Maggie Cao

Digital Logic Design Final Project: *The Simple Enigma*

System's Abstract:

My Enigma machine is taken from the industrialization of an assembly line, where each step does not require memory but a sudden and even flow of logic to create the encryption. The system is organized into three parts that are titled to separate the level of scrambling. The beginning categorizes the setting and the key in order to produce a positive difference. The difference, representing a number, is traveled into an indirect logic that equates the number into a new letter. My original design is to not repeat the counting between the DIP switches and the letter pressed during the second encryption. Instead of repeating the simple mathematics of the "beginning" again in the presence of the second switch, the new letter will be scrambled into the case of a random chart. This second-level number will then be finally passed into the wires that breathe in the infinite shifting movement of the most significant to the least significant bits. This means that the letter pressed for the same combination between the two settings, which decrypts the message, will vary because the system will reset when a new key is pressed. However, it is not possible to vary the setting so suddenly because it will make the system too random to be controlled. The ending is supposed to scramble the counting between the numbers meaning it does not count from 0…1…2…3…4…5…6…7…, but at random numbers such as 5…4…1…2…0…7…6…3, causing an additional coding element in the adder and the comparator. I want to stick to the original enigma, and the coding chart is long enough is so I followed the process of the Enigma working towards one way.

Design: "The Beginning"

Each button represents the keys from eight alphabets, A-H, which initiates the JK flip flops through the OR logic gates. Since A-H is essentially the numbers 0-7 in chronological order, the outcome of the flip flops should be 000 when button A is pressed. When the numbers match its letter, the three bits are transferred into the series of multiplexers, creating the corresponding letters A-H onto the LED display.

A=000

Key_A	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0
1	0	1	0	1	0	1

B=001

Key_B	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0
1	0	1	0	1	1	0

C=010

Key_C	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0
1	0	1	1	0	0	1

D=011

Key_D	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0
1	0	1	1	0	1	0

E=100

Key_E	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0
1	1	0	0	1	0	1

F=101

Key_F	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0
1	1	0	0	1	1	0

G=110

Key_G	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0
1	1	0	1	0	0	1

H=111

Key_H	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0
1	1	0	1	0	1	0

$J_1 = Key_E + Key_F + Key_G + Key_H$

$K_1 = Key_A + Key_B + Key_C + Key_D$

$J_2 = Key_C + Key_D + Key_G + Key_H$

$K_2 = Key_A + Key_B + Key_E + Key_F$

$J_3 = Key_B + Key_D + Key_F + Key_H$

$K_3 = Key_A + Key_C + Key_E + Key_G$

There are problems with the 4-bit comparator chips because they drive CMOS chips and its driven power is too low to be consistent with TTL chips used in the circuit, causing frequent oscillations at the 555 timer and the JK flip flops. Therefore, the design of the logic is needed in order to determine times when the key is less than or greater than DIP switch 1. The equal function chip is not necessary because Less Than or Greater Than will still have either a 1 or 0 if they are the same values for each bit.

Comparator: Version Two: Current Design

I) Least Significant Bit when Switch I is 1

Key	Sw.1	<	<	>		>	=	L	G
A/0	1	<	0	1		1	1	1	0
B/1	1	=	1	1		0	0	0	0
C/0	1	<	0	1		1	1	1	0
D/1	1	=	1	1		0	0	0	0
E/0	1	<	0	1		1	1	1	0
F/1	1	=	1	1		0	0	0	0
G/0	1	<	0	1		1	1	1	0
H/1	1	=	1	1		0	0	0	0

$L = (Key \oplus Sw.1)$ $G = Key \cdot Sw.1'$

Least Significant Bit When Switch I is 0

Key	Sw.1	>	>	<		<	=	L	G
A/0	0	=	1	1		0	1	1	1
B/1	0	>	0	1		1	0	1	0
C/0	0	=	1	1		0	1	1	1
D/1	0	>	0	1		1	0	1	0
E/0	0	=	1	1		0	1	1	1
F/1	0	>	0	1		1	0	1	0
G/0	0	=	1	1		0	1	1	1
H/1	0	>	0	1		1	0	1	0

$L = (Sw.1 \oplus Key)' \oplus Key$ $G = (Sw.1 \oplus Key)'$

II) Intermediate Bit when Switch I is 1

Key	Sw.1	<	<	>		>	=	L	G
A/0	1	<	0	1		1	1	1	0
B/0	1	<	0	1		1	1	1	0
C/1	1	=	1	1		0	0	0	0
D/1	1	=	1	1		0	0	0	0
E/0	1	<	0	1		1	1	1	0
F/0	1	<	0	1		1	1	1	0
G/1	1	=	1	1		0	0	0	0
H/1	1	=	1	1		0	0	0	0

$L = (Key \oplus Sw.1)$ $G = Key \cdot Sw.1'$

Intermediate Bit when Switch I is 0

Key	Sw.1	>	>	<		<	=	L	G
A/0	0	=	1	1		0	0	1	1
B/0	0	=	1	1		0	0	1	1
C/1	0	>	0	1		1	1	1	0
D/1	0	>	0	1		1	1	1	0
E/0	0	=	1	1		0	0	1	1
F/0	0	=	1	1		0	0	1	1
G/1	0	>	0	1		1	1	1	0
H/1	0	>	0	1		1	1	1	0

$L = (Sw.1 \oplus Key)' \oplus Key$ $G = (Sw.1 \oplus Key)'$

III) Most Significant Bit when Switch is 1

Key	Sw.1	<	<	>		>	=	L	G
A/0	1	<	0	1		1	1	1	0
B/0	1	<	0	1		1	1	1	0
C/0	1	<	0	1		1	1	1	0
D/0	1	<	0	1		1	1	1	0
E/1	1	=	1	1		0	0	0	0
F/1	1	=	1	1		0	0	0	0
G/1	1	=	1	1		0	0	0	0
H/1	1	=	1	1		0	0	0	0

$L = (Key \oplus Sw.1)$ $G = Key \cdot Sw.1'$

Most Significant Bit when Switch is 0

Key	Sw.1	>	>	<		<	=	L	G
A/0	0	=	1	1		0	0	1	1
B/0	0	=	1	1		0	0	1	1
C/0	0	=	1	1		0	0	1	1
D/0	0	=	1	1		0	0	1	1
E/1	0	>	0	1		1	1	1	0
F/1	0	>	0	1		1	1	1	0
G/1	0	>	0	1		1	1	1	0
H/1	0	>	0	1		1	1	1	0

$L = (Sw.1 \oplus Key)' \oplus Key$ $G = (Sw.1 \oplus Key)'$

Comparator: Version One: Failed Design

I) Least Significant Bit when Switch I is 1

Key	Sw.1	<	<	>		>	=	L	G
A/0	1	<	0	1		1	1	1	0
B/1	1	=	1	1		0	0	0	1
C/0	1	<	0	1		1	1	1	0
D/1	1	=	1	1		0	0	0	1
E/0	1	<	0	1		1	1	1	0
F/1	1	=	1	1		0	0	0	1
G/0	1	<	0	1		1	1	1	0
H/1	1	=	1	1		0	0	0	1

$L = (Key' \oplus Sw.1)$

$G = (Key \oplus Sw.1') + Sw.1$

Least Significant Bit When Switch I is 0

Key	Sw.1	>	>	<		<	=	L	G
A/0	0	=	1	1		0	0	0	1
B/1	0	>	0	1		1	1	1	0
C/0	0	=	1	1		0	0	0	1
D/1	0	>	0	1		1	1	1	0
E/0	0	=	1	1		0	0	0	1
F/1	0	>	0	1		1	1	1	0
G/0	0	=	1	1		0	0	0	1

