UNITED STATES
AIR FORCE
ACADEMY

ECE 383 – Embedded Computer Systems II

Lecture 24 - Direct Digital Synthesis and Linear Interpolation

UNITED STATES AIR FORCE ACADEMY

$$f = \frac{F_s \cdot X}{2^N}$$

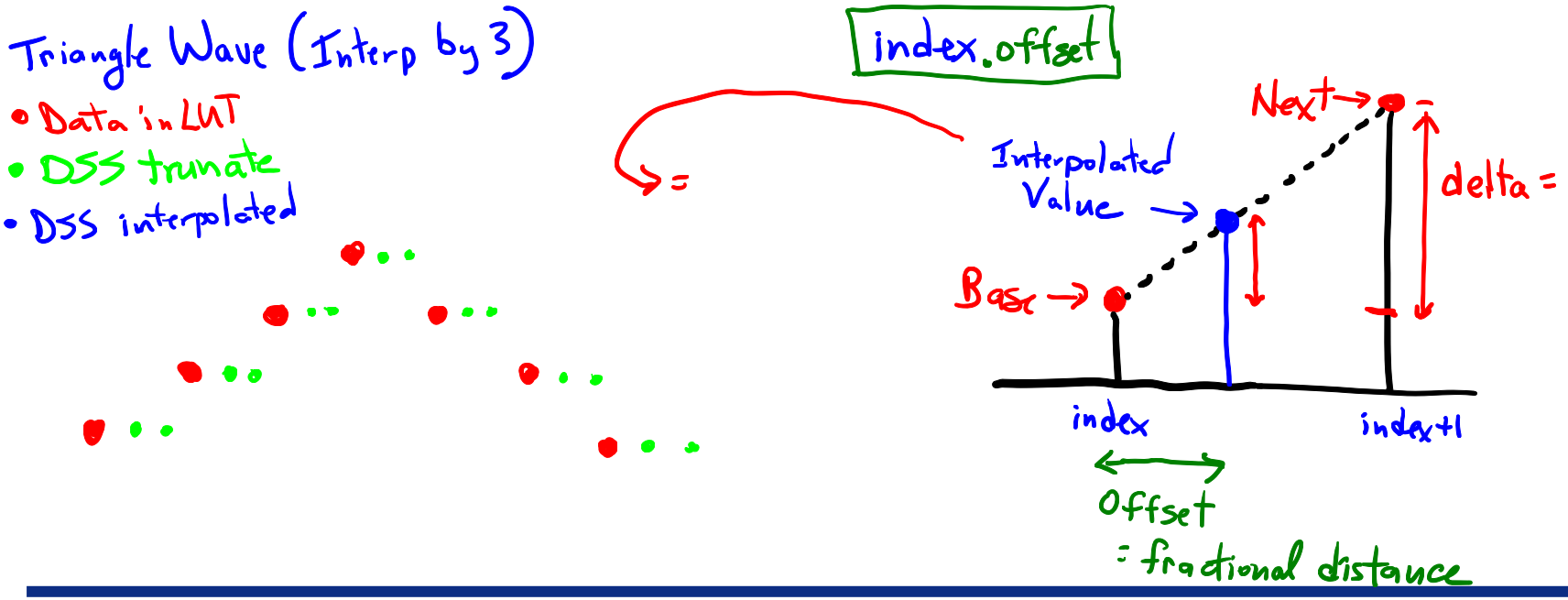
Lesson Outline

- **Project Proposals Due BOC TODAY**

$X = 1.0$

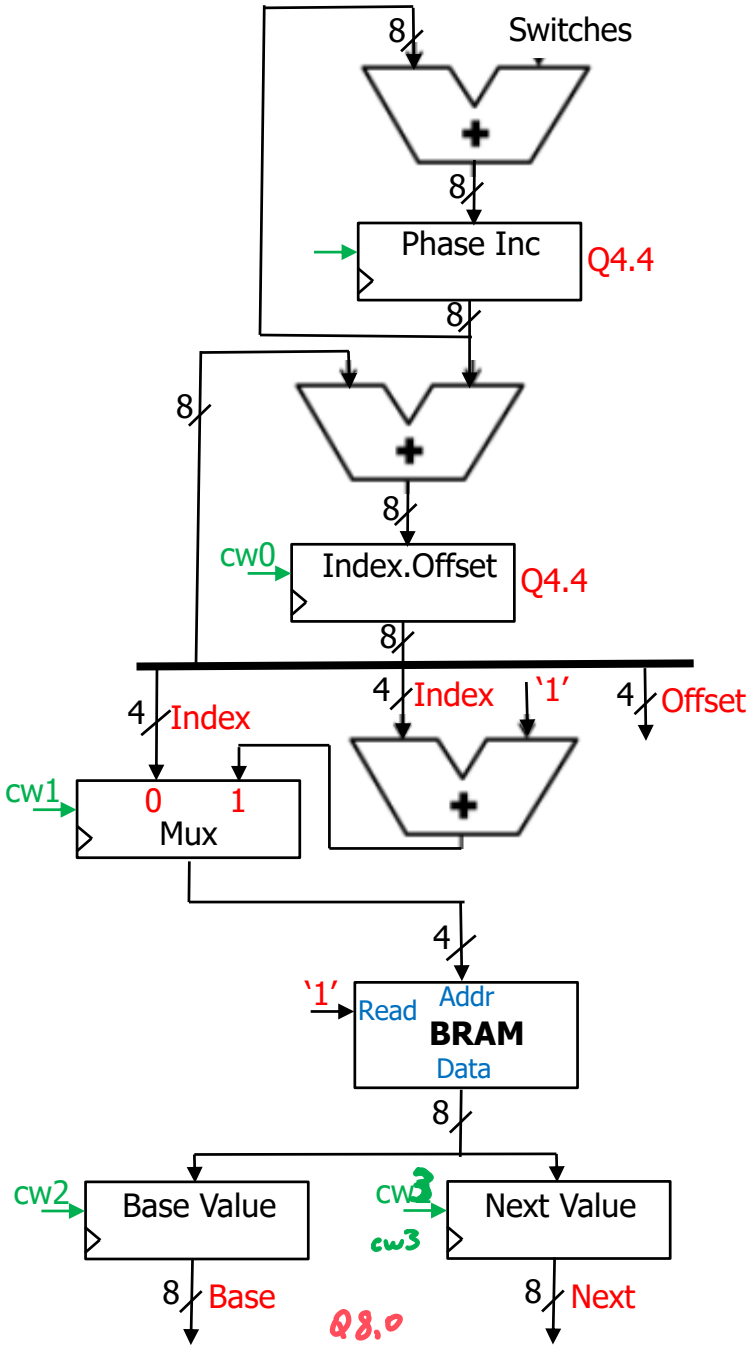
Jump by $X = 0.333\overline{3}$

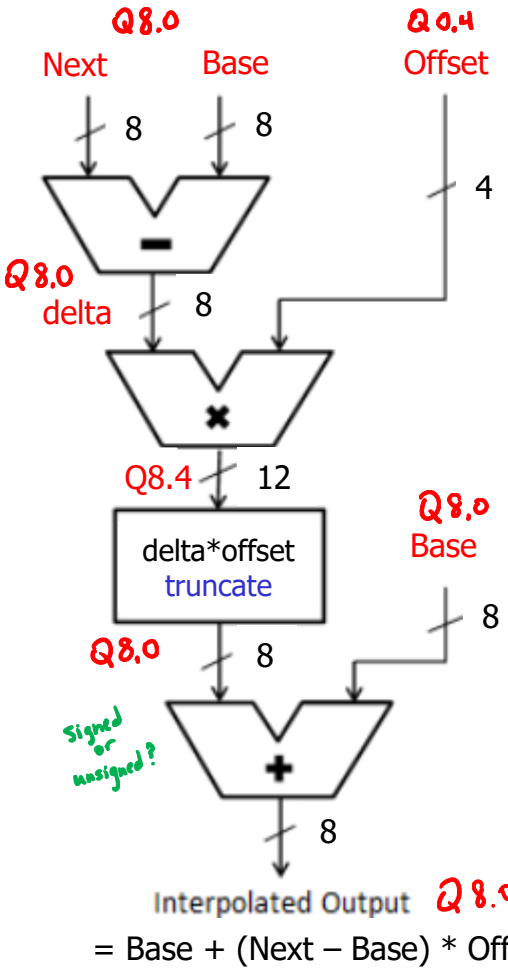
- **Direct Digital Synthesis with Interpolation**



