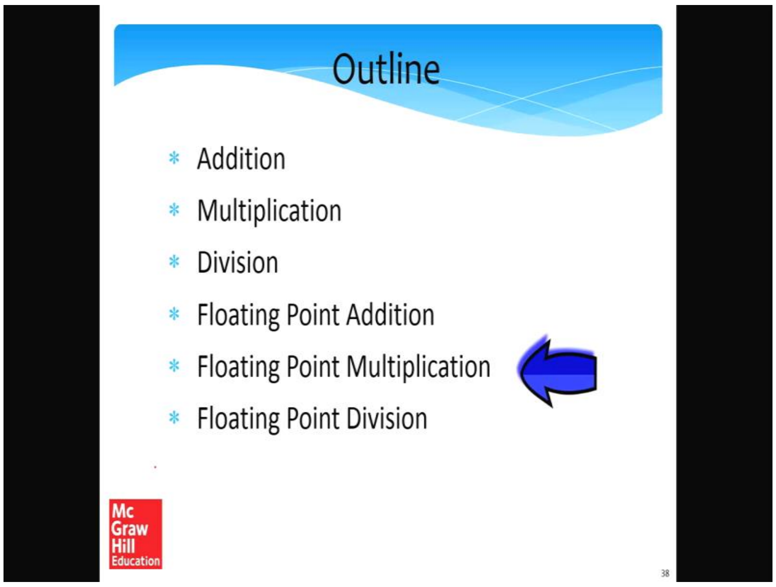
Computer Architecture

Prof. Smruti Ranjan Sarangi
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Lecture – 22
Computer Arithmetic Part-VI

(Refer Slide Time: 00:26)

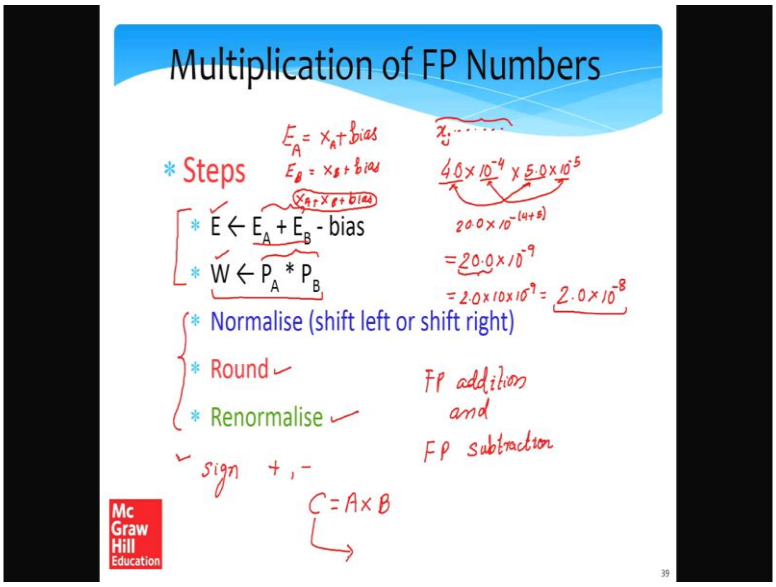


Outline

* Addition
* Multiplication
* Division
* Floating Point Addition
* Floating Point Multiplication
* Floating Point Division

Let us now take a look at floating point multiplication and division. So, floating point multiplication is along the same lines as floating point addition and subtraction. A little bit of a difference nothing much, but floating point division is different.

(Refer Slide Time: 00:42)



Multiplication of FP Numbers

- Steps
  - $E \leftarrow E_A + E_B$ - bias
  - $W \leftarrow P_A * P_B$
  - Normalise (shift left or shift right)
  - Round
  - Renormalise

$E_A = x_A + bias$
$E_B = x_B + bias$
$x_A + x_B + bias$
$x_j \cdots$
$4.0 \times 10^{-4} \times 5.0 \times 10^{-5}$
$20.0 \times 10^{-(4+5)}$
$= 20.0 \times 10^{-9}$
$= 2.0 \times 10 \times 10^{-9} = 2.0 \times 10^{-8}$
FP addition and FP subtraction
sign +, -
$C = A \times B$
39

So, let us first take a look at floating point multiplication which is easy. So, let me motivate by showing an example in decimal. So, let us assume that we have this convention that any you know decimal significant will be at the form x point some numbers some digits and there is basically one digit to the left of the decimal point. So, this will actually be 4.0 you can actually draw that.

So, this is the convention that we are using. We can multiply 2 numbers of the type 4.0 times let us say 10 to the power minus 4 times 5.0 times 10 to the power minus 5.0. So, what we would do is that first we will multiply the 2 significant 4 and 5. So, this will give us 20.0. And then what we will do is that we will multiply these 2 you know exponent terms. So, we will basically add the exponents and add minus 4 and minus 5. So, what we will get is 20.0 times 10 to the power of minus 9. So, note that after the multiplication the result here that was obtained by multiplying the significance, is not in the normal form. So, to bring it to the normal form what we can do is that we can do to make it 2.0 times 10 times 10 raised to the power of minus 9 which is 2.0 times 10 raised to the power of minus 8. So, then this comes back to the normal form. So, we will do something very similar with binary numbers over here. So, what we will do is that we will add the exponents, right. So, because this is a multiplication we will add the exponents.

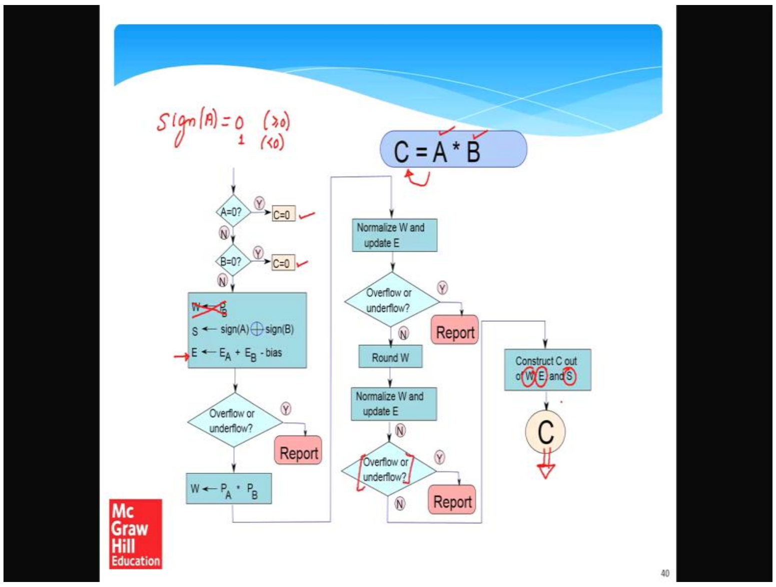
So, recall that the E field is actually the real exponent right plus the bias. So, if I then have E A E A is x A plus bias similarly if I have E B, E B is a real exponent x B plus a bias see if I add E A plus E B then the bias will get added twice and that will definitely be wrong. So, the final result that we want is x A plus x B plus bias, but since E A plus E B will add the bias twice, what we do is we add E A and E B and then subtract the bias. So, then the final result that we will get is essentially the some of the exponents represented in the bias notation. So, this will be the final exponent E. And then what we will do is we will compute temporary value W which is the product of the significant. So, the same way multiplied 4.0 times 5.0. So, whatever are the significant we will multiply them.

After that we need to normalize it, in the sense that if the significant is greater than equal to 2 then what we need to do, you know is that we need to you know shift the number to the right and to and also adjust the decimal point. So, this normalization has to be done to ensure that you know all that numbers are in the normal form. Then you know if some shifting has been done some amount of rounding can be done not can be done has to be done because if some bits are being thrown out, we want the remaining significant to reflect their effect at least.