H/1	0	>	0	1		1	1	1	0

G= (Key ⊕Sw.1′)

II) Intermediate Bit when Switch I is 1

Key	Sw.1	<	<	>		>	=	L	G
A/0	1	<	0	1		1	1	0	1
B/0	1	<	0	1		1	1	0	1
C/1	1	=	1	1		0	0	1	0
D/1	1	=	1	1		0	0	1	0
E/0	1	<	0	1		1	1	0	1
F/0	1	<	0	1		1	1	0	1
G/1	1	=	1	1		0	0	1	0
H/1	1	=	1	1		0	0	1	0

L= (Key′ ⊕ Sw.1)

G= (Key ⊕ Sw.1′)+ Sw.1

III) Most Significant Bit when Switch is 1

Key	Sw.1	<	<	>		>	=	L	G
A/0	1	<	0	1		1	1	0	1
B/0	1	<	0	1		1	1	0	1
C/0	1	<	0	1		1	1	0	1
D/0	1	<	0	1		1	1	0	1
E/1	1	=	1	1		0	0	1	0
F/1	1	=	1	1		0	0	1	0
G/1	1	=	1	1		0	0	1	0
H/1	1	=	1	1		0	0	1	0

L= (Key′ ⊕ Sw.1)

G= (Key ⊕ Sw.1′)+ Sw.1

L= (Key′ ⊕ Sw.1)+ Sw.1′

Intermediate Bit when Switch I is 0

Key	Sw.1	>	>	<		<	=	L	G
A/0	0	=	1	1		0	0	0	1
B/0	0	=	1	1		0	0	0	1
C/1	0	>	0	1		1	1	1	0
D/1	0	>	0	1		1	1	1	0
E/0	0	=	1	1		0	0	0	1
F/0	0	=	1	1		0	0	0	1
G/1	0	>	0	1		1	1	1	0
H/1	0	>	0	1		1	1	1	0

G= (Key ⊕Sw.1′)

L= (Key′ ⊕ Sw.1)+ Sw.1′

Most Significant Bit when Switch is 0

Key	Sw.1	>	>	<		<	=	L	G
A/0	0	=	1	1		0	0	0	1
B/0	0	=	1	1		0	0	0	1
C/0	0	=	1	1		0	0	0	1
D/0	0	=	1	1		0	0	0	1
E/1	0	>	0	1		1	1	1	0
F/1	0	>	0	1		1	1	1	0
G/1	0	>	0	1		1	1	1	0
H/1	0	>	0	1		1	1	1	0

G= (Key ⊕ Sw.1′)

L= (Key ⊕ Sw.1′)+ Sw.1′

Design: One-To-One Mapping Logic: First Encryption

X=GREEN

Y=WHITE

Z=BLACK

Beginning	X	Y	Z	→	Intermediate	C	B	A
A/0	0	0	0	→	F/5	1	0	1
B/1	0	0	1	→	A/0	0	0	0
C/2	0	1	0	→	E/4	1	0	0
D/3	0	1	1	→	H/7	1	1	1
E/4	1	0	0	→	B/1	0	0	1
F/5	1	0	1	→	G/6	1	1	0
G/6	1	1	0	→	C/2	0	1	0
H/7	1	1	1	→	D/3	0	1	1

C= X′Y′Z′+X′YZ′+X′YZ+XY′Z= X′Z′+Z(X⊕Y)

B= X′YZ+XY′Z+XYZ′+XYZ= Z(X⊕Y)+XY

A= X′Y′Z′+X′YZ+XY′Z′+XYZ= (Y⊕Z)′

Results:

Original Key	Switch I	Less Than (0 Logic)	Less Than (1 Logic)	Greater Than (0 Logic)	Greater Than (1 Logic)
A/000/0	0/000	111	000	111	000
B/001/1	0/000	111	000	110	001
C/010/2	0/000	111	000	101	010
D/011/3	0/000	111	000	100	011
E/100/4	0/000	111	000	011	100
F/101/5	0/000	111	000	010	101
G/110/6	0/000	111	000	001	110
H/111/7	0/000	111	000	000	111

Original Key	Switch I	Less Than (0 Logic)	Less Than (1 Logic)	Greater Than (0 Logic)	Greater Than (1 Logic)
A/000/0	1/001	111	000	110	001
B/001/1	1/001	110	001	110	001
C/010/2	1/001	111	000 001 A'	100	011 010 A'
D/011/3	1/001	110	001	100	011
E/100/4	1/001	111	000 001 A'	010	101 100 A'
F/101/5	1/001	110	001	010	101
G/110/6	1/001	111	000 001 A'	000	111 110 A'
H/111/7	1/001	110	001	000	111

Original Key	Switch I	Less Than (0 Logic)	Less Than (1 Logic)	Greater Than (0 Logic)	Greater Than (1 Logic)
A/000/0	2/010	111	000'	101	010
B/001/1	2/010	111	000 001 A'	100	011 010 A'
C/010/2	2/010	101	010	101	010
D/011/3	2/010	101	010	100	011
E/100/4	2/010	111	000 010 B'	001	110 100 B'
F/101/5	2/010	111	000 010 B'	000	111 101 B'
G/110/6	2/010	101	010	001	110
H/111/7	2/010	101	010	000	111

Original Key	Switch I	Less Than (0 Logic)	Less Than (1 Logic)	Greater Than (0 Logic)	Greater Than (1 Logic)
A/000/0	3/011	111	000	100	011
B/001/1	3/011	110	001	100	011
C/010/2	3/011	101	010	100	011
D/011/3	3/011	110	011	100	011
E/100/4	3/011	111	000 011 B'A'	000	111 100 B'A'
F/101/5	3/011	110	001 011 B'	000	111 101 B'
G/110/6	3/011	101	010 011 A'	000	111 110 A'
H/111/7	3/011	100	011	000	111

Original Key	Switch I	Less Than (0 Logic)	Less Than (1 Logic)	Greater Than (0 Logic)	Greater Than (1 Logic)
A/000/0	4/100	111	000	011	100
B/001/1	4/100	111	000 001 A'	010	101 100 A'
C/010/2	4/100	111	000 010 B'	001	110 100 B'
D/011/3	4/100	111	000 011 B'A'	000	111 100 B'A'
E/100/4	4/100	011	100	011	100
F/101/5	4/100	011	100	010	101
G/110/6	4/100	011	100	001	110
H/111/7	4/100	011	100	000	111