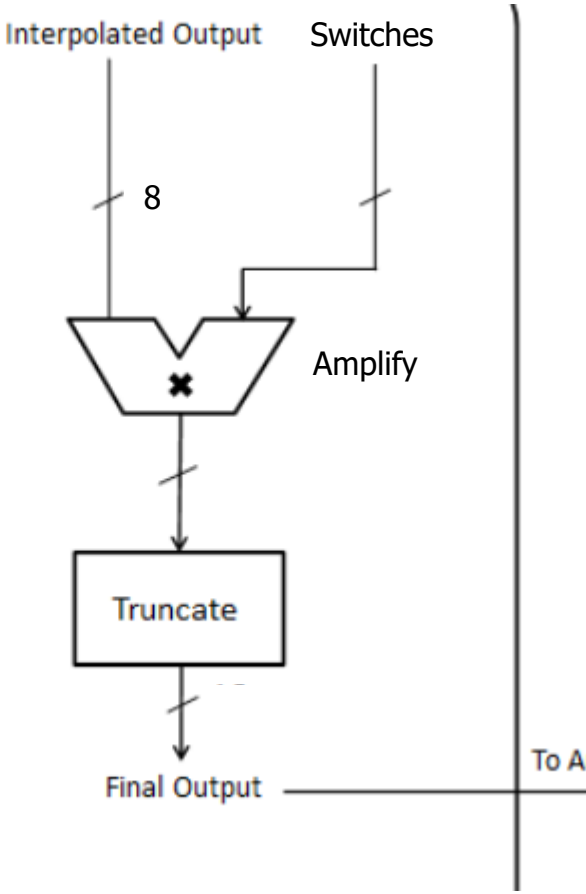
Interpolation Block Diagram

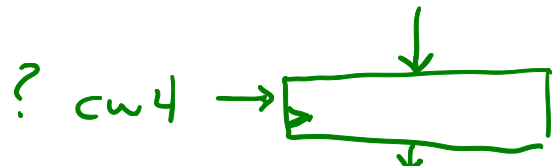
signed or unsigned?





= Base + (Next – Base) * Offset





Audio Codec produces _____ samples (18-bit)

Lab4 Function Generator produces _____ samples

LUT – Look Up Table

UNITED STATES AIR FORCE ACADEMY

LUT – Look Up Table

- Let's say that you wanted to compute the square root of an value using the microBlaze.
- If you were lucky enough to have a compiler which provided this function you could just use the math library functions.
- If on the other hand, you did not have the use of such a library, you would have to figure out a way to compute the square root.

UNITED STATES AIR FORCE ACADEMY

LUT – Look Up Table

- There are many ways to compute the SQRT Function:
- Crack open a book to find a mathematical expression
  - Unfortunately this typically leads to timely computations
- You could enumerate an every possible value of x and its square root.
  - We could then look-up a SQRT in the table, by going to the row corresponding to the x and retrieving its value.
  - This approach seems silly since it would use a lot of space
  - We could reduce this space by eliminating entries
  - We save space at the expense of introducing errors.
- If you wanted the SQRT for x and its entry wasn't in the table you would have to use the closest x in the table

UNITED STATES AIR FORCE ACADEMY

LUT – Look Up Table

- A good compromise among all three of these design constraints:
  - Space
  - Time
  - Error
- Is to use Interpolation in a partial Look-Up-Table (LUT)

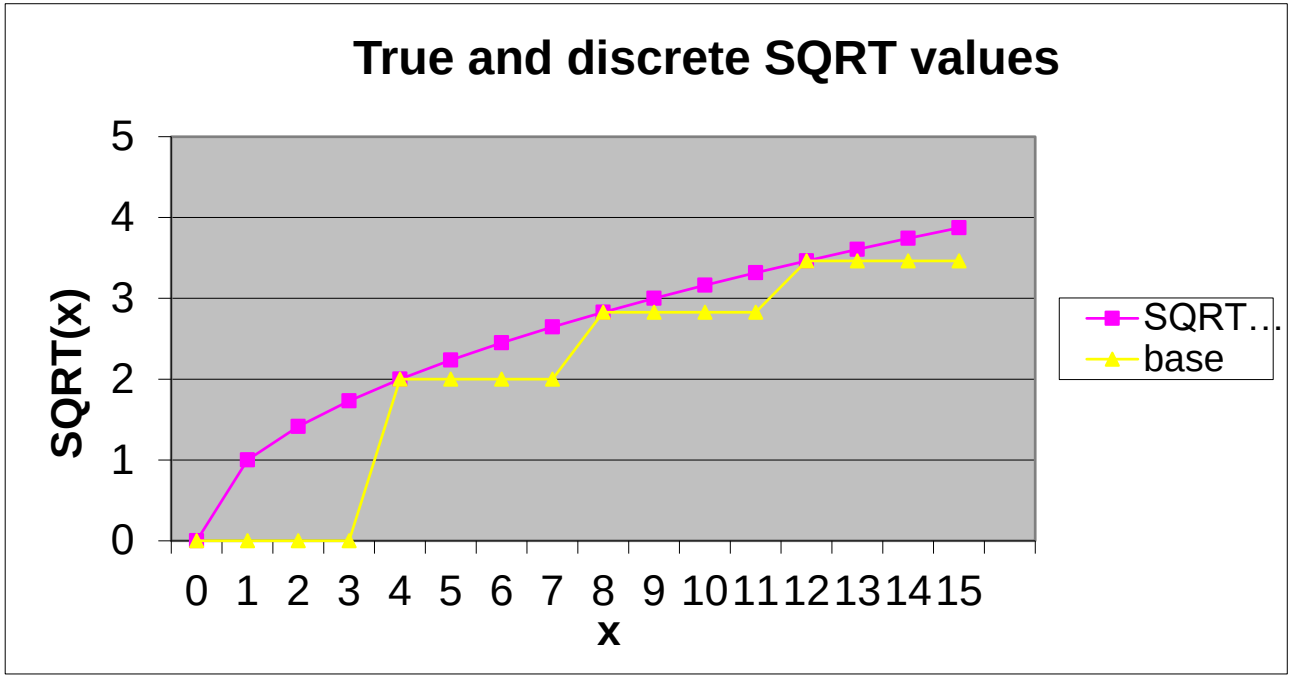
UNITED STATES AIR FORCE ACADEMY

Interpolation

- "Interpolation is a mathematical method of creating missing data. ... There are many methods of interpolation, but one simple method would be to generate a new value by using the average of the value of the two values on either side of the one to be created.“
- This average is also referred to as linear interpolation.
- For example if you have a value of x which is 1/2 way between 0 and 4 then you *assume* that the SQRT is 1/2 between 0 and 2.