Rounding can cause further problems. So, we have to renormalize. So, mind you these steps are exactly the same as floating point addition and subtraction. So, they are exactly the same as we had in floating point addition and subtraction. So, I am not you know discussing them. So, all these steps remain the same. The only steps that are different is that we add the exponents. And since we would be adding the exponents in the biased form we will be essentially adding the bias twice. So, the bias at least one of them needs to be subtracted. Second we need to multiply the significant. And the rest of the stages remain the same. So, the complexity of this is same as the time taken is the same as the time for add subtract and multiply, which is all in login time then one more thing that I did not mention in the sign of the result. So, well if both the number are the same sign the sign of the result is positive. So, any both of the numbers have opposite sign the sign of the result is negative.

So, once we have the sign of the result, we have the new significant which has been normalized and the new exponent we can construct the result. So, we can give given 2 floating point numbers with the multiply operation, we can construct the result and we can return it all right.

(Refer Slide Time: 06:40)

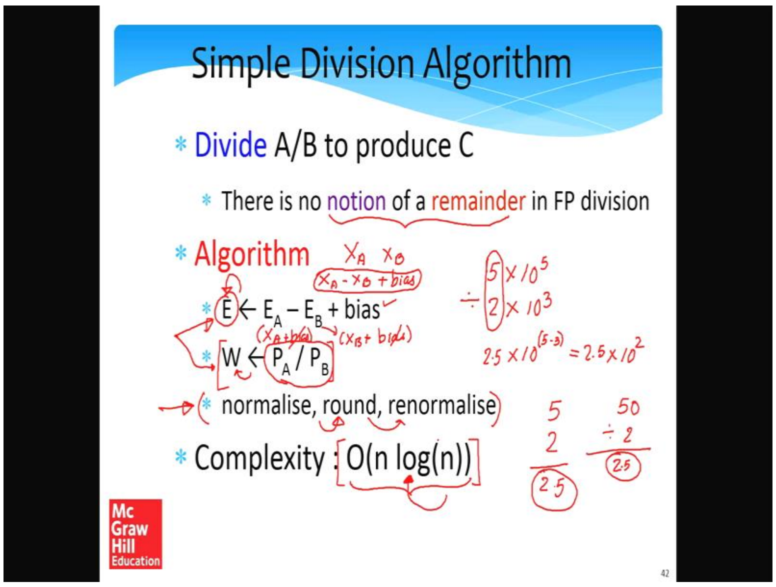


So, let me now summarize the same pro process the same procedure with a help of a flow chart. So, let us assume that we are multiplying 2 floating point numbers A and B and the product of the numbers will be saved in C. So, if A is first let us do some very simple checks, if A is equal to 0 then we can set C to 0 and we can get out. Else if B is equal to 0 we can set C to 0 and we can get out. Otherwise some amount of additional work is required. So let us see what is the additional work like sorry this is an extra step that needs to be removed to. So first thing that we do is we compute the sign of the result. So, what we essentially do is that we will compute the sign of the result here and the sign of the result is an exclusive art of the signs of A and B. See both of them have the same sign right. So basically the way that we are representing the sign of A or B is by the sign bit say the sign is 0 means that the number is greater than or equal to 0, if the sign bit is one it means that the number is less than 0.

See as you can see if both the numbers have the same sign their exclusive or is 0. So, a number is positive. If they have different signs the exclusive or is 1 and the number is negative. So, the sign bit is 1. Then we compute the final exponent E which is a plus E B minus the bias. And after that we see if there is an overflow or underflow because you know we are increasing the exponent. So, if we have exceeded the range of the number system we report this and we exit. Otherwise we multiply the significant P A times P B. And save it in W. We normalize W and update the exponent.

Again there is a possibility overflow underflow if that is the case we report. Otherwise we do some rounding, then we renormalize and update the exponent. Then if there is you know then again there is a possibility of us exceeding the range of the number system. If that has happened, we report otherwise we move to constructing the result. So, given the new significant, given the new exponent and you know the computer signed bit we can combine all of these 3 pieces of information, to create the result or the product C which should be returned you know, which is pretty much the output of this particular circuit.

(Refer Slide Time: 09:42)



Simple Division Algorithm

- Divide A/B to produce C
  - There is no notion of a remainder in FP division
- Algorithm
  - $E \leftarrow E_A - E_B + bias$
  - $W \leftarrow P_A / P_B$
  - normalise, round, renormalise
- Complexity : $O(n \log(n))$

X_A X_B
$(X_A - X_B + bias)$
$(X_A + bias)$ $(X_B + bias)$
5×10^5
$\div 2 \times 10^3$
$2.5 \times 10^{(5-3)} = 2.5 \times 10^2$
5 / 2 / 2.5
50 ÷ 2 / 2.5
42

Now let us take a look at floating point division. Let us first take a look at a simple division algorithm. So, let us divide A by B to produce C. So, the first difference between floating point division and regular integer division is that in floating point division there is no notion of a remainder. There is nothing called a remainder in floating point division. Because in integer division we had a remainder, the reason was that our quotient could only be an integer, but in floating point division the quotient can be a floating point itself. So, there is no notion of a remainder this makes our job easy. After this let us try to design a very simple division algorithm, the same way we have division in decimal. Say in decimal let us say that we want to divide 2 numbers. So, what we do is that. So, we can let us say you know consider an example let us try to divide. 4 divided by let us see instead of 4 let me make it 5.

So, what we do is that we will first divide the significant. So, 5 divided by 2 is 2.5. Then what we will do is we will adjust. So, we will basically compute the new exponent which is 10 to the power 5 minus 3. So, in in multiplication we add the exponents and division we subtract the exponents. So, what we get is 2.5 times 10 to the power 2. So, this is exactly what we need to do in division. We need to subtract the exponents and we need to divide the significant. After that the rest of the steps will be the same. So, let us first take a look at subtracting the exponents. So, let X A and X B be the real exponents. So, they are represented in their biased form. So, what we need to do is if you are dividing a by b we need to compute X A minus X B. Since they are stored in the biased form they will be stored as X A minus X B plus bias. So, recall that E A is X A plus bias.

Similarly, E B is X B plus bias you know plus the bias right. So, a minus E B is same as X A minus X B because a bias gets cancelled out. That is the reason we need to add the bias once again to the result of the subtraction to compute the final exponent which is saved in biased form all right. So, what do we do once again, we extract the exponent field from A and B. We subtract them and then we add the bias. So, why do we add the bias, the reason we add the bias is because the E fields are essentially stored in biased notation themselves and they are basically equal to the real exponent plus the bias all right.

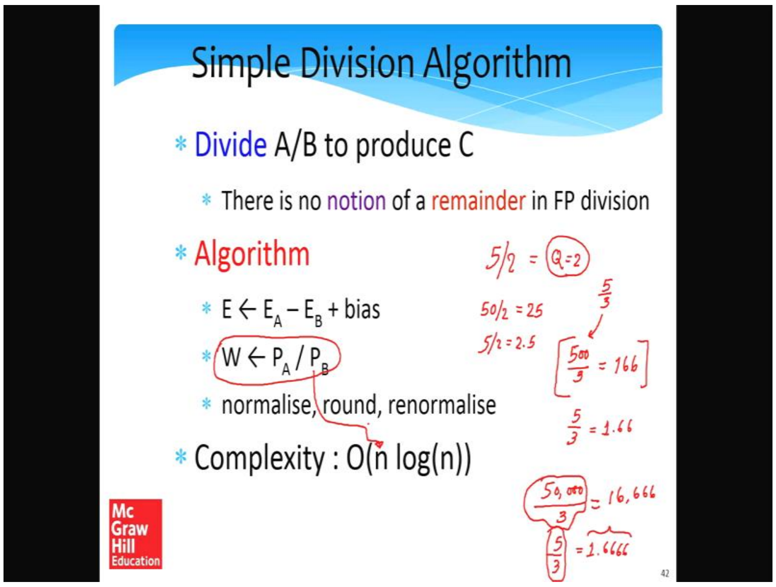
So, once we subtract E A minus E B the bias gets subtracted, but since the final E field also needs to be saved in bias notation, we need to add the bias once again. Such that what we are essentially encoding is X A minus X B plus the bias subsequently what we do over here is that we divide the significant. So, we divide P A by P B and we save that in the result W right. So, E and the W is a standard convention that we have been using up till now.