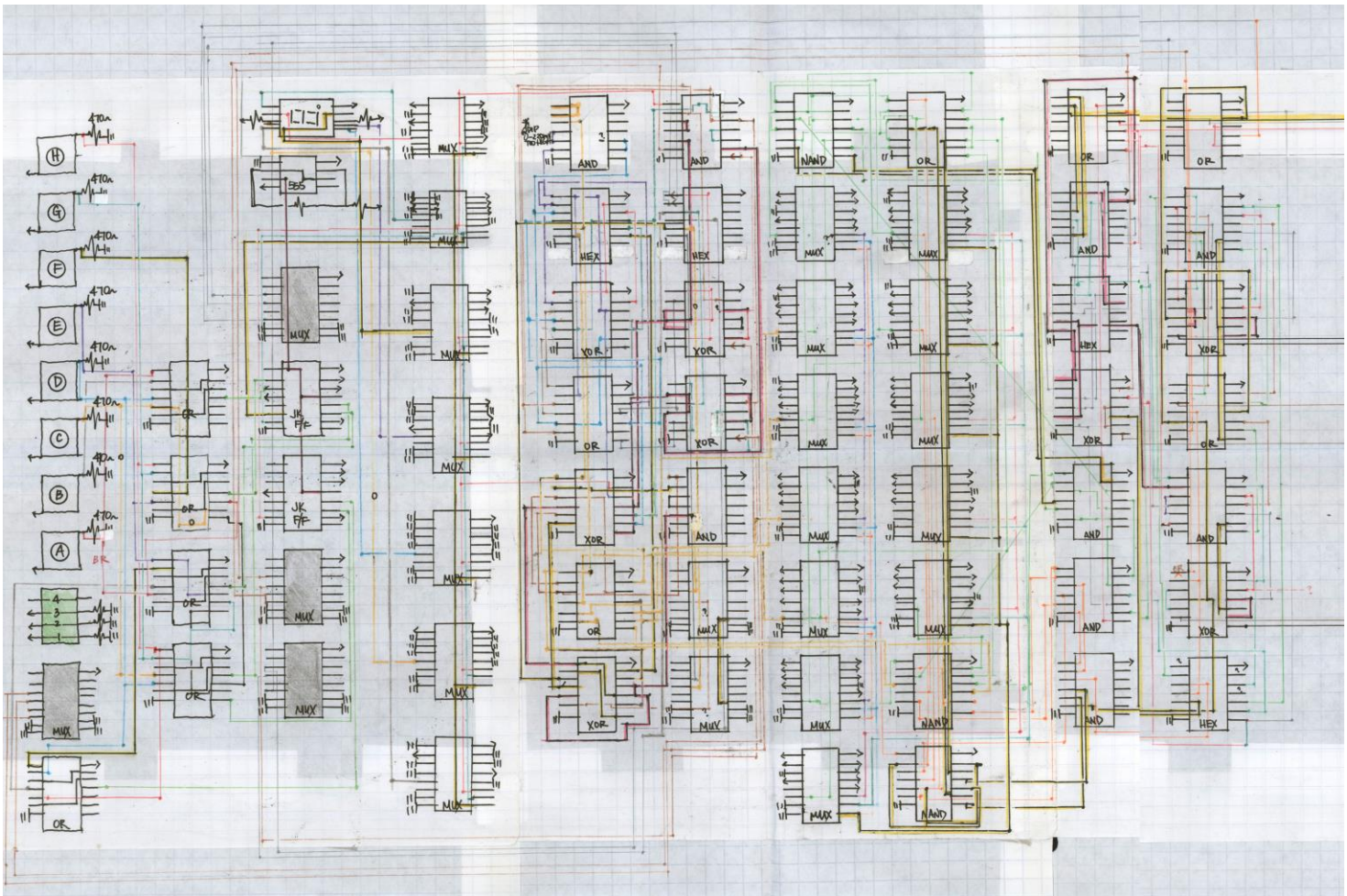
Original Key	Switch I	Less Than (0 Logic)	Less Than (1 Logic)	Greater Than (0 Logic)	Greater Than (1 Logic)
A/000/0	5/101	111	000	010	101
B/001/1	5/101	110	001	010	101
C/010/2	5/101	111	000 010 B'	000	111 101 B'
D/011/3	5/101	110	001 011 B'	000	111 101 B'
E/100/4	5/101	111	100	010	101
F/101/5	5/101	110	101	010	101
G/110/6	5/101	011	100 101 A'	000	111 110 A'
H/111/7	5/101	010	101	000	111

Original Key	Switch I	Less Than (0 Logic)	Less Than (1 Logic)	Greater Than (0 Logic)	Greater Than (1 Logic)
A/000/0	6/110	111	000	001	110
B/001/1	6/110	111	000 001 A'	000	111 110 A'
C/010/2	6/110	101	010	001	110
D/011/3	6/110	101	010 011 A'	000	111 110 A'
E/100/4	6/110	011	100	001	110
F/101/5	6/110	011	100 101 A'	000	111 110 A'
G/110/6	6/110	001	110	001	110
H/111/7	6/110	001	110	000	111

Original Key	Switch I	Less Than (0 Logic)	Less Than (1 Logic)	Greater Than (0 Logic)	Greater Than (1 Logic)
A/000/0	7/111	111	000	000	111
B/001/1	7/111	110	001	000	111
C/010/2	7/111	101	010	000	111
D/011/3	7/111	100	011	000	111
E/100/4	7/111	011	100	000	111
F/101/5	7/111	010	101	000	111
G/110/6	7/111	001	110	000	111
H/111/7	7/111	000	111	000	111

Conclusion:

As visible in the red numbers at its leftmost corner, the logic to generate the difference between the original key pressed and DIP switch 1 only works for about 60% of the combination possibilities. This is because the equations for Version II are only being constructed and tested on the circuit at this point. Version I's logic equations seemed to be repetitive and impractical despite the fact that Version II's truth value is contradictory. L's values cannot be the opposite of G when G is completely off at zero logic. However, I ignored the problems and decided to use a larger amount of chips than necessary in order to get the subtraction to work without an overflow. I forced the incorrect results to become valid at its corresponding line on its right side by using multiplexer logic that is being joined by AND and OR gates. Two categories of 12 multiplexers, one representing the original alphabets from A-H and the other symbolizing the number of the DIP switch 0-7, are being used to determine if the incorrect lines above exist when configured. I thought a memory chip will reduce the amount of multiplexers I will use, but I tried and there is a problem between reading the chip quickly and getting it to write "1" when DIP setting and the key, the ones that needs to be altered, when they are existing in the current machine. I don't want to delay the process so I simply decided to use more chips even though I have done this design twice. When I prove that the combination between the DIP switch and the key is true, I cascaded the mux from the original design to a new mux because a NOT logic is required to get the comparator to be put precisely into the adder.



Design: "The Intermediate": Failed Design

Following the first one-to-one logic encryption from the "beginning", the "intermediate" starts by flowing between two groups of multiplexers: one for DIP switch 1, and the other for the key bits that the "beginning" has ended. One multiplexer identifies whether or not DIP switch 1 is true to the coding of the chart while the second multiplexer determines its letter encrypted from the "beginning". The use of AND gates combines the logic of the two multiplexers in order to prove that both of them is true. Since only one key can individually be pressed, the OR gates will unite all the codes that specifies the numerical encryption. The encryption will always go back to the original code that A is 000, B is 001, C is 010 and so forth. The OR gates will power the three series of on or off switches that contains the information 000, 001, 010, 011, 100, 101, 110, and 111 on its resistor side. The information will only be read if the opposing side of the switch is at 3.5-5 volts. The design failed because the ON/OFF switches could not be powered correctly, causing oscillations in the JK flip-flops and the 555 timers.