UNITED STATES AIR FORCE ACADEMY

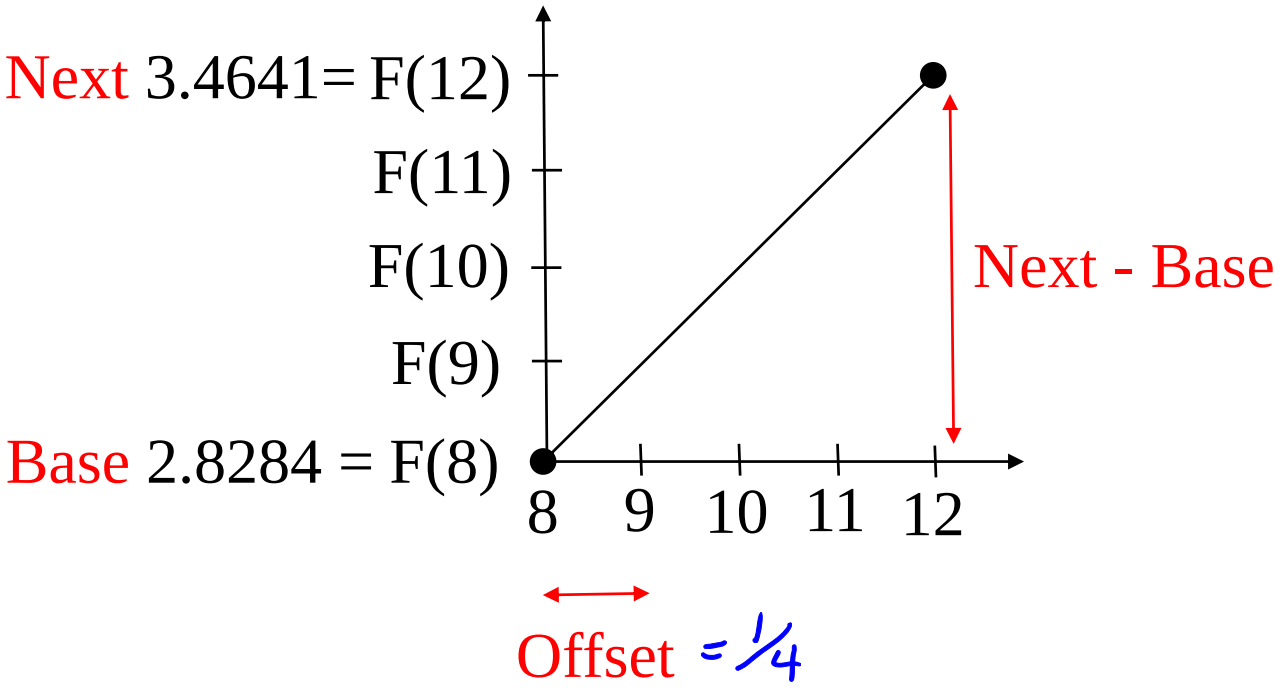
Example Interpolation



UNITED STATES AIR FORCE ACADEMY

Linear Interpolation

- We know that if f(x)=y and f(x+4)=z then we estimate the intermediate values of f(x+1), f(x+2), and f(x+3) by drawing a straight line between y and z and using the points on this line to estimate the function between x and x+4. ≈ BASE + OFFSET • DELTA
- For example, let: F(8)=2.8284 and F(12)=3.4641 then
  - F(9) = 2.8284 + 1/4(3.4641-2.8284) = 2.987
  - F(10) = 2.8284 + 2/4(3.4641-2.8284) = 3.146
  - F(11) = 2.8284 + 3/4(3.4641-2.8284) = 3.305
- We understand that this is an approximation and consequently we will have error, but sometimes close is better than exact in embedded computing especially when time is of the essence.


Next 3.4641= F(12)
F(11)
F(10)
F(9)
Base 2.8284 = F(8)
8
9
10
11
12
Next - Base
Offset = 1/4

UNITED STATES AIR FORCE ACADEMY

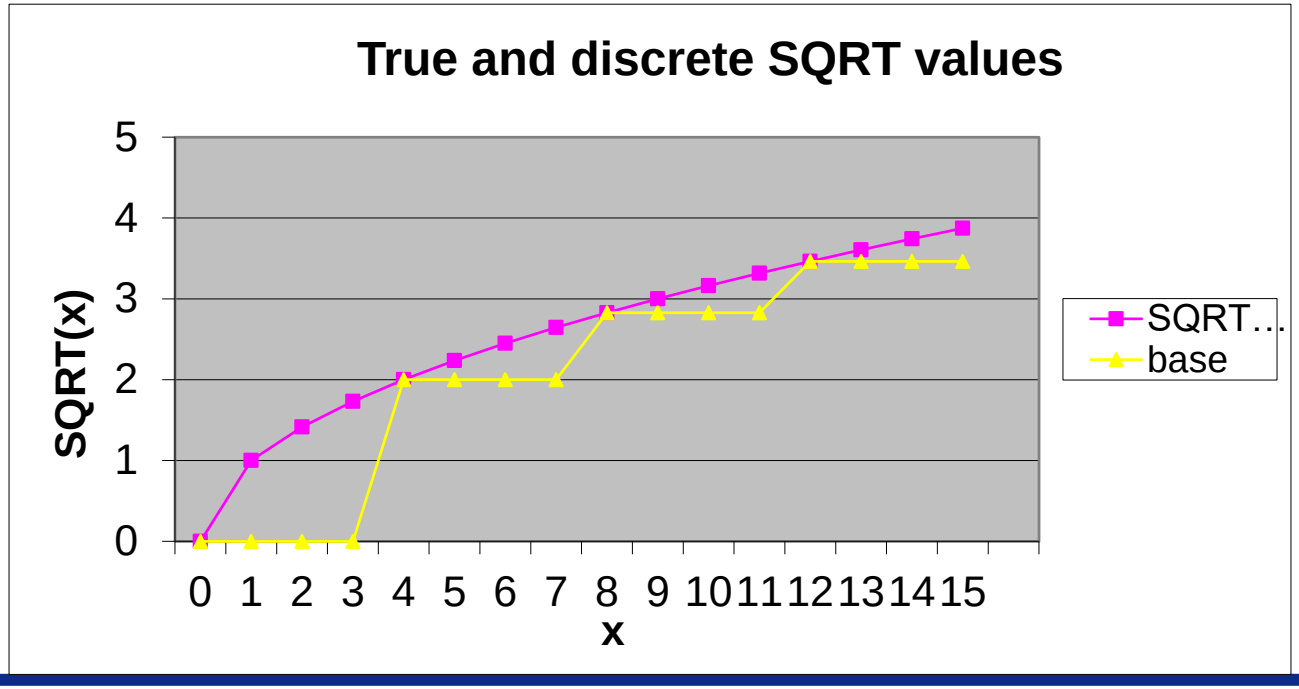
Linear Interpolation

```
//-----------------------------------------
//      A code chunk to perform linear
//      interpolation of some unknown fnc at
//      x+i where i is between 0 and 4
//      inclusive.  You are given that
//      f(x)=y              f(x+4)=z
//      You are given i, please return f(x+i)
//-----------------------------------------
        delta = (z-y)>>2;                   delta = (i*(z-y))>>2;
        f = y + delta*i;                    f = y + delta;
```

- It would be better to do the division by 4 (shift right by 2-bits) after the multiplication of delta*i because the difference (z-y) might be small and the division may result in a 0 value.

Square Root Interpolation 4 Point LUT

index	x	sqrt(x)
0	0	0
1	4	2
2	8	2.828427
3	12	3.464102
4	16	4



UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation Equation

- Let's consider how to use interpolation to find a better value for the SQRT(9). Clearly, the SQRT will be between SQRT(8) and SQRT(12).
- How much between?
- Well 1/4 of the way because 9 is a 1/4 of the way between 9 and 12.
- Have the class write an equation describing SQRT(9) in terms of SQRT(12) and SQRT(8).

SQRT(9) = SQRT(8) + 1/4*(SQRT(12)-SQRT(8))

UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation Equation

- Now write the equation replacing the 1/4 by a statement using 9,8, and 12.

SQRT(9) = SQRT(8) + (9-8)/(12-8)*(SQRT(12)-SQRT(8))

BASE + (RISE / RUN) • DELTA

UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation Spreadsheet

- Now divide the class into 16 sections to compute all the values of SQRT for 0-15.
- Compare these to the excel spreadsheet (4pt SQRT tab).
  - In this spreadsheet, I've broken the computation down so that you can see what the individual parts are doing…see next slide

UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation Spreadsheet

- **SQRT(9) = SQRT(8) + (9-8)/(12-8)*(SQRT(12)-SQRT(8))**
- **base** - this is the SQRT(8) term in the equation above. It represents the base value from which we will interpolate.
- **offset** - this is the (9-8)/(12-8) term in the equation above. It represents how much offset you are into the interval.
- **delta** - this is the SQRT(12)-SQRT(8) term in the equation above. Its how much range the function covers between the two values in the LUT.
- **base + offset*delta** - this is the value of the SQRT function for the input given in the "x" column.
- **error(x)** - this is the absolute values of the difference between the linear interpolation value and the true value of the SQRT function.