After this what we do is we normalize the result that we have to do. If some bits were discarded, we do some rounding and then we renormalize. So these 3 steps will always be common in all floating point algorithms. It is not just addition or subtraction it is also multiplication division and anything else that the final output has to be normalized. After that if some bits are being discarded there is a need for rounding and if that again causes the number to be in a non-normal form it needs to be renormalized once again. And in the book you will find a proof that renormalizing only once is sufficient you do not have to renormalize twice all right. So I will not discuss the specifics, but it is just that these 3 steps will be common in all algorithms.

So, what is the complexity of this? So, the complexity of this entire algorithm is same as the complexity of the division here which is the most complicated operation and what we have seen is we have seen 2 algorithms for you know integer division. So, mind you this division is an integer division. And we have seen 2 algorithms for integer division and they take order of n log in time all right. So this is basically here they take order of n log n time. So, actually in this case it is not exactly integer division because when you are dividing 5 by 2 to get 2.5 we are not doing integer division rather what you are doing is that you are dividing 50 divided by 2. So, then we are getting 25 and then you are putting the decimal point between 2 and 5.

So, this is what we have learned in our you know primary school, that to divide let us say you know one number be another to increase the precision you can left shift the dividend by certain positions, and then right shift the final quotient right. So, that will give us the final quotient. So, that will give us the floating point division, but essentially what we are doing is we are dividing numbers, I will have bit we are increasing the precision by first left shifting the dividend and then right shifting the quotient right. The same way that we you know divide 2 numbers to get decimal output the same primary school algorithm that can be done, but in any case the integer division algorithm set we have learnt take order of n log n time. So, they are slow.

(Refer Slide Time: 16:44)



Simple Division Algorithm

- Divide A/B to produce C
  - There is no notion of a remainder in FP division
- Algorithm
  - $E \leftarrow E_A - E_B + bias$
  - $W \leftarrow P_A / P_B$
  - normalise, round, renormalise
- Complexity : O(n log(n))

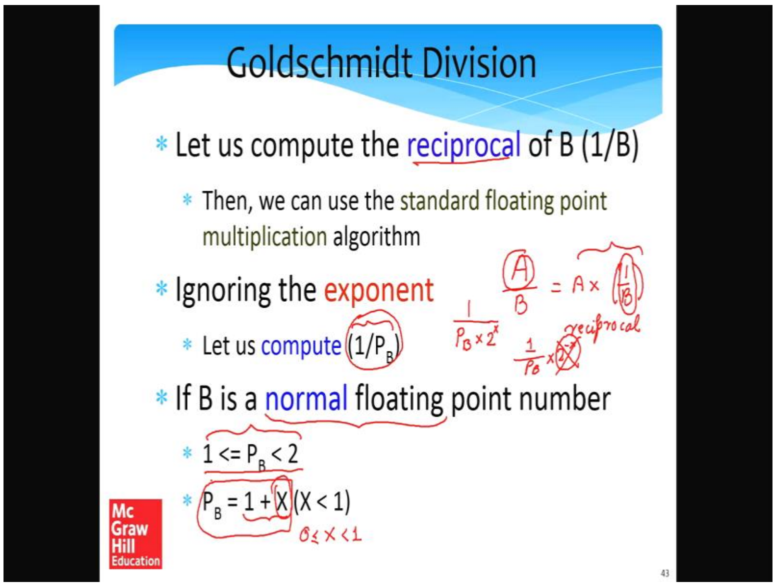
So, it turns out that we can design algorithms which are much faster and which are better in terms of their properties. So, let us look at them next, but the summary of this particular slide is that the final result is pretty much coming out of a process of division and this is the slowest operation. Because we are essentially using integer division I will bit in a slightly different fashion and so that is the reason, but anyway this is a slow operation takes n log n time, using both the restoring and non-restoring algorithms that is the reason the overall algorithm is slow. And what is the slightly different fashion.

So, let us say if I want to divide 5 by 2, I can always say that look the quotient is equal to 2 it is regular integer division, but what we have learnt in school in primary school is slightly different. If I want to divide 5 by 2 what I actually end up doing is I divide 50 by 2. So, I get 25, then I realize that since I had multiplied 5 by 10, I will sort of you know compensate for that. So, if 50 by 2 is 25, I will say that 5 by 2 is 2.5. And let us say if I want more precision. So, I want to divide 5 by 3 what I can instead do if I need you know more precision in the output is that, I will first compute what is 500 by 3. So, 500 by 3 in regular integer division is 166.

So, I will say that look since 500 by 3 is 166 5 by 3 is 166 divided by 100, which is 1.66. So, similarly instead of 500 by 3 I could have done 5000 by 3. And then or I could have done 50000 divided by 3. So, this would have given me 16,666. Then I could have said that look now 5 by 3 is 1.666. So, we increase the number of digits after the decimal point. So, we increase the precision of the result so, but what I am doing in each and every step is regular integer division. So, I am first computing the result of one more division, for the dividend has been shifted to the left and then to compute 5 by 3, I am inserting the decimal point at the right and appropriate location.

So, the same thing can be done over here this is not integer regular integer division, I can shift let us say this significant of a, by certain places to the left perform the division and then shift the quotient that many places to the right. So, that has not been mentioned to avoid cluttering of the slide, but this is pretty much what I would do, you know what I do anyway for normal division right normal primary school division that also can be done, but in any case the long and short of it is that this is the slow operation. So, I can do something which will make all my operations much faster.

(Refer Slide Time: 20:03)



Goldschmidt Division

- Let us compute the reciprocal of B (1/B)
  - Then, we can use the standard floating point multiplication algorithm
- Ignoring the exponent
  - Let us compute $(1/P_B)$
- If B is a normal floating point number
  - $1 <= P_B < 2$
  - $P_B = 1 + X \; (X < 1)$

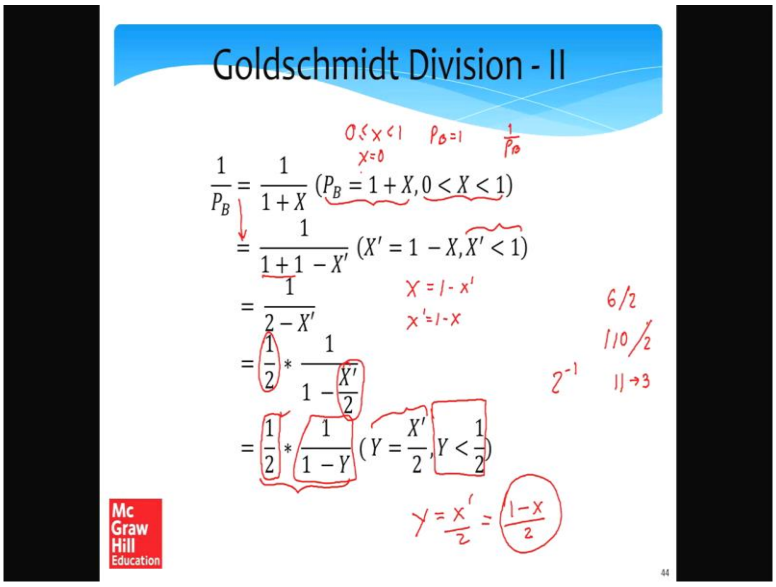
$\frac{A}{B} = A \times \left(\frac{1}{B}\right)$
reciprocal
$\frac{1}{P_B \times 2^x}$
$\frac{1}{P_B} \times \cancel{2^x}$
$0 \leq X < 1$
43

So, let us look at goldsmith division. So, we will talk about 2 division algorithms. So, one is goldsmith division. So, my basic insight is like this. Let us say I want to compute A divided by B. So, A divided by B is A multiplied with 1 by B I have a fast algorithm for multiplication. So, pretty much my problem reduces to finding what is 1 by B. So, 1 by b is known as the reciprocal. So, if I have a fast algorithm for finding the reciprocal of a number then what I can do is I can multiply that you know multiply 1 by B with A and compute A by B. So, let us thus try to compute the reciprocal of B which is 1 by B.