Second Switch	0	1	2	3	4	5	6	7	→	Next Level
C/2	A/7	A/5	A/1	A/3	A/6	A/0	A/4	A/2	→	A/0
H/7	B/5	B/1	B/3	B/6	B/0	B/4	B/2	B/7	→	C/2
F/5	C/1	C/3	C/6	C/0	C/4	C/2	C/7	C/5	→	E/4
B/1	D/3	D/6	D/0	D/4	D/2	D/7	D/5	D/1	→	G/6
D/3	E/6	E/0	E/4	E/2	E/7	E/5	E/1	E/3	→	B/1
G/6	F/0	F/4	F/2	F/7	F/5	F/1	F/3	F/6	→	D/3
A/0	G/4	G/2	G/7	G/5	G/1	G/3	G/6	G/0	→	F/5
E/4	H/2	H/7	H/5	H/1	H/3	H/6	H/0	H/4	→	H/7

Color	Key=	Result	Key/Column	--	--	--	--	--	--	--
Red	A/0	000	F/0	E/1	D/2	C/3	B/4	A/5	H/6	G/7
Green	B/1	001	C/0	B/1	A/2	H/3	G/4	F/5	E/6	D/7
Yellow	C/2	010	H/0	G/1	F/2	E/3	D/4	C/5	B/6	A/7
Purple	D/3	011	D/0	C/1	B/2	A/3	H/4	G/5	F/6	E/7
Blue	E/4	100	G/0	F/1	E/2	D/3	C/4	B/5	A/6	H/7
Orange	F/5	101	B/0	A/1	H/2	G/3	F/4	E/5	D/6	C/7
Gray	G/6	110	E/0	D/1	C/2	B/3	A/4	H/5	G/6	F/7
Brown	H/7	111	A/0	H/1	G/2	F/3	E/4	D/5	C/6	B/7

A=000) F/0+E/1+D/2+C/3+B/4+A/5+H/6+G/7

B=001) C/0+B/1+A/2+H/3+G/4+F/5+E/6+D/7

C=010) H/0+G/1+F/2+E/3+D/4+C/5+B/6+A/7

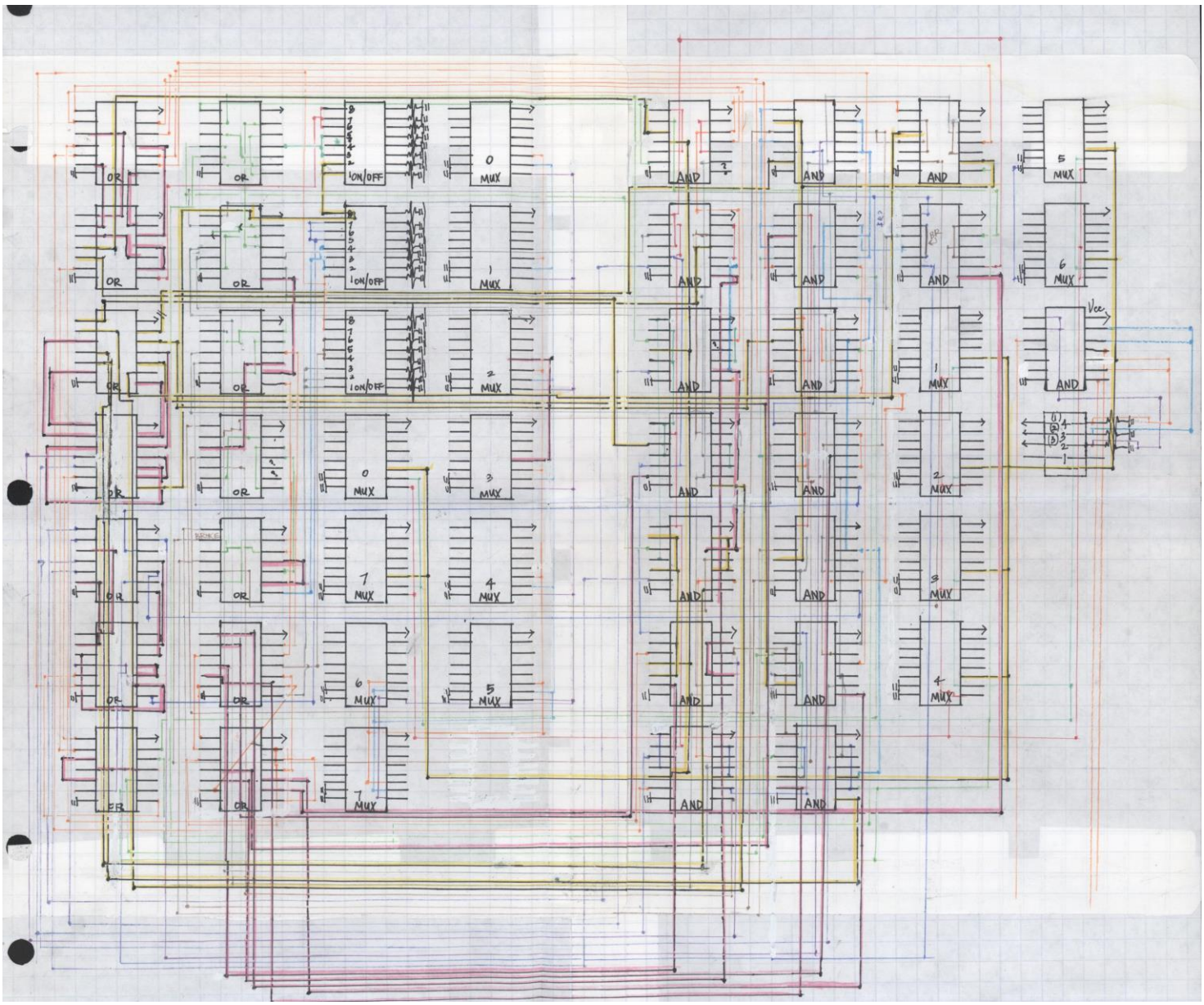
D=011) D/0+C/1+B/2+A/3+H/4+G/5+F/6+E/7

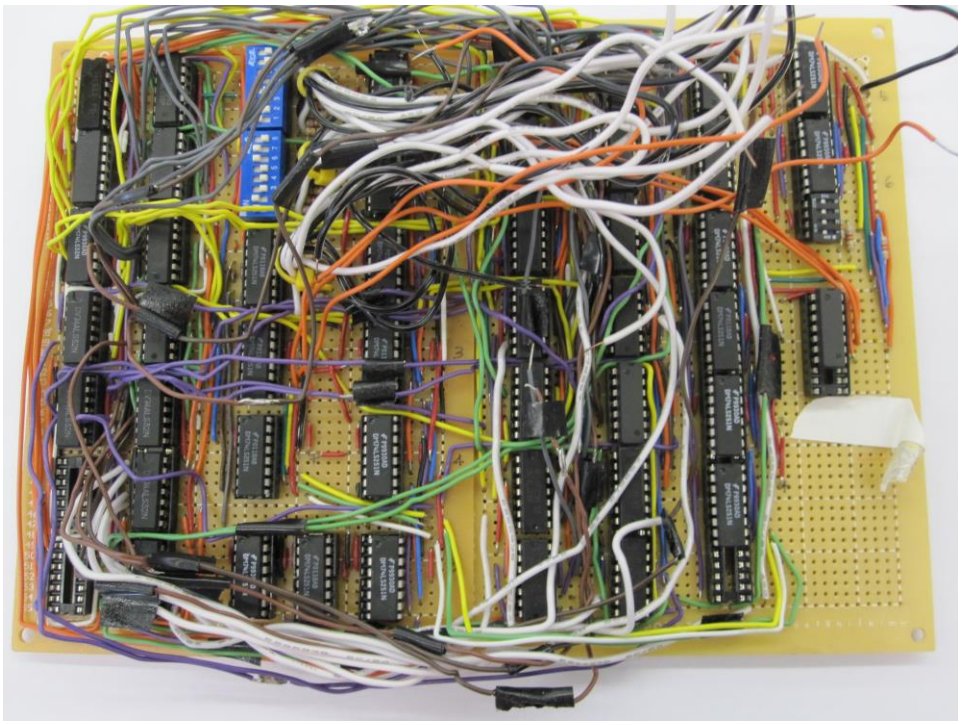
E=100) G/0+F/1+E/2+D/3+C/4+B/5+A/6+H/7

F=101) B/0+A/1+H/2+G/3+F/4+E/5+D/6+C/7

G=110) E/0+D/1+C/2+B/3+A/4+H/5+G/6+F/7

H=111) A/0+H/1+G/2+F/3+E/$+D/5+C/6+B/7





Design: "Final Encrypt: The Reflector": Proposed Idea

An opposing shift "1" or "0" delay is necessary between the most significant to the least significant bits of the reflector to form a constant scramble of numbers. Similar to the mechanical rotors in the original Enigma, the final encryption will alternate every time the alphabetical key is pressed. The reflector allows the two DIP switches to present a new combination code within the next time period, which is specified by unique series of on and off controls. Once a new key is being signaled into the logic, each bit