UNITED STATES AIR FORCE ACADEMY

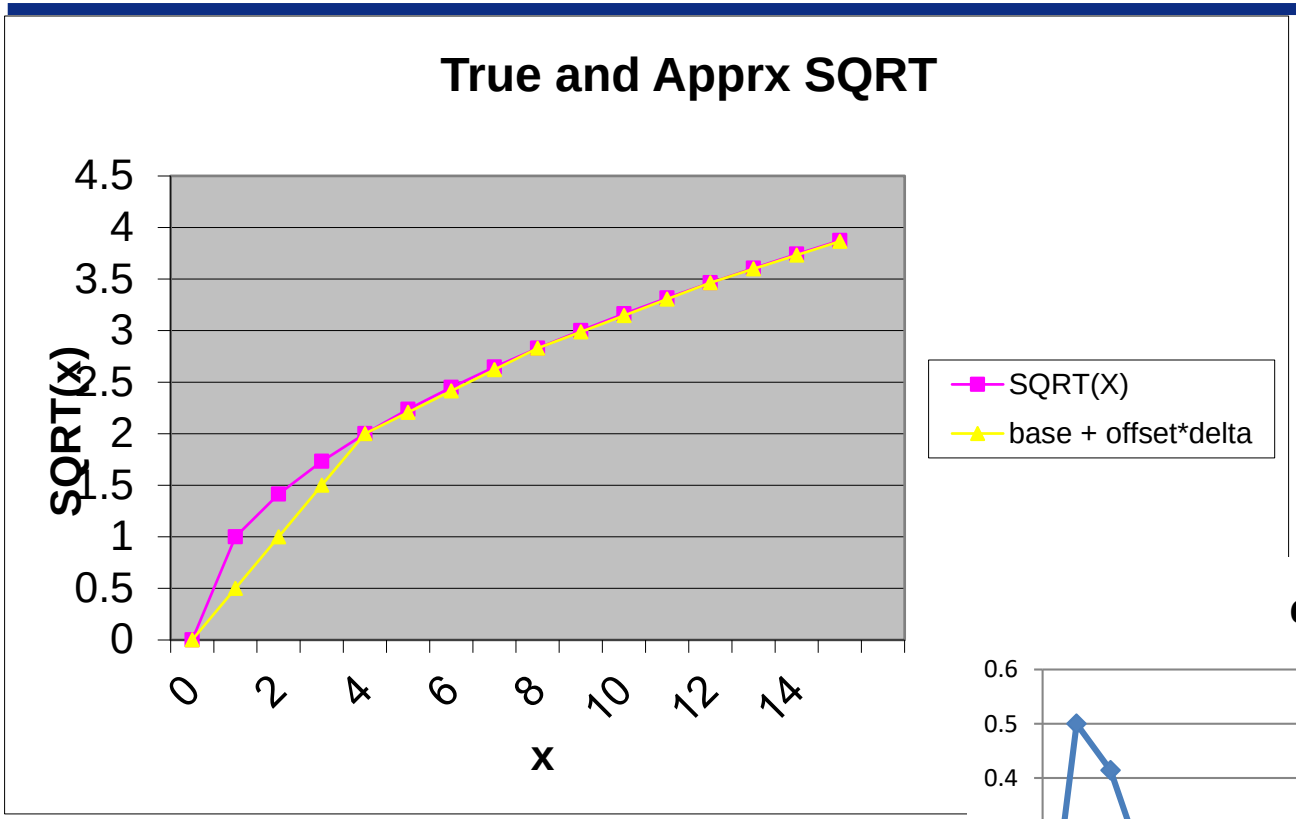
Square Root Interpolation Spreadsheet - 4 Point LUT

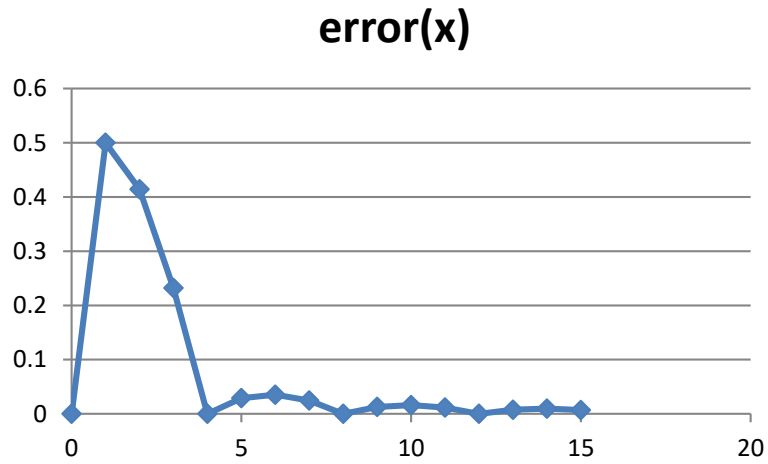
* Not 16 pt LUT

X	SQRT(X)	base	offset	delta	base + offset*delta	error(x)
0	0	0	0	2	0	0
1	1	0	0.25	2	0.5	0.5
2	1.414214	0	0.5	2	1	0.414214
3	1.732051	0	0.75	2	1.5	0.232051
4	2	2	0	0.828427	2	0
5	2.236068	2	0.25	0.828427	2.207106781	0.028961
6	2.44949	2	0.5	0.828427	2.414213562	0.035276
7	2.645751	2	0.75	0.828427	2.621320344	0.024431
8	2.828427	2.828427	0	0.635674	2.828427125	0
9	3	2.828427	0.25	0.635674	2.987345747	0.012654
10	3.162278	2.828427	0.5	0.635674	3.14626437	0.016013
11	3.316625	2.828427	0.75	0.635674	3.305182993	0.011442
12	3.464102	3.464102	0	0.535898	3.464101615	0
13	3.605551	3.464102	0.25	0.535898	3.598076211	0.007475
14	3.741657	3.464102	0.5	0.535898	3.732050808	0.009607
15	3.872983	3.464102	0.75	0.535898	3.866025404	0.006958

UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation 4 Point LUT





Red = LUT					
Blue = Interpolated					
Input			**Output**		
	index.offset				
index	**Q2.2**	**X**	**sqrt(X)**	**Q2.6**	**Hex**
0	00.00	0	0	00.000000	00
	00.01	1			
	00.10	2			
	00.11	3			
1	01.00	4	2	10.000000	80
	01.01	5			
	01.10	6			
	01.11	7			
2	10.00	8	2.8284	10.110101	B5
	10.01	9			
→	10.10	10			
	10.11	11			
3	11.00	12	3.4641	11.011101	DD
	11.01	13			
	11.10	14			
	11.11	15			
		16	4		

:BASE + offset · detta

←

UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation Data Type

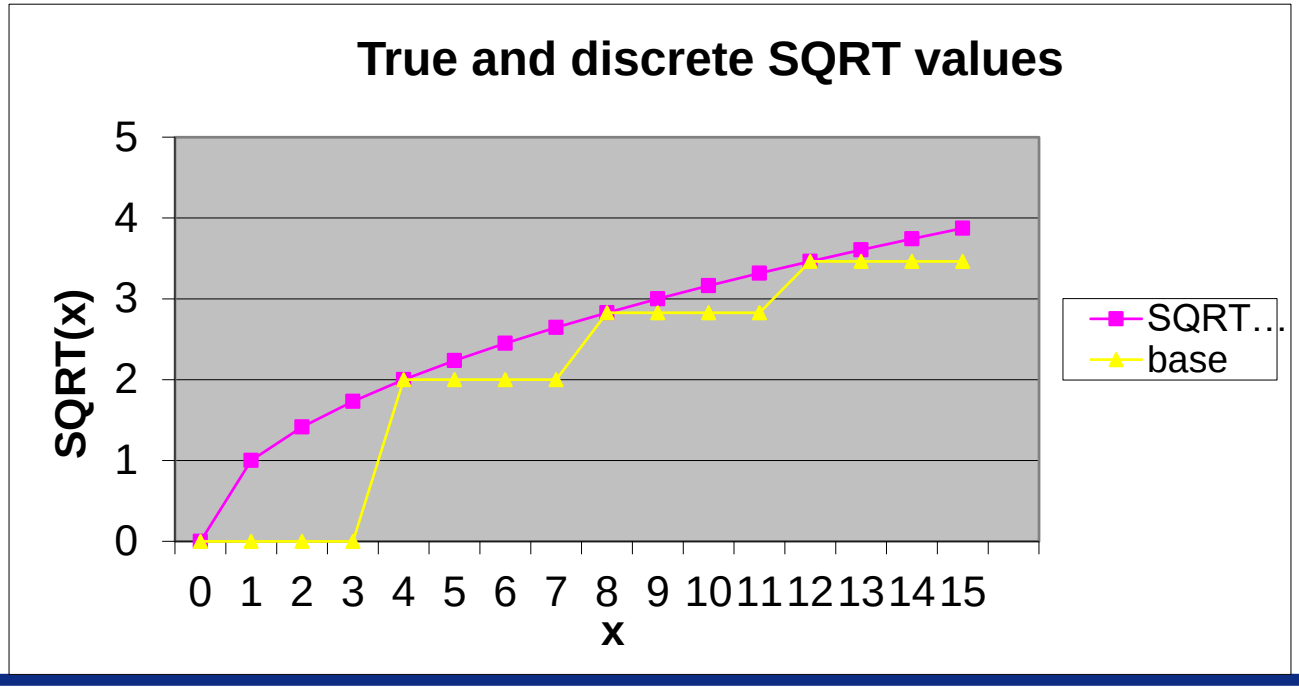
```
//-----------------------------------------
// Fnc      SQRT
// In       A 4-bit integer
// Out      An approximate SQRT 8-bit fixed point with decimal at 6th bit.
// Pur      This function computes a linearly Interpolated value for the SQRT.  The
//          5th entry in the lut is an approximation to 4.
//-----------------------------------------
int8 SQRT(int8 x) {
   int8 lut[5] = {0x00, 0x80, 0xB5, 0xDE, 0xFF};
   int8 index;
   fixed base;

   index = x >> 2;
   base = lut[index];
   offset = (x & 0x03)
   delta = lut[index+1] - lut[index];
   return(base + offset*delta)>>2;
} // end SQRT
```

UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation LUT

index	x	sqrt(x)	sqrt * 2^6	Trunc Down	2.6 Binary	2.6 Hex
0	0	0	0	0	00.000000	00
1	4	2	128	128	10.000000	80
2	8	2.828427	181.0193	181	10.110101	B5
3	12	3.464102	221.7025	221	11.011101	DD
4	16	4	256	255	11.111111	FF



UNITED STATES AIR FORCE ACADEMY

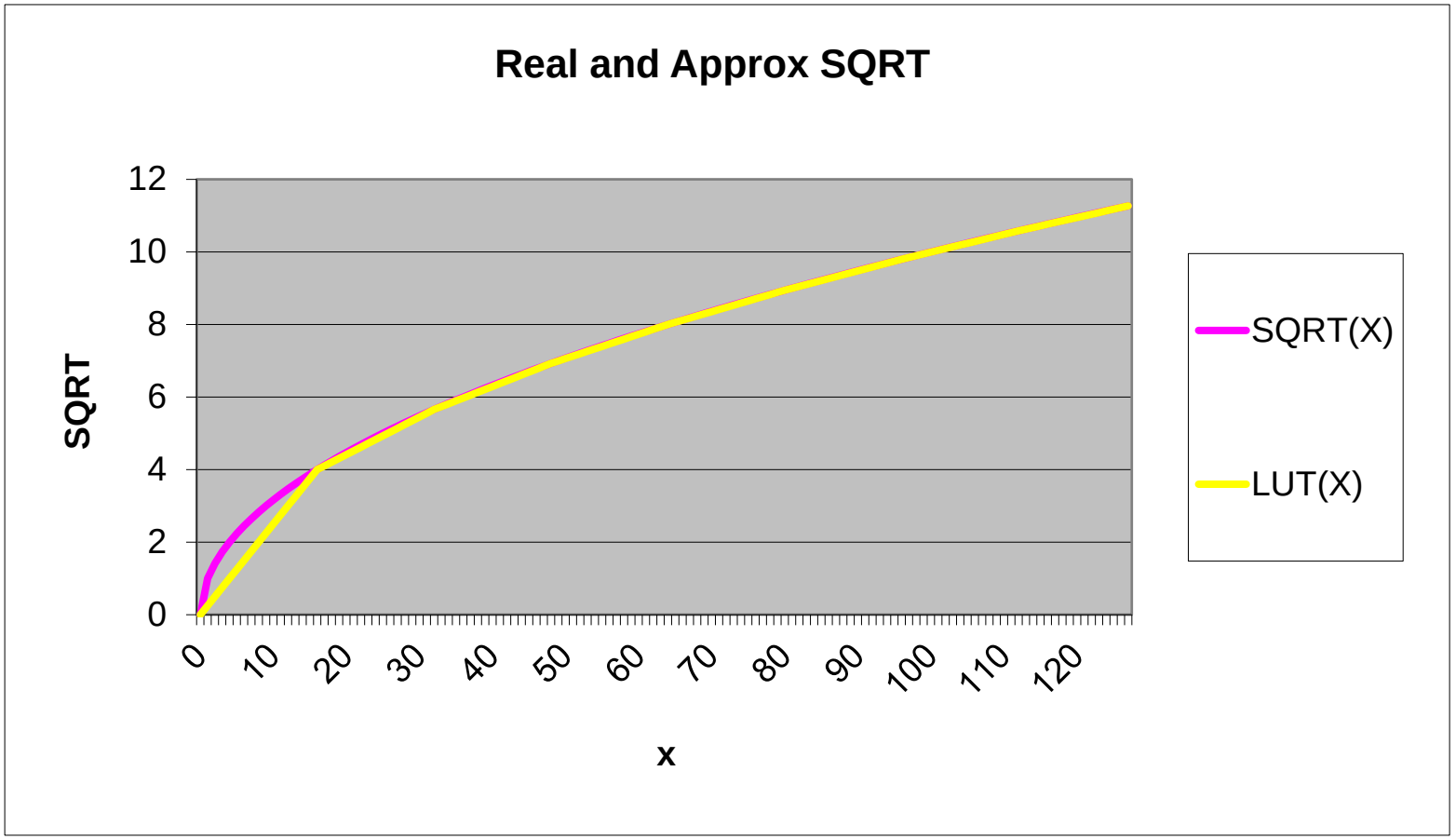
Square Root Interpolation 8 Point LUT

LUT

index	x	sqrt(x)	sqrt * 2^6	Trunc down	10.6 Binary Upper 10-Bits	Lower 6-Bits
0	0	0	0	0	0000000000	000000
1	16	4	256	256	0000000100	000000
2	32	5.656854	362.0387	362	0000000101	101010
3	48	6.928203	443.405	443	0000000110	111011
4	64	8	512	512	0000001000	000000
5	80	8.944272	572.4334	572	0000001000	111100
6	96	9.797959	627.0694	627	0000001001	110011
7	112	10.58301	677.3123	677	0000001010	100101
8	128	11.31371	724.0773	724	0000001011	010100