Then we can use standard multiplication, also if B is represented in the normal form which is P B times in the 2 to the power of X which is the exponent then you know I can compute 1 by P B I mean 1 by the significant times 2 raised to the power minus 6. So, this part is simple this can be ignored. So, I can just take the significant and try to compute 1 by P B which is the core part of the operation.

So, let us first consider the case of a normal floating point number and of course, if B is a denormal number we can do something similar, but I will not you know discuss that in interest of time, but a very easy extension can be done to handle denormal numbers as well. So, let me just consider the case of normal significant, where a normal floating point significant is between 1 and 2. In the sense that 1 is less than equal to P B which is less than 2. So, if this is the case, I can represent the floating point significant P B as 1 plus X, 1 plus X where X is less than 1. So, P B is between 1 and 2. So, since I can thus represent it as something in the form 1 plus X where 0 is less than equal to X and X itself is less than 1. So, this can be done. Given the fact that this can be done, let us now do some more algebra.

(Refer Slide Time: 22:35)



Goldschmidt Division - II

$0 \leq x < 1$ $\quad P_B = 1$ $\quad \frac{1}{P_B}$

$x = 0$

$$\frac{1}{P_B} = \frac{1}{1+X} \left(\underbrace{P_B = 1 + X}, \underbrace{0 < X < 1}\right)$$
$$= \frac{1}{\underline{1+1} - X'} \left(X' = 1 - X, \overbrace{X' < 1}\right)$$
$$= \frac{1}{2 - X'}$$
$$= \boxed{\frac{1}{2}} * \frac{1}{1 - \boxed{\frac{X'}{2}}}$$
$$= \underbrace{\boxed{\frac{1}{2}} * \boxed{\frac{1}{1-Y}}} \left(\overbrace{Y = \frac{X'}{2}}, \boxed{Y < \frac{1}{2}}\right)$$

$X = 1 - x'$

$x' = 1 - x$

$6/2$

$110/2$

2^{-1} $\quad 11 \to 3$

$$y = \frac{x'}{2} = \boxed{\frac{1-x}{2}}$$

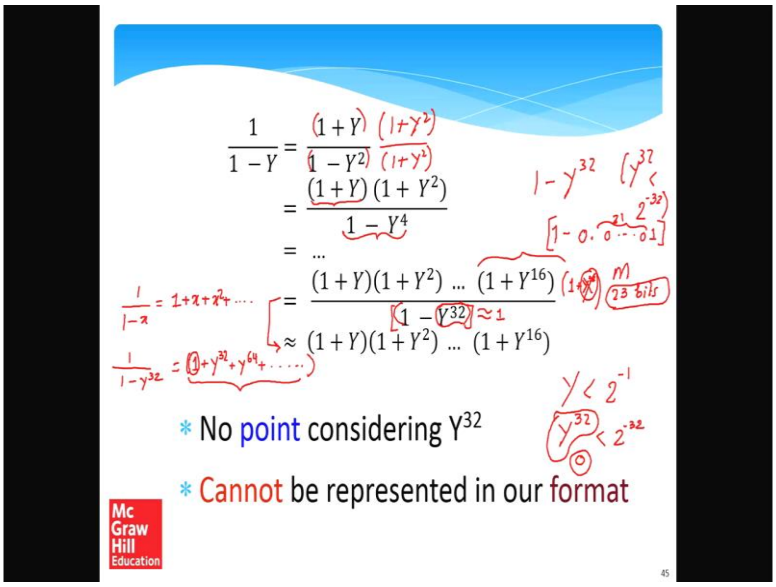
44

So, the algebra that we will do is like this, that the first thing that we will do is that we will slightly clean up one special case. So, we have said that 0 is less than equal to X is less than 1, but look at what happens when X is equal to 0. So, when X is equal to 0 then the significant is essentially 1. If the significant is 1 the reciprocal of the significant is also 1. So, it is a very trivial case and this can be handled separately. So, at least in this discussion let us assume that the significant is of the form P B equals 1 plus X and 0 is less than X is less than 1. So, X is strictly between 0 and 1 that will make it will make our life simple. Then let us go to the next step which is let us say that X is of the form 1 minus X dash or alternatively X dash is of the form 1 minus X. So, since X is strictly between 0 and 1 also X dash is strictly between 0 and 1. So, we can say that X dash is less than 1. Some more algebra 1 plus 1 is 2. So, this is 1 by 2 minus X dash, I take the half outside then I have 1 divided by 1 minus X dash by 2.

So, this is half times 1 by 1 minus Y whereas, say that Y is equal to X dash by 2. So, this term is being represented as Y. Since X dash is less than 1 Y is less than half, so since X dash is less than 1 and Y is equal to X dash by 2 Y becomes less than half. So, we finally, end up with this expression over here when 1 by the significant is half times a number of the form 1 by 1 minus Y. So, Y is equal to X dash by 2 which is equal to 1 minus X divided by 2 which can be computed very easily right, using a regular subtraction and a shift right and a right shift it can be computed very easily.

So, what we can do is that actually let me go back. So, since we are multiplying a number by half and everything is in base 2. So, multiplying a number by half pretty much means multiplying a number by 2 raised to the power of minus 1 or dividing by the number by 2 is that we will compute the result of 1 by 1 minus Y and right shift it by one position. So, that is the same as multiplying it by half. So, for example, you know 6 divided by 2, or 6 times half 6 is 1 1 0 this, if I if I am dividing by 2 at the same as right shifting this by one position. So, the result is 1 1 which is 3. So, this half is not a big deal even multiplying a binary number with half is same as right shifting that is also not a big deal. The difficulty lies in computing 1 by 1 minus Y if we can do that quickly we are done.

(Refer Slide Time: 25:52)



$$\frac{1}{1-Y} = \frac{(1+Y)}{(1-Y^2)}$$
$$= \frac{(1+Y)(1+Y^2)}{1-Y^4}$$
$$= \ldots$$
$$= \frac{(1+Y)(1+Y^2)\ \ldots\ (1+Y^{16})}{1-Y^{32}}$$
$$\approx (1+Y)(1+Y^2)\ \ldots\ (1+Y^{16})$$

* No point considering Y^{32}
* Cannot be represented in our format

45

$\frac{1}{1-x} = 1+x+x^2+\ldots$
$\frac{1}{1-y^{32}} = (1+y^{32}+y^{64}+\ldots)$
$1-y^{32}$
$[1-0.0\ldots01]$
2^{-21}
2^{-32}
≈ 1
m
23 bits
$Y < 2^{-1}$
$Y^{32} < 2^{-32}$
0

So, 1 by 1 minus Y is equal to let us see. So, let us multiply 1 plus Y to the top to the numerator and 1 plus Y to the denominator. So, we get 1 plus Y and recall that 1 minus Y times 1 plus Y is 1 minus Y square. So, similarly we can do the same once again. So, we can multiply 1 plus Y square to the top, and we multiply 1 plus Y square also to the bottom. So, we can basically do this. So, we multiply 1 plus Y square to the bottom as well. So, 1 minus Y square times 1 plus Y square is 1 minus Y to the power 4. So, we can continue in that way. So, then what we will see is that we will have the numerator will be 1 plus Y 1 plus Y square till 1 plus Y to the power of 16, and the denominator will be 1 minus Y to the power 32.

