []
    input_seq = IP if direction == 1 else IP_INV
    for i in input_seq:
        return_value.append(input[i-1])
    return return_value
```

```python
# direction: plus for right, minus for left
def rotate_each_half(input, direction):
    first_half = deque(input[:len(input)//2])
    last_half = deque(input[len(input)//2:])
    first_half.rotate(direction)
    last_half.rotate(direction)

    return list(first_half) + list(last_half)


# type: 1 for key1 2 for key2
def get_key(input, type):
    key = []
    key = p10(input)
    key = rotate_each_half(key, -1 if type == 1 else -3)
    key = p8(key)

    return key


# input: 4 bits
# key: 8 bits
def get_f(input, key):
    # return [0,0,0,1]
    print("R", input)
    ep_r = exp(input)
    print("ep_R", ep_r)
    ep_xor = []
    for i in range(len(ep_r)):
        ep_xor.append(ep_r[i] ^ key[i])

    print("ep_R ^ key", ep_xor)

    sboxes = get_redbox(ep_xor[:len(ep_xor)//2]) + get_greenbox(ep_xor[len(ep_xor)//2:])
    print("sboxes", sboxes)
    p4(sboxes)
    print("p4", p4(sboxes))
    return p4(sboxes)


def get_box_index(a,b):
    if a == 0 and b == 0:
        return 0
    elif a == 0 and b == 1:
        return 1
    elif a == 1 and b == 0:
        return 2
    else:
        return 3
```

```python
# input: 8 bits
def get_redbox(input):
    print("redbox input", input)
    idx1 = get_box_index(input[0], input[3])
    idx2 = get_box_index(input[1], input[2])
    print("idx1", idx1, ", idx2", idx2)

    red = RED_BOX[idx1][idx2]
    print("get_redbox", red)

    return red


# input: 8 bits
def get_greenbox(input):
    print("greenbox input", input)
    idx1 = get_box_index(input[0], input[3])
    idx2 = get_box_index(input[1], input[2])
    print("idx1", idx1, ", idx2", idx2)

    green = GREEN_BOX[idx1][idx2]
    print("get_greenbox", green)

    return green


def get_fk(key, input):
    first_half = input[:len(input)//2]
    last_half = input[len(input)//2:]

    f = get_f(last_half, key)
    print("F", f)
    new_half = []
    for i in range(len(f)):
        new_half.append(first_half[i] ^ f[i])

    return new_half + last_half


def get_switch(input):
    return input[len(input)//2:] + input[:len(input)//2]


def main():
    print("Simplified DES")
    print("INPUT", INPUT)
    print("KEY", KEY)
    print("")

    key1 = get_key(KEY, 1)
    print("key1", key1)
    key2 = get_key(KEY, 2)
    print("key2", key2)
    print("")

    ip = get_ip(INPUT, 1)
    print("ip", ip)
    print("")

    fk1 = get_fk(key1, ip)
    print("fk1", fk1)
    print("")

    switch = get_switch(fk1)
    print("switch", switch)
    print("")

    fk2 = get_fk(key2, switch)
    print("fk2", fk2)
    print("")

    ip_inv = get_ip(fk2, -1)
    print("ip_inv", ip_inv)
    print("")

    print("RESULT", ip_inv)

if __name__ == "__main__":
    main()
```