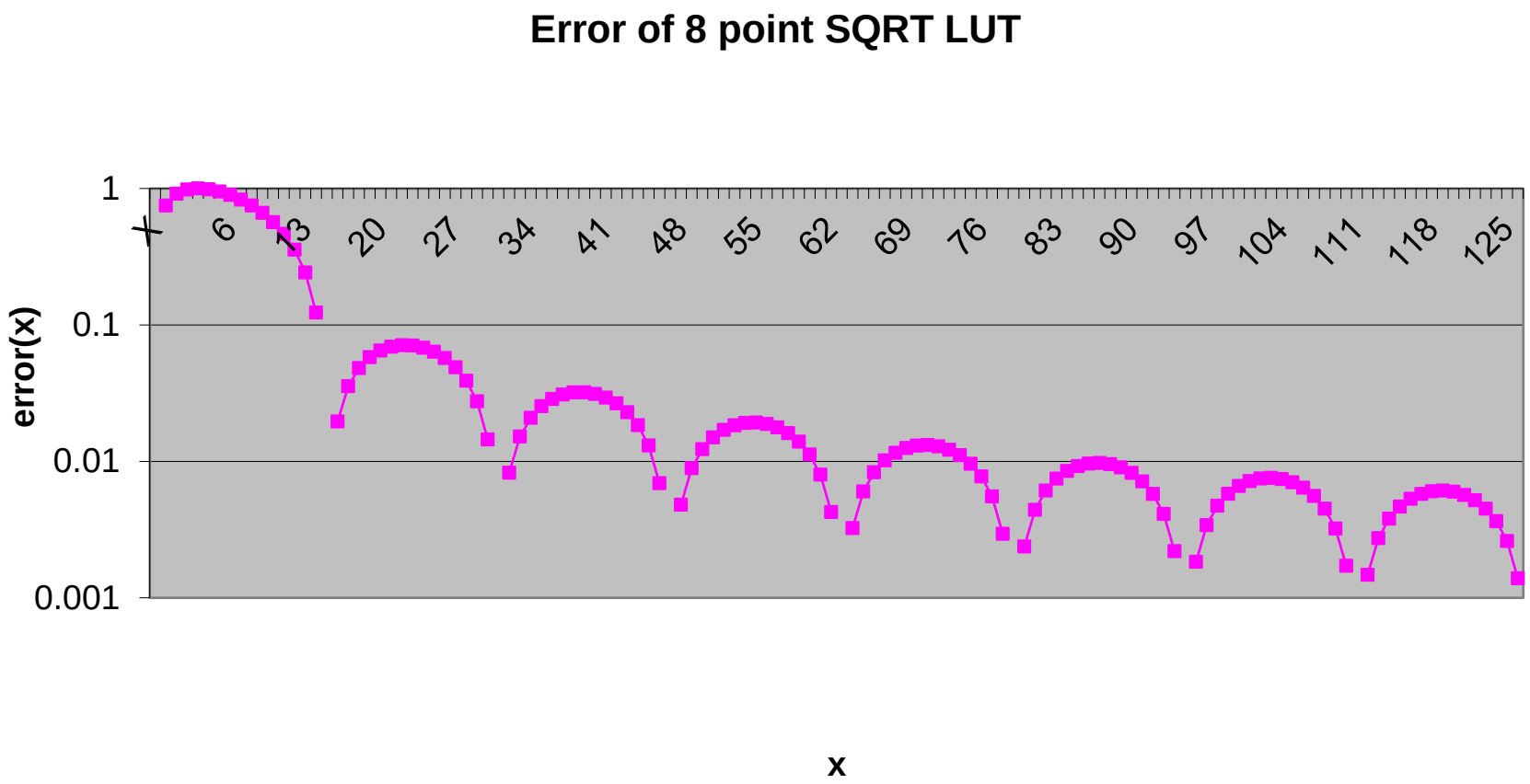
UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation 8 Point LUT



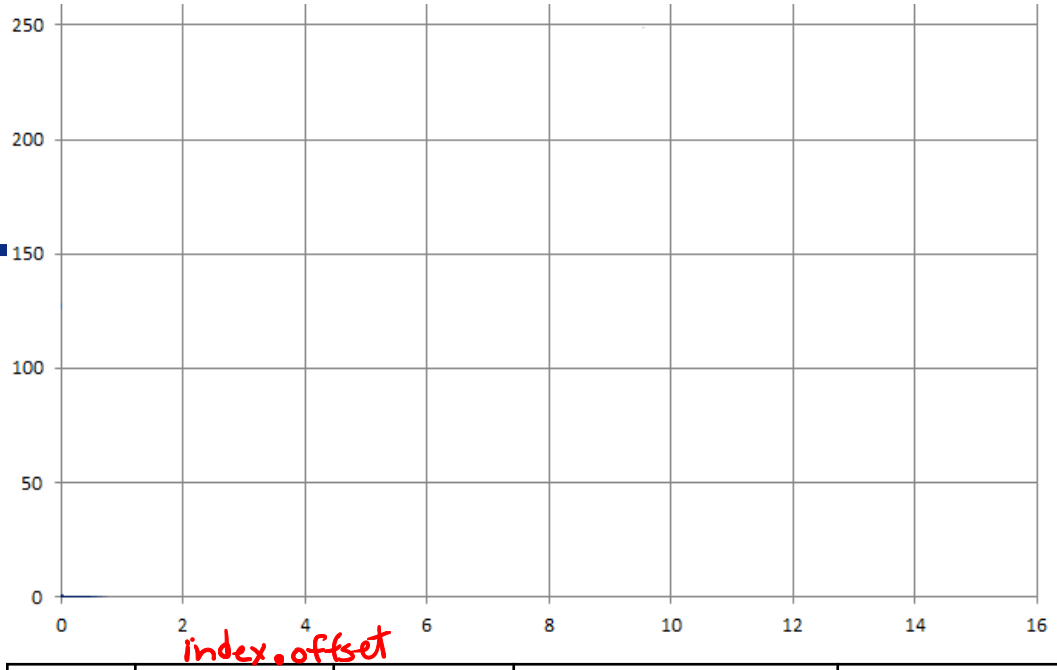
UNITED STATES AIR FORCE ACADEMY

Square Root Interpolation LUT



UNITED STATES AIR FORCE ACADEMY

Practice



Phase Increment
X = 0001.1011

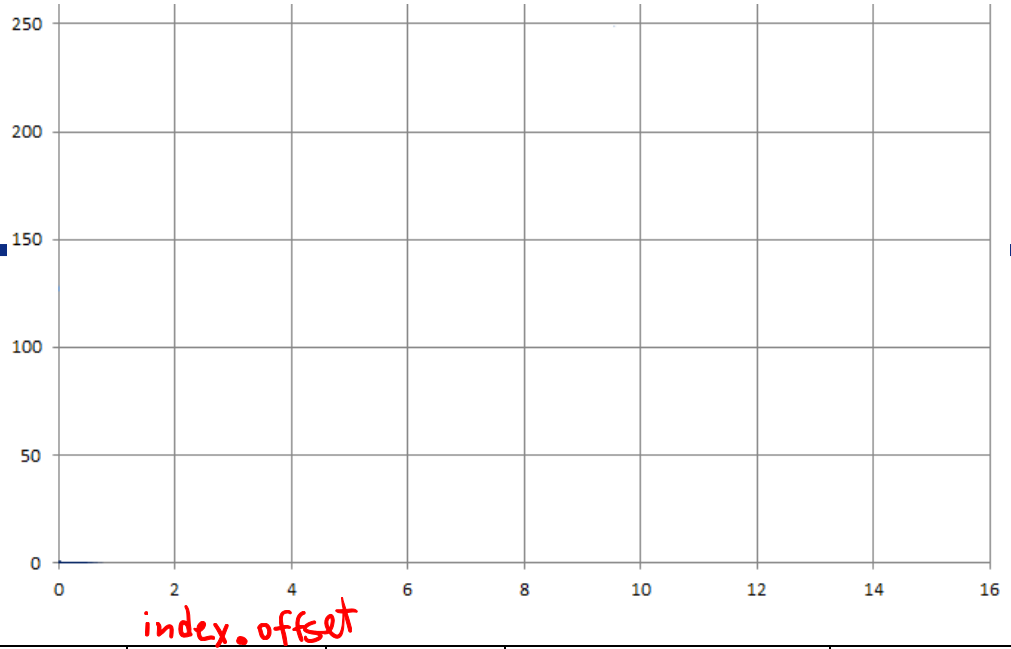
index . offset

0	127
1	175
2	216
3	244
4	253
5	244
6	216
7	175
8	127
9	78
10	37
11	9
12	0
13	9
14	36
15	78

Time	Index	Base	Delta	Offset*Delta	Base + Offset*Delta
0	0000.0000	127	175-127 = 48 = 110000	0.0000*110000 = 00000.0000	127+0 = 127
1	0001.1011	175	216-175 = 41 = 101001	0.1011*101001 = 11100.0011	175+28 = 203
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					

UNITED STATES AIR FORCE ACADEMY

Solution



x = 0001.1011

index.offset

0	127
1	175
2	216
3	244
4	253
5	244
6	216
7	175
8	127
9	78
10	37
11	9
12	0
13	9
14	36
15	78

Time	Index	Base	Delta		Offset*Delta		Base + Offset*Delta
0	0000.0000	127	175-127 = 48 = 110000		0.0000*101010 = 00000.0000		127+0 = 127
1	0001.1011	175	216-175 = 41 = 101001		0.1011*101001 = 11100.0011		175+28 = 203
2	0011.0110	244	9	00001001	3.375	0011.0110	247
3	0101.0001	244	-28	1111100100	-1.75	111110.0100	242
4	0110.1100	216	-41	1111010111	-30.75	100001.0100	185
5	1000.0111	127	-49	1111001111	-21.4375	101010.1001	106
6	1010.0010	37	-28	1111100100	-3.5	111100.1000	34
7	1011.1101	9	-9	1111110111	-7.3125	111000.1011	2
8	1101.1000	9	27	00011011	13.5	1101.1000	23
9	1111.0011	78	49	00110001	9.1875	1001.0011	87
10	0000.1110	127	48	00110000	42	0000101010.0000	169
11	0010.1001	216	28	00011100	15.75	1111.1100	232
12	0100.0100	253	-9	1111110111	-2.25	111101.1100	251
13	0101.1111	244	-28	1111100100	-26.25	100101.1100	218
14	0111.1010	175	-48	1111010000	-30	1000100000	145