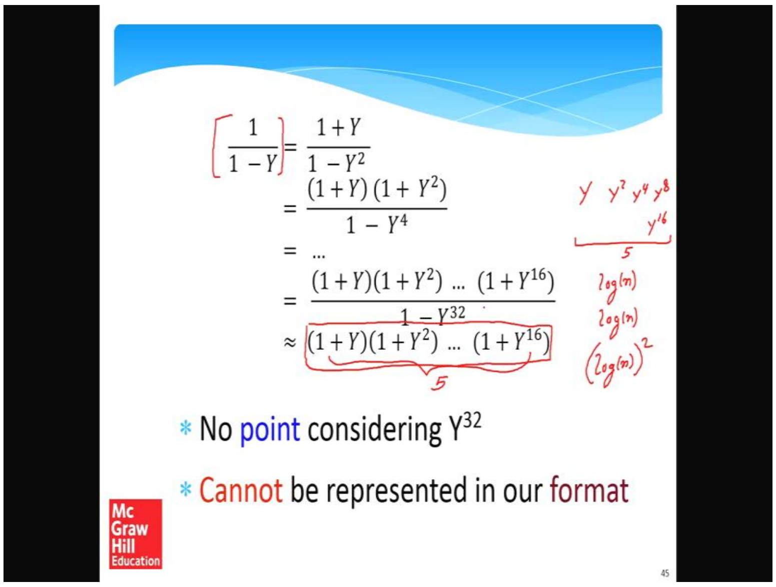
So, here is an interesting observation that we can make. The given the fact that we are working with a 32-bit system where the mantissa has only 23 bits right. In the given that you know that is a fact for us irrespective of the value of Y right. So what do we know about Y right it is instead of seeing irrespective of the value let us see what do we know about Y. Let us go to the previous slide. What we know is Y is less than half. Right this is something that we know that Y is less than 2 raised to the power minus 1. So, Y to the power 32 is less than 2 to the power minus 32 all right. So, if this number is less than 2 raise to the power minus 32. So, this is well outside 32 bits. So, this Y to the power 32 in our system is practically 0.

The reason being that it cannot be represented at all within our 23 mantissa bits right, I cannot represent the number 1 minus Y to the power 32. Where and the reason that is the case is that 2 to the power minus 32 basically means that is that we have 0 then we have. So, then we have 31 1s, sorry we have 31 0s. And then we have a 1. So, 1 minus this quantity simply cannot be represented using 23 bits. So, even more valid mantissa can have captured or represent this number. Because this number is simply you know Y to the power 32 is too small to be captured in our number system and 1 minus Y to the power 32 is too close to 1 and there is no way of representing this.

As a result, you know there is no point in multiplying this with even more terms and this term 1 minus Y to the power 32 is practically 1. Because irrespective of you know whatever it is value is, it cannot be represented. And also it makes no sense to multiply the numerator and denominator with other terms. For example, even if you multiply this with 1 plus Y to the power of 32, it will make no difference at all. The reason being that Y to the power 32 cannot be represented; it will not affect any of the other terms. Since it will not affect any of the other terms, this number is approximately equivalent to 1 plus Y 1 plus multiplied with 1 plus Y square till 1 plus Y to the power 16. So, we can actually we are actually discarding the denominator all right. And the reason we are discarding the denominator is because the denominator is too close to 1 and it is in fact, close that our number system will not be able to distinguish between just the numerator and the numerator divided by the denominator.

The reason it will not be able to distinguish is for another reason. So, let us consider you know one reason we have we have given let us also consider one more reason. So, what is you know 1 by any number of the form 1 by 1 minus X right. This is a number of the form so on and so forth right. Say any number of the form 1 by 1 minus is equal to see, just see if I expand the denominator in you know using this particular series expansion. So, the 1 will get multiplied with the numerator, but if any of the other terms get multiplied with the numerator there will be no way of representing the product, because all of those numbers will be less than 2 to the power minus 32 given the fact that Y is less than half and they cannot be represented. So, since they cannot be represented they will have to be ignored, and that is why we ignore the denominator and what we have essentially created maybe I can clean the slide and show.

(Refer Slide Time: 31:56)



$$\left[\frac{1}{1-Y}\right] = \frac{1+Y}{1-Y^2}$$
$$= \frac{(1+Y)(1+Y^2)}{1-Y^4}$$
$$= \ldots$$
$$= \frac{(1+Y)(1+Y^2)\ \ldots\ (1+Y^{16})}{1-Y^{32}}$$
$$\approx \boxed{\underbrace{(1+Y)(1+Y^2)\ \ldots\ (1+Y^{16})}_{5}}$$

$\underbrace{Y \quad Y^2 \quad Y^4 \quad Y^8 \quad Y^{16}}_{5}$

$\log(n)$

$\log(n)$

$(\log(n))^2$

* No point considering Y^{32}
* Cannot be represented in our format

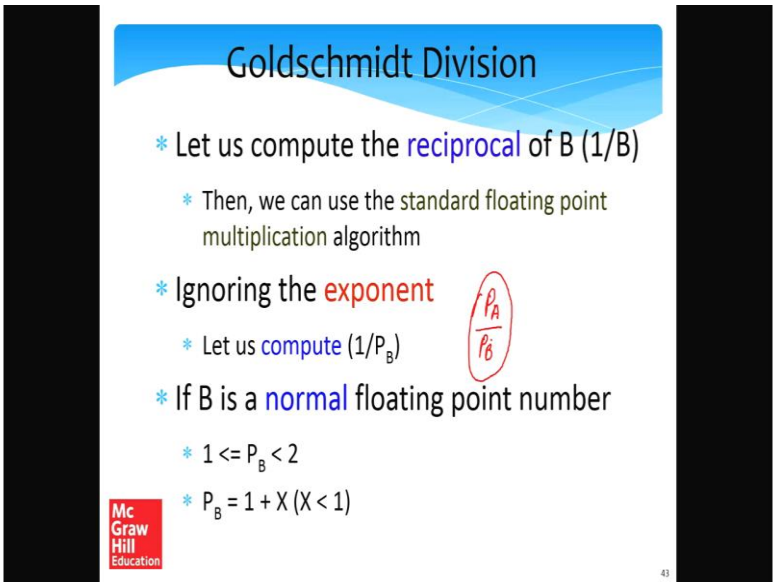
45

What we have essentially created is we have replaced a fraction with a numerator and a denominator with just a product of terms without a denominator at all. And the product of terms is 1 plus Y times 1 plus Y square till all the wave till 1 plus Y to the power 16 all right.

So, this is exactly what we have done it is a product. And this can be very quickly computed in our system. So, what we can do is, first we can compute Y, then we can compute Y times Y which is Y square, then we compute Y square times Y square which is Y to the power 4 which is the same to get Y to the power 8 and then we do the same to get Y to the power 16. So, this is 5 steps. So, if it is an n bit number system this will take log n steps for in every step there is a multiplication that again takes log n steps. So, total complexity is log n square. So, this is significantly faster than the n log n time algorithm that we have seen.

And the reason it is faster is because we pretty much use this expansion over here right you know this time if an expansion is used. All right using this particular form we have been able to quickly compute 1 by 1 minus Y right. So, 1 by 1 minus Y is pretty much a product of terms of the form 1 plus Y 1 plus Y square times 1 plus Y to the power 4 times 1 plus Y to the power 8 times 1 plus Y to the power 16 pretty much is the product of 5 terms. For each term we compute a power of Y. So, we compute the sum or the product of these 5 terms and this is our result. Once we have 1 by 1 minus Y we need to multiply it with half which is the same as right shifting it by one position.