column of the reflector will be shifted independently from either a "1" to a "0" or a "0" to a "1". Its consecutive columns will remain the same in memory until the next key is introduced. The following key is purely dependant on the encrypted results generated by the previous key because the bit columns are being shifted towards the left respectively. When the user presses the key at the ninth time, the system will start back at its primary state. The alphabet to numerical code in the "beginning" maps the reused numbers that were created by the three-bit interchanging columns, equating a final letter of the machine. I did not do this because the system is too complex, and I should stick to the simplicity of the original Enigma, which is a series of repetitive counting initiated by the setting, and the one-to-one mappings that codes the number into a new letter. Therefore the final design, not including the reflector, the plugboard and reversing the current, is actually the simple Enigma machine.

Key=	X_1	Y_1	Z_1	Pri-mary	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	A	0	0	0
B/1	0	0	1	→	E	1	0	0
C/2	0	1	0	→	B	0	0	1
D/3	0	1	1	→	F	1	0	1
E/4	1	0	0	→	C	0	1	0
F/5	1	0	1	→	G	1	1	0
G/6	1	1	0	→	D	0	1	1
H/7	1	1	1	→	H	1	1	1

Key=	X_1	Y_1	Z_1	1st Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	E	1	0	0
B/1	0	0	1	→	A	0	0	0
C/2	0	1	0	→	F	1	0	1
D/3	0	1	1	→	B	0	0	1
E/4	1	0	0	→	G	1	1	0
F/5	1	0	1	→	C	0	1	0
G/6	1	1	0	→	H	1	1	1
H/7	1	1	1	→	D	0	1	1

Key=	X_1	Y_1	Z_1	2nd Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	G	1	1	0
B/1	0	0	1	→	C	0	1	0
C/2	0	1	0	→	H	1	1	1
D/3	0	1	1	→	D	0	1	1
E/4	1	0	0	→	E	1	0	0
F/5	1	0	1	→	A	0	0	0
G/6	1	1	0	→	F	1	0	1
H/7	1	1	1	→	B	0	0	1

Key=	X_1	Y_1	Z_1	3rd Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	H	1	1	1
B/1	0	0	1	→	D	0	1	1
C/2	0	1	0	→	G	1	1	0
D/3	0	1	1	→	C	0	1	0
E/4	1	0	0	→	F	1	0	1
F/5	1	0	1	→	B	0	0	1
G/6	1	1	0	→	E	1	0	0
H/7	1	1	1	→	A	0	0	0

Key=	X_1	Y_1	Z_1	4th Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	D	0	1	1
B/1	0	0	1	→	H	1	1	1
C/2	0	1	0	→	C	0	1	0
D/3	0	1	1	→	G	1	1	0
E/4	1	0	0	→	B	0	0	1
F/5	1	0	1	→	F	1	0	1
G/6	1	1	0	→	A	0	0	0
H/7	1	1	1	→	E	1	0	0

Key=	X_1	Y_1	Z_1	5th Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	B	0	0	1
B/1	0	0	1	→	F	1	0	1
C/2	0	1	0	→	A	0	0	0
D/3	0	1	1	→	E	1	0	0
E/4	1	0	0	→	B	0	1	1
F/5	1	0	1	→	H	1	1	1
G/6	1	1	0	→	C	0	1	0
H/7	1	1	1	→	G	1	1	0

Key=	X_1	Y_1	Z_1	6th Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	A	0	0	0
B/1	0	0	1	→	E	1	0	0
C/2	0	1	0	→	B	0	0	1
D/3	0	1	1	→	F	1	0	1
E/4	1	0	0	→	C	0	1	0
F/5	1	0	1	→	G	1	1	0
G/6	1	1	0	→	D	0	1	1
H/7	1	1	1	→	H	1	1	1

Key=	X_1	Y_1	Z_1	7th Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	F	1	0	1
B/1	0	0	1	→	B	0	0	1
C/2	0	1	0	→	E	1	0	0
D/3	0	1	1	→	A	0	0	0
E/4	1	0	0	→	H	1	1	1
F/5	1	0	1	→	D	0	1	1
G/6	1	1	0	→	G	1	1	0
H/7	1	1	1	→	C	0	1	0

Key=	X_1	Y_1	Z_1	8th Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	H	1	1	1
B/1	0	0	1	→	D	0	1	1
C/2	0	1	0	→	G	1	1	0
D/3	0	1	1	→	C	0	1	0
E/4	1	0	0	→	F	1	0	1
F/5	1	0	1	→	B	0	0	1
G/6	1	1	0	→	E	1	0	0
H/7	1	1	1	→	A	0	0	0

Key=	X_1	Y_1	Z_1	9th Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	G	1	1	0
B/1	0	0	1	→	C	0	1	0
C/2	0	1	0	→	H	1	1	1
D/3	0	1	1	→	D	0	1	1
E/4	1	0	0	→	E	1	0	0
F/5	1	0	1	→	A	0	0	0
G/6	1	1	0	→	F	1	0	1

H/7	1	1	1	→	B	0	0	1

Key=	X_1	Y_1	Z_1	10^{th} Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	C	0	1	0
B/1	0	0	1	→	G	1	1	0
C/2	0	1	0	→	D	0	1	1
D/3	0	1	1	→	H	1	1	1
E/4	1	0	0	→	B	0	0	0
F/5	1	0	1	→	E	1	0	0
G/6	1	1	0	→	B	0	0	1
H/7	1	1	1	→	G	1	0	1

Key=	X_1	Y_1	Z_1	11^{th} Key	ENC. KEY	C_1	B_1	A_1
A/0	0	0	0	→	A	0	0	0
B/1	0	0	1	→	E	1	0	0
C/2	0	1	0	→	B	0	0	1
D/3	0	1	1	→	F	1	0	1
E/4	1	0	0	→	C	0	1	0
F/5	1	0	1	→	G	1	1	0
G/6	1	1	0	→	D	0	1	1
H/7	1	1	1	→	H	1	1	1

Design: One-To-One Mapping Logic: Second Encryption

The "intermediate" and the "end" is not the final product of my project. It is part of the project's experimental process that had been pondered, and tried but did not work. The second encryption is just a repeat of the "beginning" part of the Enigma, except that the logic that decodes a final new letter is a different logic system as shown below.