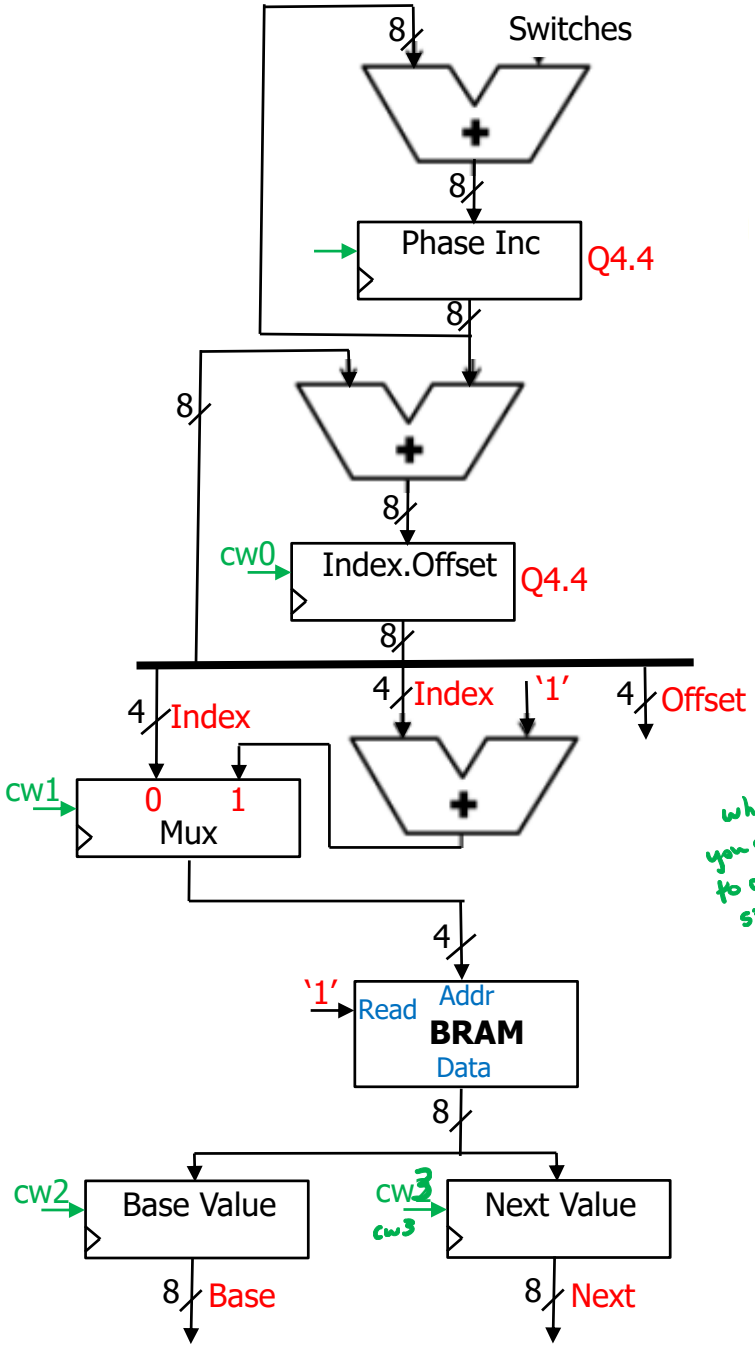
Lut Index	Lut Value
0	127
1	175
2	216
3	244
4	253
5	244
6	216
7	175
8	127
9	78
10	37
11	9
12	0
13	9
14	36
15	78

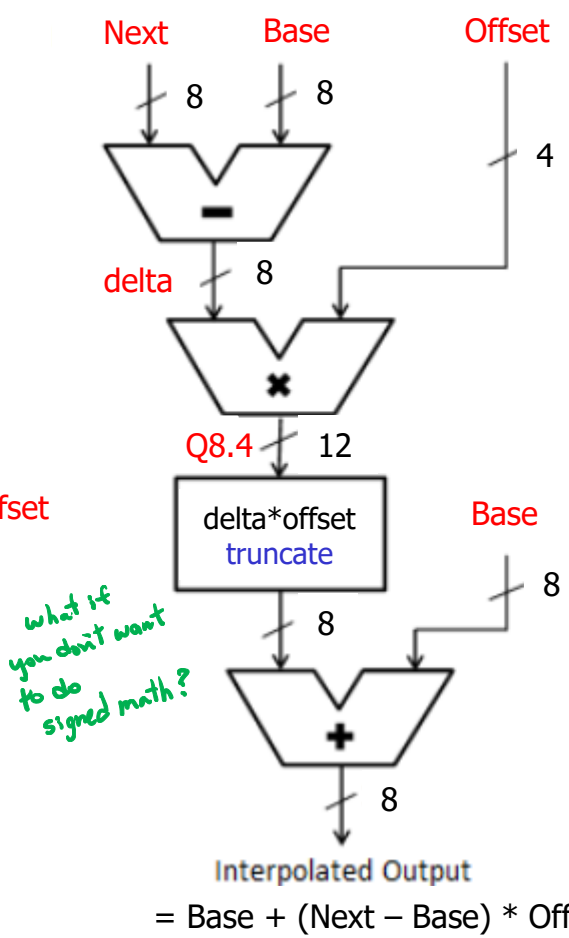
Time	Bin Index	Dec Index	Trunc Dec Index	Base			Delta	Binary Delta
0	00000000	0	0	127	175	127	48	175-127 = 42 = 101010
1	00011011	1.6875	1	175	216	175	41	216-175 = 41 = 101001
2	00110110	3.375	3	244	253	244	9	00001001
3	01010001	5.0625	5	244	216	244	-28	1111100100
4	01101100	6.75	6	216	175	216	-41	1111010111
5	10000111	8.4375	8	127	78	127	-49	1111001111
6	10100010	10.125	10	37	9	37	-28	1111100100
7	10111101	11.8125	11	9	0	9	-9	1111110111
8	11011000	13.5	13	9	36	9	27	00011011
9	11110011	15.1875	15	78	127	78	49	00110001
10	00001110	0.875	0	127	175	127	48	00110000
11	00101001	2.5625	2	216	244	216	28	00011100
12	01000100	4.25	4	253	244	253	-9	1111110111
13	01011111	5.9375	5	244	216	244	-28	1111100100
14	01111010	7.625	7	175	127	175	-48	1111010000