(Refer Slide Time: 34:07)



Goldschmidt Division

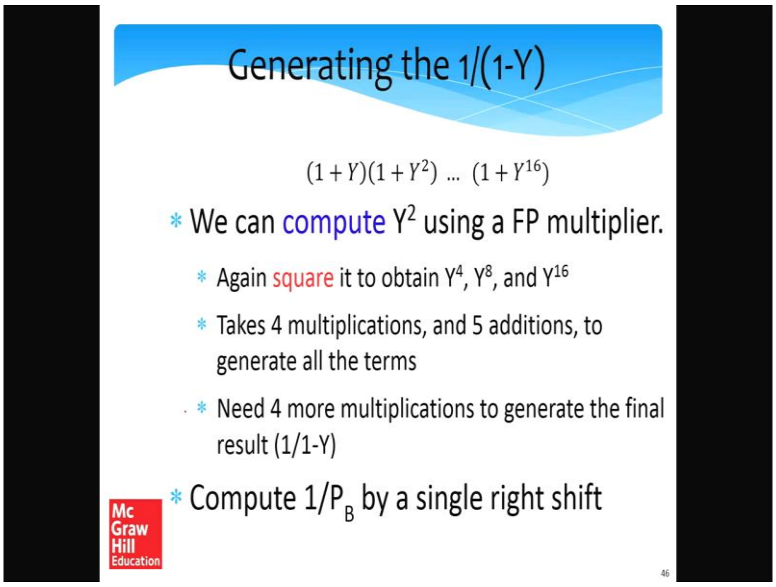
- Let us compute the reciprocal of B (1/B)
  - Then, we can use the standard floating point multiplication algorithm
- Ignoring the exponent
  - Let us compute $(1/P_B)$
- If B is a normal floating point number
  - $1 <= P_B < 2$
  - $P_B = 1 + X$ $(X < 1)$

$\frac{P_A}{P_B}$

43

So, this will give us the reciprocal of the significant, then we again go back and give in a reciprocal of the significant we will multiply it you know with the significant of A. So, we basically get P A divided by P B. And then the rest of the division algorithm will be the same as we are essentially getting this point I will bit with a much faster method the rest of the steps are exactly the same.

(Refer Slide Time: 34:22)



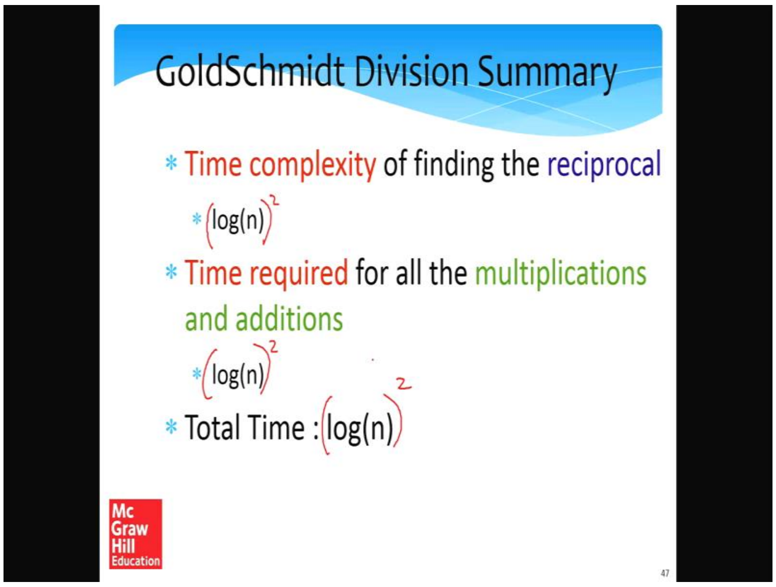
Generating the 1/(1-Y)

$$(1+Y)(1+Y^2)\ \dots\ (1+Y^{16})$$

- We can compute Y^2 using a FP multiplier.
  - Again square it to obtain Y^4, Y^8, and Y^{16}
  - Takes 4 multiplications, and 5 additions, to generate all the terms
  - Need 4 more multiplications to generate the final result (1/1-Y)
- Compute $1/P_B$ by a single right shift

46

(Refer Slide Time: 34:33)

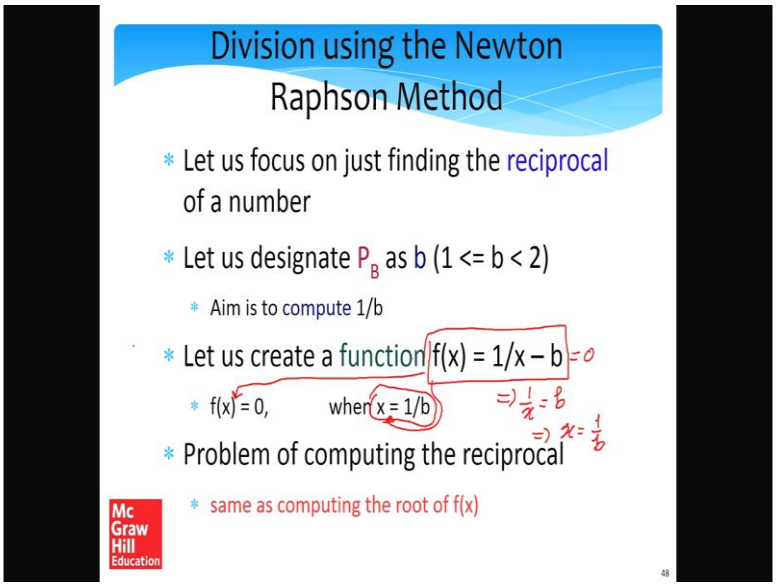


GoldSchmidt Division Summary

- Time complexity of finding the reciprocal
  - $(\log(n))^2$
- Time required for all the multiplications and additions
  - $(\log(n))^2$
- Total Time : $(\log(n))^2$

So, the complexity of this I am sorry there is a small error in this slide. So, time complexity of finding the reciprocal is log of n square right. As we had computed on the last slide there is log of n square. And the time required thus for all the multiplications in log of n square and the total time in log of n square. Sorry this is one path that I will fix in the subsequent versions of the slide, but this is what it should be all right.

(Refer Slide Time: 35:05)



Division using the Newton Raphson Method

- Let us focus on just finding the reciprocal of a number
- Let us designate P_B as b (1 <= b < 2)
  - Aim is to compute 1/b
- Let us create a function $\boxed{f(x) = 1/x - b} = 0$
  - f(x) = 0, when $\boxed{x = 1/b}$

$\Rightarrow \frac{1}{x} = b$

$\Rightarrow x = \frac{1}{b}$

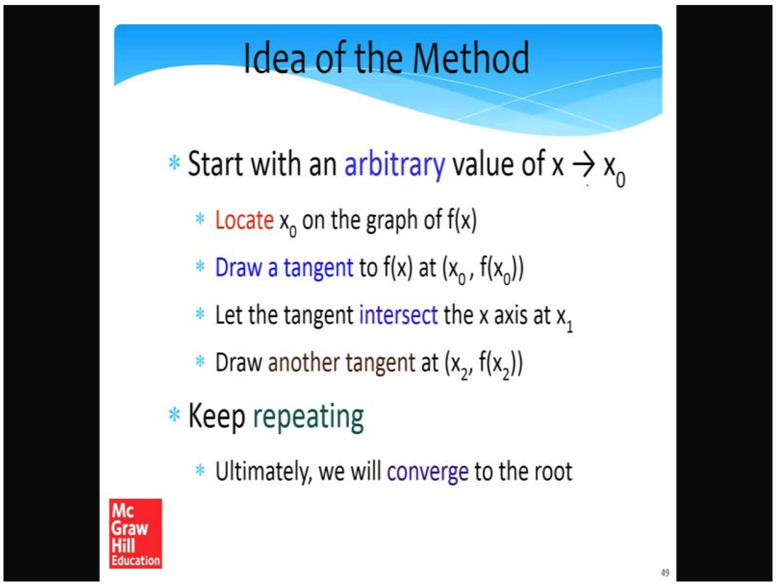
- Problem of computing the reciprocal
  - same as computing the root of f(x)

So, let us now use a slightly different kind of method to find the reciprocal of P B using the using a different method called the newton Raphson method. So, newton Raphson and goldsmith are similar methods, but let us at least for the sake of it learn a new method.