X=GREEN

Y=WHITE

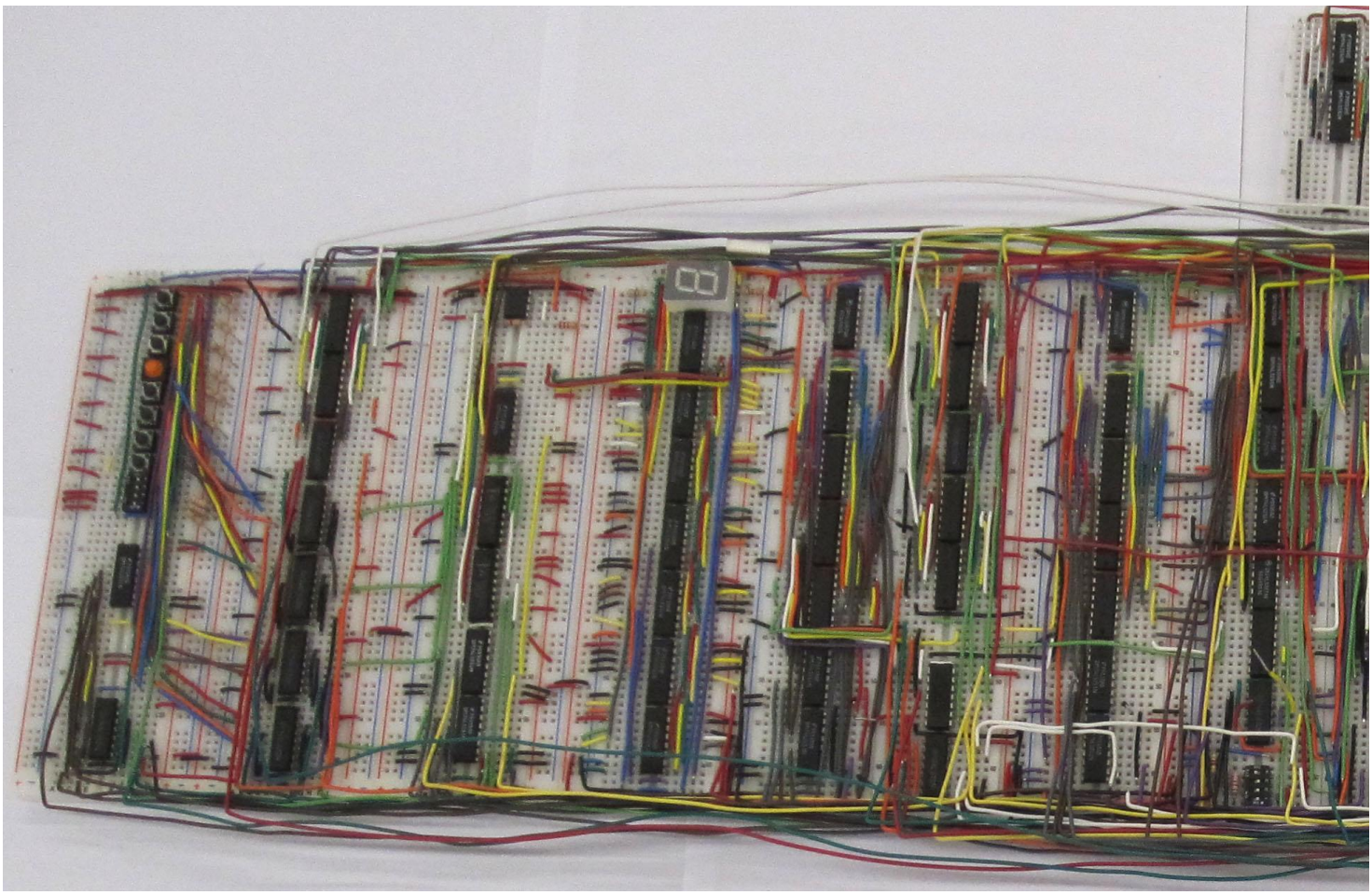
Z=BLACK

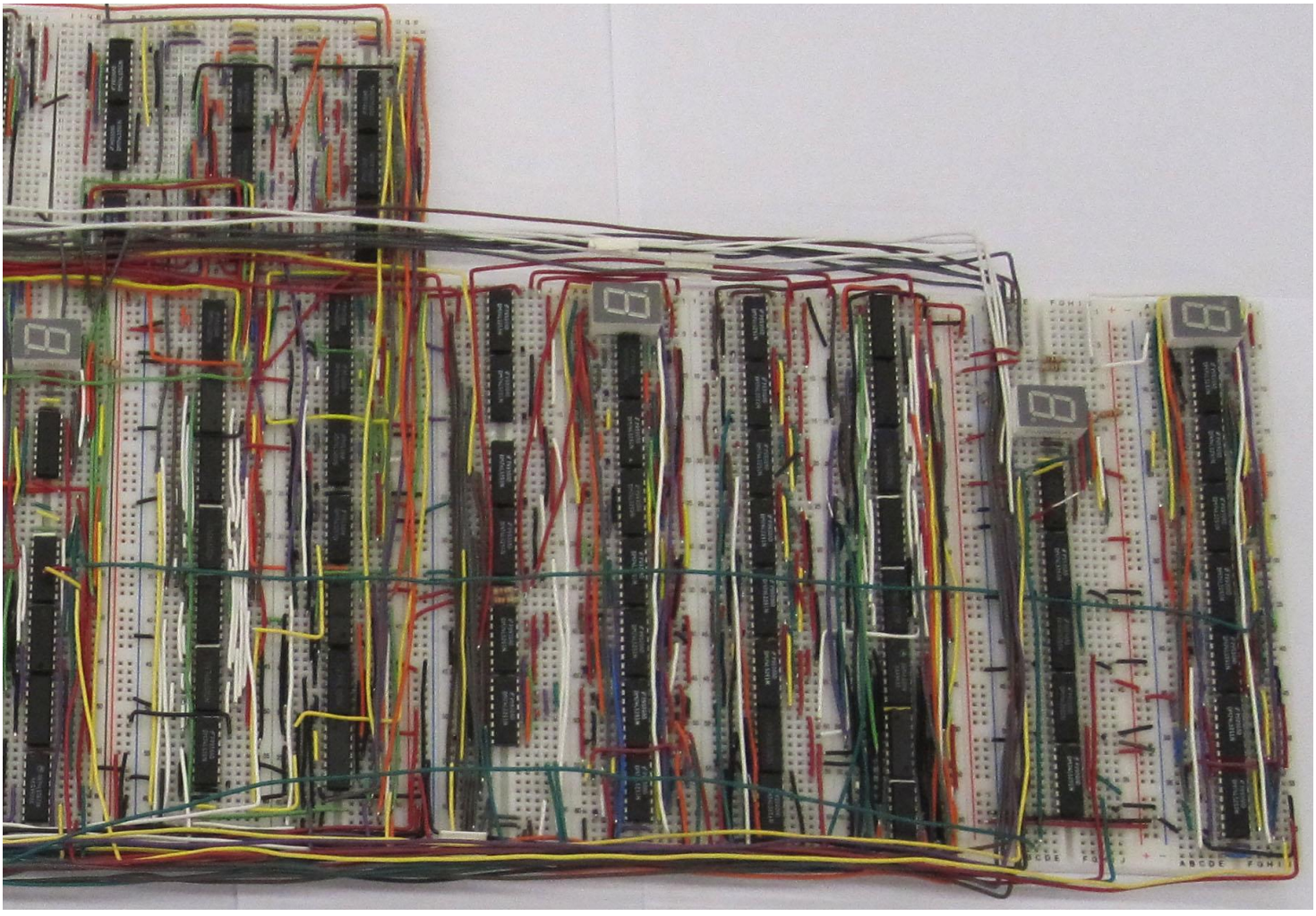
Intermediate	X	Y	Z	→	End	C	B	A
A/0	0	0	0	→	C/5	0	1	0
B/1	0	0	1	→	H/0	1	1	1
C/2	0	1	0	→	F/4	1	0	1
D/3	0	1	1	→	B/7	0	0	1
E/4	1	0	0	→	D/1	0	1	1
F/5	1	0	1	→	G/6	1	1	0
G/6	1	1	0	→	A/2	0	0	0
H/7	1	1	1	→	E/3	1	0	0