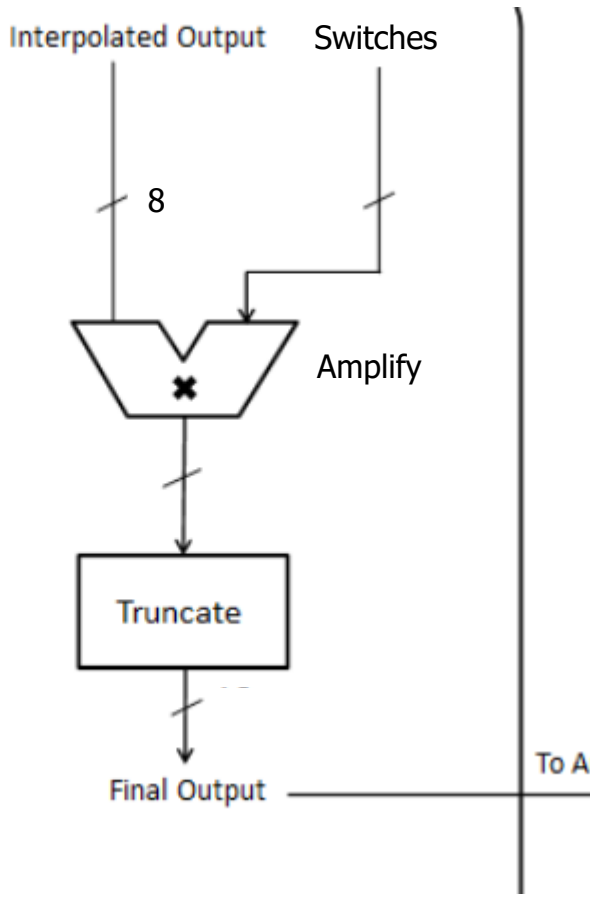
more columns in X/S →

Offset	Bin Offset	Offset*Delta	Offset*Delta	Base + Offset*Delta	Base + Offset*Delta	Final Value
0	0000	0	0.0000*101010 = 00000.0000	127	127+0 = 127	127
0.6875	1011	28.1875	0.1011*101001 = 11100.0011	203.1875	175+28 = 203	203
0.375	0110	3.375	00110110	247.375	247	247
0.0625	0001	-1.75	1111100100	242.25	242	242
0.75	1100	-30.75	1000010100	185.25	185	185
0.4375	0111	-21.4375	1010101001	105.5625	105	106
0.125	0010	-3.5	1111001000	33.5	33	34
0.8125	1101	-7.3125	1110001011	1.6875	1	2
0.5	1000	13.5	11011000	22.5	22	23
0.1875	0011	9.1875	10010011	87.1875	87	87
0.875	1110	42	00001010100000	169	169	169
0.5625	1001	15.75	11111100	231.75	231	232
0.25	0100	-2.25	1111011100	250.75	250	251
0.9375	1111	-26.25	1001011100	217.75	217	218
0.625	0000	-30	1000100000	145	145	145

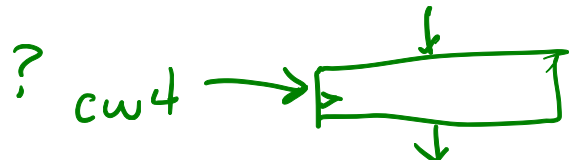
Interpolation Block Diagram



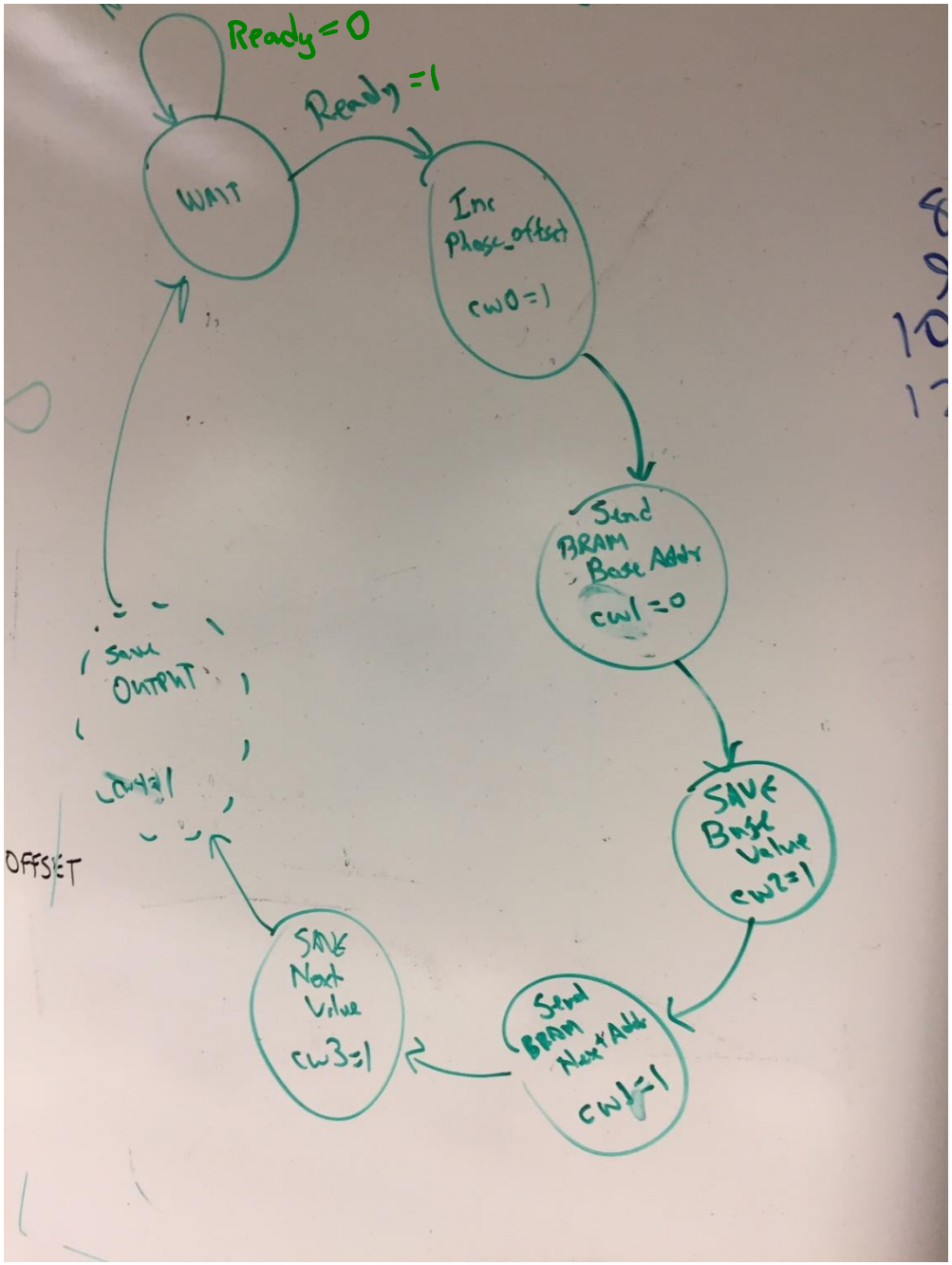




= Base + (Next – Base) * Offset



State Machine?



BRAM

```
--------------------------------------------------------------------------------
--    BRAM_SDP_MACRO:  Simple Dual Port RAM 7 Series
--    Source:          Xilinx HDL Libraries Guide, version 2012.4
--    Link:            https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug953-vivado-
                       7series-libraries.pdf
--    Page:            10
--------------------------------------------------------------------------------

sampleMemory: BRAM_SDP_MACRO
      generic map (
            BRAM_SIZE => "18Kb",                -- Target BRAM, "18Kb" or "36Kb"
            DEVICE => "7SERIES",                -- Target device: "VIRTEX5", "VIRTEX6", "SPARTAN6", "7SERIES"
            DO_REG => 0,                        -- Optional output register disabled
            INIT => X"0000000000000000",        -- Initial values on output port
            INIT_FILE => "NONE",                --
            WRITE_WIDTH => 16,                  -- Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
            READ_WIDTH => 16,                   -- Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
            SIM_COLLISION_CHECK => "NONE",      -- Simulation collision check
            SRVAL => X"0000000000000000",       -- Set/Reset value for port output
            INIT_00 => X"8BC28AF98A31896988A087D8870F8647857E84B583EC8323825A819180C88000",
            INIT_01 => X"9830976A96A595DF9519945293 8C92C591FE913790708FA98EE18E198D528C8A",
            INIT_02 => X"A462A3A2A2E0A21FA15DA09B9FD89F169E529D8F9CCB9C079B439A7F99BA98F5",
            INIT_3E => X"7247717F70B76FF06F296E616D9A6CD46C0D6B476A8069BA68F5682F676A66A5",
            INIT_3F => X"7ECF7E067D3D7C747BAC7AE37A1A7951788977C076F7762F7567749F73D6730E")
      port map (
            DO => DO,                           -- Output read data port, width defined by READ_WIDTH parameter
            RDADDR => vecAddrRead,              -- Input address, width defined by port depth
            RDCLK => clk,                       -- 1-bit input clock
            RST => reset,                       -- active high reset
            RDEN => '1',                        -- read enable
            REGCE => '1',                       -- 1-bit input read output register enable - ignored
            DI => DI,                           -- Dummy write data - never used in this application
            WE => "00",                         -- write to neither byte
            WRADDR => "0000000000",             -- Dummy place holder address
            WRCLK => clk,                       -- 1-bit input write clock
            WREN => '0');                       -- we are not writing to this RAM
```

who makes this?

signed or unsigned?

2nd value

1st 16-bit value

skip

index

only Read!

- **See example spreadsheet for creating BRAM LUT (or use python, java, matlab…)**
- **Block Diagram?**
- **Intro to lab#4**

UNITED STATES AIR FORCE ACADEMY

Lab4 Intro