So, given any number let us try to find the reciprocal of a number. Let us designate P B as a generic number b where b is of the same range 1 is less than equal to b which is less than 2. So, the aim is to compute 1 by b. So, let us do one thing. Let us create a function f x where f x is equal to 1 by x minus b right. So, let us you know arbitrarily create a function f x of the form 1 by x minus b right. So, f x is equal to 0. So, an x is equal to 1 by b. So, what we can say that when f x is equal to 0 right. So, that is when 1 by x is equal to b or rather b or rather sorry x is equal to 1 by b. So, we have created a function of this type. So, the problem of computing the reciprocal right, the reciprocal of b is the same as computing the route of f x. So, the route of f x is where f x becomes 0 and where does f x becomes 0 well f x becomes 0 when x is equal to 1 by b.

See if I can take this equation and find the point where it becomes 0 then essentially I will find that point is where x equal to 1 by b and that is the reciprocal which we want to compute right. So, I am essentially what I am doing is I am converting the problem of computing a reciprocal 1 by some number as computing the route of an equation. So, we will use a traditional equation solving technique to actually compute the reciprocal of a number and this technique is the newton Raphson method.

(Refer Slide Time: 37:21)



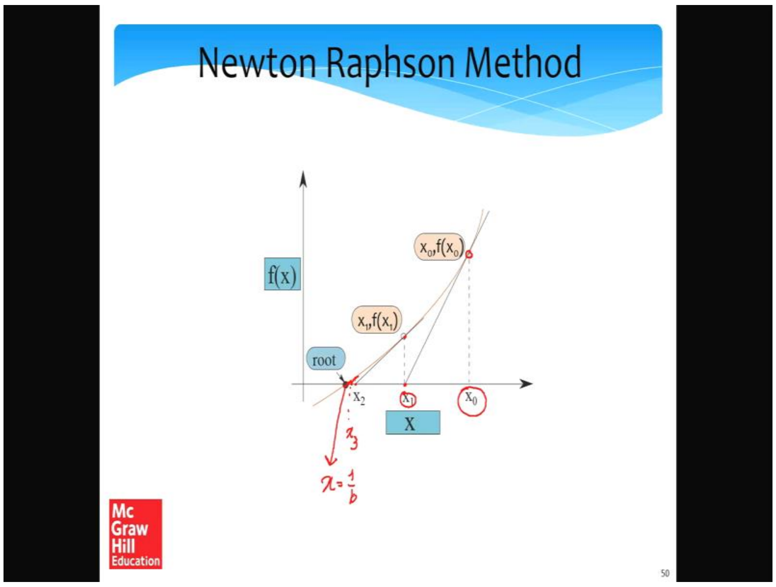
Idea of the Method

- Start with an arbitrary value of x $\rightarrow x_0$
  - Locate x_0 on the graph of f(x)
  - Draw a tangent to f(x) at $(x_0, f(x_0))$
  - Let the tangent intersect the x axis at x_1
  - Draw another tangent at $(x_2, f(x_2))$
- Keep repeating
  - Ultimately, we will converge to the root

49

So, let us do one thing. Let us start with an arbitrary value of x which is x 0 right. So, let us guess a value arbitrary value and start from there. Let us locate x 0 on the graph of f or the graph of f x. Let us draw a tangent f x at this point and let us keep repeating.

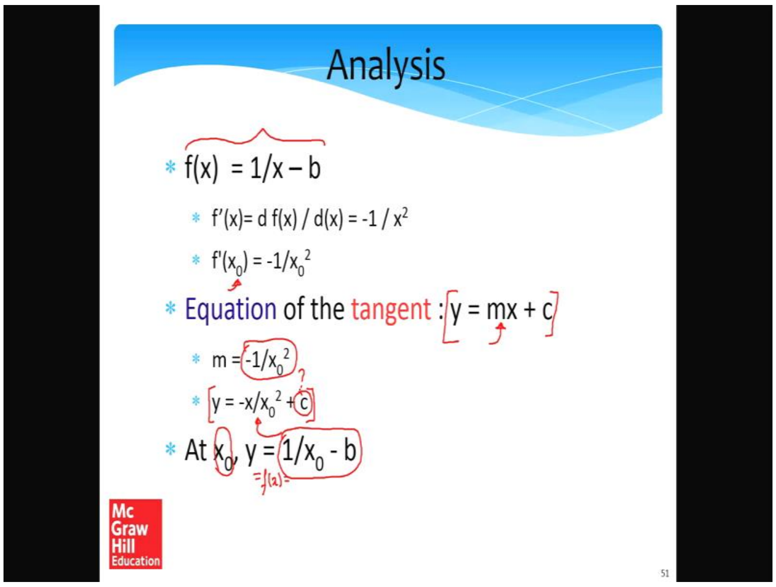
(Refer Slide Time: 37:42)



So, a picture is worth a thousand words. So, I will tell you what we do let us arbitrarily guess a value of x which is x 0 right. And then we locate the point x 0 on the curve of f x. So, this is the curve of f x and let us assume that you know for some f x this is the curve. So, what we will do is at that this point let us compute a tangent to this curve. So, you may compute a tangent to this curve at some point this will intersect the x axis. So, let this point be x 1. Again let us locate the point on the curve y equals f x where is x coordinate is x 1 which is this point let us again compute a tangent to the curve.

So, let it intersect the x axis at point x 2 let us again locate the point on f x whose x coordinate is x 2 again we compute the tangent to the curve and let this point be x 3. So, in this manner we just can keep going till we reach the final route. And the route is where we expect x to be equal to 1 by p the standard equation solution method is called the newton Raphson method. And so here it is a mathematical method may be take some point on the curve we keep computing tangents, then you know till we hit the route. So, the way that we would actually do it in a in a hardware is that we will implement this math in the hardware we will start from a random point x 0, find the point on the curve compute the tangents see where it intersects the x axis again find the curve and keep repeating the process.

(Refer Slide Time: 39:29)



Analysis

- $\overbrace{f(x)} = 1/x - b$
  - $f'(x) = d\,f(x) / d(x) = -1 / x^2$
  - $f'(x_0) = -1/x_0^2$
- Equation of the tangent : $[y = mx + c]$
  - $m = \boxed{-1/x_0^2}$
  - $[y = -x/x_0^2 + \boxed{c}]$
- At $\boxed{x_0}$, $y = \underbrace{\boxed{1/x_0 - b}}_{=f(x)=}$

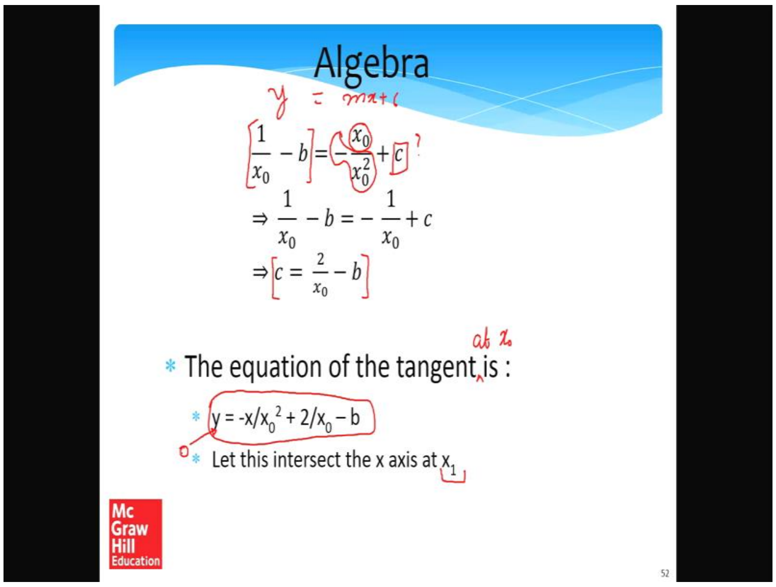
Mc Graw Hill Education

51

So, recall that my equation was f x is equal to 1 by x minus b. So, computing the tangent means finding the derivative at x, right at some point x wherever you want to compute the tangent at. So, f dash of x is d f x by d x which in this case is minus 1 by x square, where x 0 the derivative of the curve is minus 1 divided by x 0 square, the equation of the tangent. So, let us find the equation of tangent. So, let us say the equation of general equation of a straight line y equals m x plus c.