$C= X'Y'Z+X'YZ'+XY'Z+XYZ= X(Y\oplus Z)'+XZ$

$B= X'Y'Z'+X'Y'Z+XY'Z'+XY'Z=Y'$

$A= X'Y'Z+X'YZ'+X'YZ+XY'Z'= X'(Y\oplus Z)+X'YZ+XY'Z'$





Enigma Coding Sheet

KEY	SETTING SW.1: A	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: A	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	0/A	→	A=F/5	F/5	A/0	0/A	→	A=C/0	C/0
B/1	B/1	1/B	→	B=A/0	A/0	B/1	1/B	→	B=H/1	H/1
C/2	C/2	2/C	→	C=E/4	E/4	C/2	2/C	→	C=F/2	F/2
D/3	D/3	3/D	→	D=H/7	H/7	D/3	3/D	→	D=B/3	B/3
E/4	E/4	4/E	→	E=B/1	B/1	E/4	4/E	→	E=D/4	D/3
F/5	F/5	5/F	→	F=G/6	G/6	F/5	5/F	→	F=G/5	G/5
G/6	G/6	6/G	→	G=C/2	C/2	G/6	6/G	→	G=A/6	A/6
H/7	H/7	7/H	→	H=D/3	D/3	H/7	7/H	→	H=E/7	E/7

KEY	SETTING SW.1: A	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: B	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	0/A	→	A=F/5	F/5	A/0	4/E	→	A=C/0	D/3
B/1	B/1	1/B	→	B=A/0	A/0	B/1	1/B	→	B=H/1	H/7
C/2	C/2	2/C	→	C=E/4	E/4	C/2	3/D	→	C=F/2	B/1
D/3	D/3	3/D	→	D=H/7	H/7	D/3	6/G	→	D=B/3	A/0
E/4	E/4	4/E	→	E=B/1	B/1	E/4	0/A	→	E=D/4	C/2
F/5	F/5	5/F	→	F=G/6	G/6	F/5	5/F	→	F=G/5	G/6
G/6	G/6	6/G	→	G=C/2	C/2	G/6	1/B	→	G=A/6	H/7
H/7	H/7	7/H	→	H=D/3	D/3	H/7	2/C	→	H=E/7	F/5

KEY	SETTING SW.1: A	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: C	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	0/A	→	A=F/5	F/5	A/0	3/D	→	A=C/0	B/1
B/1	B/1	1/B	→	B=A/0	A/0	B/1	2/C	→	B=H/1	F/5
C/2	C/2	2/C	→	C=E/4	E/4	C/2	2/C	→	C=F/2	F/5
D/3	D/3	3/D	→	D=H/7	H/7	D/3	5/F	→	D=B/3	G/6
E/4	E/4	4/E	→	E=B/1	B/1	E/4	1/B	→	E=D/4	H/7
F/5	F/5	5/F	→	F=G/6	G/6	F/5	4/E	→	F=G/5	D/3
G/6	G/6	6/G	→	G=C/2	C/2	G/6	0/A	→	G=A/6	C/2
H/7	H/7	7/H	→	H=D/3	D/3	H/7	1/B	→	H=E/7	H/7

KEY	SETTING SW.1: A	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: D	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	0/A	→	A=F/5	F/5	A/0	2/C	→	A=C/0	F/5
B/1	B/1	1/B	→	B=A/0	A/0	B/1	3/D	→	B=H/1	B/1
C/2	C/2	2/C	→	C=E/4	E/4	C/2	1/B	→	C=F/2	H/7
D/3	D/3	3/D	→	D=H/7	H/7	D/3	4/E	→	D=B/3	D/3
E/4	E/4	4/E	→	E=B/1	B/1	E/4	2/C	→	E=D/4	F/5
F/5	F/5	5/F	→	F=G/6	G/6	F/5	3/D	→	F=G/5	B/1
G/6	G/6	6/G	→	G=C/2	C/2	G/6	1/B	→	G=A/6	H/7
H/7	H/7	7/H	→	H=D/3	D/3	H/7	0/A	→	H=E/7	C/2

KEY	SETTING SW.1: A	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: E	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	0/A	→	A=F/5	F/5	A/0	1/B	→	A=C/0	H/7
B/1	B/1	1/B	→	B=A/0	A/0	B/1	4/E	→	B=H/1	D/3
C/2	C/2	2/C	→	C=E/4	E/4	C/2	0/A	→	C=F/2	C/2
D/3	D/3	3/D	→	D=H/7	H/7	D/3	3/D	→	D=B/3	B/1
E/4	E/4	4/E	→	E=B/1	B/1	E/4	3/D	→	E=D/4	B/1
F/5	F/5	5/F	→	F=G/6	G/6	F/5	2/C	→	F=G/5	F/5
G/6	G/6	6/G	→	G=C/2	C/2	G/6	2/C	→	G=A/6	F/5
H/7	H/7	7/H	→	H=D/3	D/3	H/7	1/B	→	H=E/7	H/7

KEY	SETTING SW.1: A	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: F	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	0/A	→	A=F/5	F/5	A/0	0/A	→	A=C/0	C/2
B/1	B/1	1/B	→	B=A/0	A/0	B/1	5/F	→	B=H/1	G/6
C/2	C/2	2/C	→	C=E/4	E/4	C/2	1/B	→	C=F/2	H/7
D/3	D/3	3/D	→	D=H/7	H/7	D/3	2/C	→	D=B/3	F/5
E/4	E/4	4/E	→	E=B/1	B/1	E/4	4/E	→	E=D/4	D/3
F/5	F/5	5/F	→	F=G/6	G/6	F/5	1/B	→	F=G/5	H/7
G/6	G/6	6/G	→	G=C/2	C/2	G/6	3/D	→	G=A/6	B/1
H/7	H/7	7/H	→	H=D/3	D/3	H/7	2/C	→	H=E/7	F/5

KEY	SETTING SW.1: A	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: G	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	0/A	→	A=F/5	F/5	A/0	1/B	→	A=C/0	H/7
B/1	B/1	1/B	→	B=A/0	A/0	B/1	6/G	→	B=H/1	A/0
C/2	C/2	2/C	→	C=E/4	E/4	C/2	2/C	→	C=F/2	F/5
D/3	D/3	3/D	→	D=H/7	H/7	D/3	1/B	→	D=B/3	H/7
E/4	E/4	4/E	→	E=B/1	B/1	E/4	5/F	→	E=D/4	G/6
F/5	F/5	5/F	→	F=G/6	G/6	F/5	0/A	→	F=G/5	C/2
G/6	G/6	6/G	→	G=C/2	C/2	G/6	4/E	→	G=A/6	D/3
H/7	H/7	7/H	→	H=D/3	D/3	H/7	3/D	→	H=E/7	B/1

KEY	SETTING SW.1: A	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: H	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	0/A	→	A=F/5	F/5	A/0	2/C	→	A=C/0	F/5
B/1	B/1	1/B	→	B=A/0	A/0	B/1	7/H	→	B=H/1	E/4
C/2	C/2	2/C	→	C=E/4	E/4	C/2	3/D	→	C=F/2	B/1
D/3	D/3	3/D	→	D=H/7	H/7	D/3	0/A	→	D=B/3	C/2
E/4	E/4	4/E	→	E=B/1	B/1	E/4	6/G	→	E=D/4	A/0
F/5	F/5	5/F	→	F=G/6	G/6	F/5	1/B	→	F=G/5	H/7
G/6	G/6	6/G	→	G=C/2	C/2	G/6	5/F	→	G=A/6	G/6
H/7	H/7	7/H	→	H=D/3	D/3	H/7	4/E	→	H=E/7	D/3

KEY	SETTING SW.1: B	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: A	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	1/B	→	A=F/5	A/0	A/0	0/A	→	A=C/0	C/2
B/1	B/1	0/A	→	B=A/0	F/5	B/1	5/F	→	B=H/1	G/6
C/2	C/2	1/B	→	C=E/4	A/0	C/2	0/A	→	C=F/2	C/2
D/3	D/3	2/C	→	D=H/7	E/4	D/3	4/E	→	D=B/3	D/3
E/4	E/4	3/D	→	E=B/1	H/7	E/4	7/H	→	E=D/4	E/4
F/5	F/5	4/E	→	F=G/6	B/1	F/5	1/B	→	F=G/5	H/7
G/6	G/6	5/F	→	G=C/2	G/6	G/6	6/G	→	G=A/6	A/0
H/7	H/7	6/G	→	H=D/3	C/2	H/7	2/C	→	H=E/7	F/5

KEY	SETTING SW.1: B	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: B	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	1/B	→	A=F/5	A/0	A/0	1/B	→	A=C/0	H/7
B/1	B/1	0/A	→	B=A/0	F/5	B/1	4/E	→	B=H/1	D/3
C/2	C/2	1/B	→	C=E/4	A/0	C/2	1/B	→	C=F/2	H/7
D/3	D/3	2/C	→	D=H/7	E/4	D/3	3/D	→	D=B/3	B/1
E/4	E/4	3/D	→	E=B/1	H/7	E/4	6/G	→	E=D/4	A/0
F/5	F/5	4/E	→	F=G/6	B/1	F/5	0/A	→	F=G/5	C/2
G/6	G/6	5/F	→	G=C/2	G/6	G/6	5/F	→	G=A/6	G/6
H/7	H/7	6/G	→	H=D/3	C/2	H/7	1/B	→	H=E/7	H/7

KEY	SETTING SW.1: B	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: C	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	1/B	→	A=F/5	A/0	A/0	2/C	→	A=C/0	F/5
B/1	B/1	0/A	→	B=A/0	F/5	B/1	3/D	→	B=H/1	B/1
C/2	C/2	1/B	→	C=E/4	A/0	C/2	2/C	→	C=F/2	F/5
D/3	D/3	2/C	→	D=H/7	E/4	D/3	2/C	→	D=B/3	F/5
E/4	E/4	3/D	→	E=B/1	H/7	E/4	5/F	→	E=D/4	G/6
F/5	F/5	4/E	→	F=G/6	B/1	F/5	1/B	→	F=G/5	H/7
G/6	G/6	5/F	→	G=C/2	G/6	G/6	4/E	→	G=A/6	D/3
H/7	H/7	6/G	→	H=D/3	C/2	H/7	2/C	→	H=E/7	F/5

KEY	SETTING SW.1: B	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: D	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	1/B	→	A=F/5	A/0	A/0	3/D	→	A=C/0	B/1
B/1	B/1	0/A	→	B=A/0	F/5	B/1	2/C	→	B=H/1	F/5
C/2	C/2	1/B	→	C=E/4	A/0	C/2	3/D	→	C=F/2	B/1
D/3	D/3	2/C	→	D=H/7	E/4	D/3	1/B	→	D=B/3	H/7
E/4	E/4	3/D	→	E=B/1	H/7	E/4	4/E	→	E=D/4	D/3
F/5	F/5	4/E	→	F=G/6	B/1	F/5	2/C	→	F=G/5	F/5
G/6	G/6	5/F	→	G=C/2	G/6	G/6	3/D	→	G=A/6	B/1
H/7	H/7	6/G	→	H=D/3	C/2	H/7	4/E	→	H=E/7	D/3

KEY	SETTING SW.1: B	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: E	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	1/B	→	A=F/5	A/0	A/0	4/E	→	A=C/0	D/3
B/1	B/1	0/A	→	B=A/0	F/5	B/1	1/B	→	B=H/1	H/7
C/2	C/2	1/B	→	C=E/4	A/0	C/2	4/E	→	C=F/2	D/3
D/3	D/3	2/C	→	D=H/7	E/4	D/3	0/A	→	D=B/3	C/2
E/4	E/4	3/D	→	E=B/1	H/7	E/4	3/D	→	E=D/4	B/1
F/5	F/5	4/E	→	F=G/6	B/1	F/5	3/D	→	F=G/5	B/1
G/6	G/6	5/F	→	G=C/2	G/6	G/6	2/C	→	G=A/6	F/5
H/7	H/7	6/G	→	H=D/3	C/2	H/7	2/C	→	H=E/7	F/5

KEY	SETTING SW.1: B	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: F	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	1/B	→	A=F/5	A/0	A/0	5/F	→	A=C/0	G/6
B/1	B/1	0/A	→	B=A/0	F/5	B/1	0/A	→	B=H/1	C/2
C/2	C/2	1/B	→	C=E/4	A/0	C/2	5/F	→	C=F/2	G/6
D/3	D/3	2/C	→	D=H/7	E/4	D/3	1/B	→	D=B/3	H/7
E/4	E/4	3/D	→	E=B/1	H/7	E/4	2/C	→	E=D/4	F/5
F/5	F/5	4/E	→	F=G/6	B/1	F/5	4/E	→	F=G/5	D/3
G/6	G/6	5/F	→	G=C/2	G/6	G/6	1/B	→	G=A/6	H/7
H/7	H/7	6/G	→	H=D/3	C/2	H/7	3/D	→	H=E/7	B/1

KEY	SETTING SW.1: B	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: G	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	1/B	→	A=F/5	A/0	A/0	6/G	→	A=C/0	A/0
B/1	B/1	0/A	→	B=A/0	F/5	B/1	1/B	→	B=H/1	H/7
C/2	C/2	1/B	→	C=E/4	A/0	C/2	6/G	→	C=F/2	A/0
D/3	D/3	2/C	→	D=H/7	E/4	D/3	2/C	→	D=B/3	F/5
E/4	E/4	3/D	→	E=B/1	H/7	E/4	1/B	→	E=D/4	H/7
F/5	F/5	4/E	→	F=G/6	B/1	F/5	5/F	→	F=G/5	G/6
G/6	G/6	5/F	→	G=C/2	G/6	G/6	0/A	→	G=A/6	C/2
H/7	H/7	6/G	→	H=D/3	C/2	H/7	4/E	→	H=E/7	D/3

KEY	SETTING SW.1: B	COUNT		ENCRYPTION	NEW LETTER	SETTING SW.2: H	COUNT		ENCRYPTION	FINAL LETTER
A/0	A/0	1/B	→	A=F/5	A/0	A/0	7/H	→	A=C/0	E/4
B/1	B/1	0/A	→	B=A/0	F/5	B/1	2/C	→	B=H/1	F/5
C/2	C/2	1/B	→	C=E/4	A/0	C/2	7/H	→	C=F/2	E/4
D/3	D/3	2/C	→	D=H/7	E/4	D/3	3/D	→	D=B/3	B/1
E/4	E/4	3/D	→	E=B/1	H/7	E/4	0/A	→	E=D/4	C/2
F/5	F/5	4/E	→	F=G/6	B/1	F/5	6/G	→	F=G/5	A/0
G/6	G/6	5/F	→	G=C/2	G/6	G/6	1/B	→	G=A/6	H/7
H/7	H/7	6/G	→	H=D/3	C/2	H/7	5/F	→	H=E/7	G/6

*And so on…The Encryption is repetitive because it is just a one-to-one mapping of from the setting's subtracted number to its new letter. The red lines signify the decryption setting code.