So, in this case we know m which is the slope a slope at x 0 is minus 1 by x 0 square. So, essentially the equation of the tangent is y equals minus x by x 0 square plus a constant c. So, we need to now find the constant c. So, what do we know we know that at x 0 y is equal to right the value of y which is equal to f x is equal to 1 by x 0 minus b. So, this is something that we know because at x 0 f x is 1 by x 0 minus b. So, this is something that we know this can be substituted in this equation to find the value of c.

(Refer Slide Time: 40:55)



Algebra

$y = mx + c$

$$\left[\frac{1}{x_0} - b\right] = -\frac{x_0}{x_0^2} + \boxed{c}\ ?$$

$$\Rightarrow \frac{1}{x_0} - b = -\frac{1}{x_0} + c$$

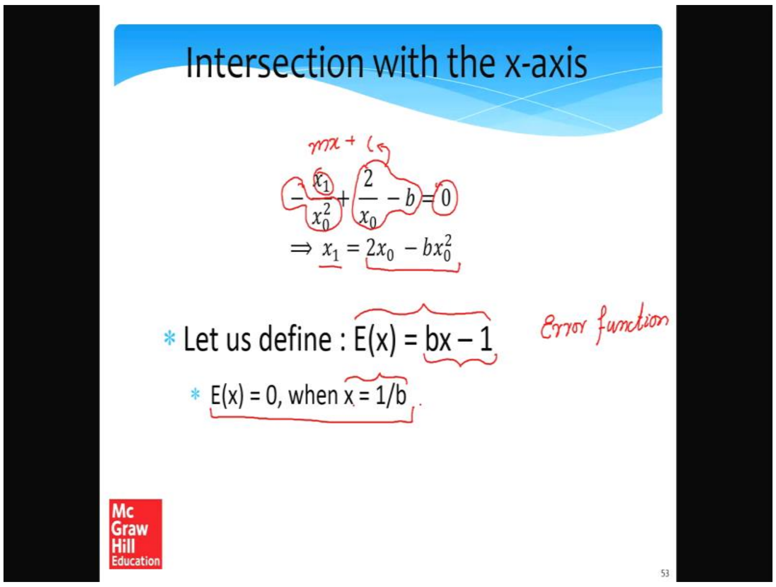
$$\Rightarrow \left[c = \frac{2}{x_0} - b\right]$$

- The equation of the tangent at x_0 is :
  - $y = -x/x_0^2 + 2/x_0 - b$
  - Let this intersect the x axis at x_1

So, once we do that what we see is y equals right m x plus c. So, at x 0 this is the value of y 1 by x 0 minus b m x of this part was m minus 1 by x 0 square and at x 0 the value of x is x 0 of course, plus c which is the constant whose value we want to find out. So, what we will do is we will do a little bit more of algebra and it will come that you know the last stage will tell us that c is equal to 2 by x 0 minus b. So, the equation of the tangent becomes y equals minus x by x 0 square. So, the equation of the tangent of course, you can say at x 0 right. So, this is something that can be said that the equation of the tangent at x 0 is y equals minus x divided by x 0 square plus 2 by x 0 minus b right the m x plus c we know m and c. So, this is the equation.

So, let this curve that is this tangent this line intersects the x axis at point x 1. If that is the case what we can do is. So, we need to find this point x 1. So, what we know is that when we intersect the x axis y is equal to 0.

(Refer Slide Time: 42:18)



Intersection with the x-axis

$$\overset{mx + c}{-\frac{x_1}{x_0^2} + \frac{2}{x_0} - b = 0}$$

$$\Rightarrow x_1 = 2x_0 - bx_0^2$$

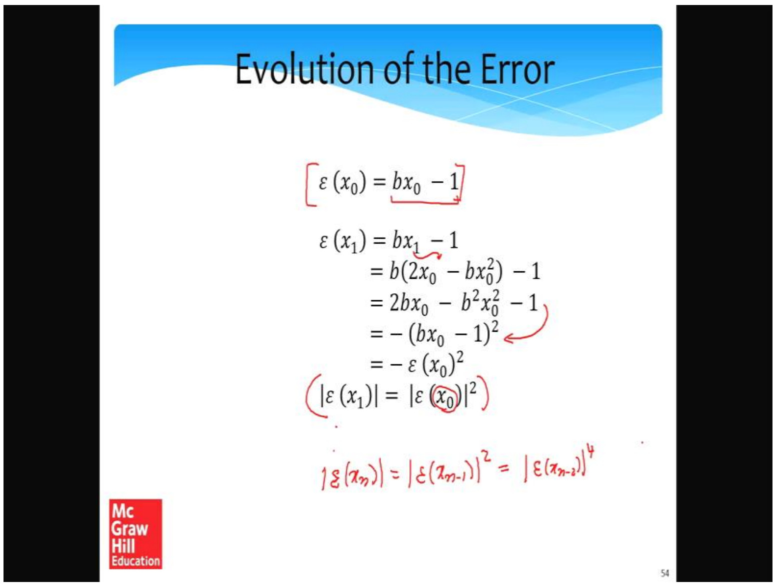
- Let us define : $E(x) = bx - 1$
  - $E(x) = 0$, when $x = 1/b$

Error function
53

So, we can thus say that you know 0 and this is this needs to be m x plus c. So, since we have computed this as a value of c and this is the value of m value of x value where this intersects the x axis is x 1. So, this is the equation that we can create and if we solve this equation we will find that x 1 is equal to 2 x 0 minus b x 0 whole square. So, let us now define one more function called an error function. So, recall that we have defined 2 functions arbitrarily we have guessed and we hope that you know it will help us one such function that we have defined is f x which is 1 by x minus b. So, the insight here was at the route of the function. So, when f x is 0 we will have x as 1 by B which is what we want to compute. Now we are guessing one more function called the error function. So, the error function is defined as the like this you know. So, these functions defining them is actually an art. For different function you need to define them differently. So, in this case this is how we are defining them.

So, the error function E x is b x minus 1 that is the way we are defining it. So, what is the advantage of defining it this way? That when we reach the route x where x is equal to 1 by b. The error becomes equal to 0 you just put x here say if x is 1 by b, b turns 1 by b is one and 1 minus 1 is 0. So, the error is 0. So, our aim is also to make the error as 0 an error happens to be 0, when we reach the route of the function f x.

(Refer Slide Time: 44:10)



Evolution of the Error

$$\varepsilon(x_0) = \underline{bx_0 - 1}$$

$$\begin{aligned}\varepsilon(x_1) &= bx_1 - 1\\ &= b(2x_0 - bx_0^2) - 1\\ &= 2bx_0 - b^2x_0^2 - 1\\ &= -(bx_0 - 1)^2\\ &= -\varepsilon(x_0)^2\end{aligned}$$

$$|\varepsilon(x_1)| = |\varepsilon(x_0)|^2$$

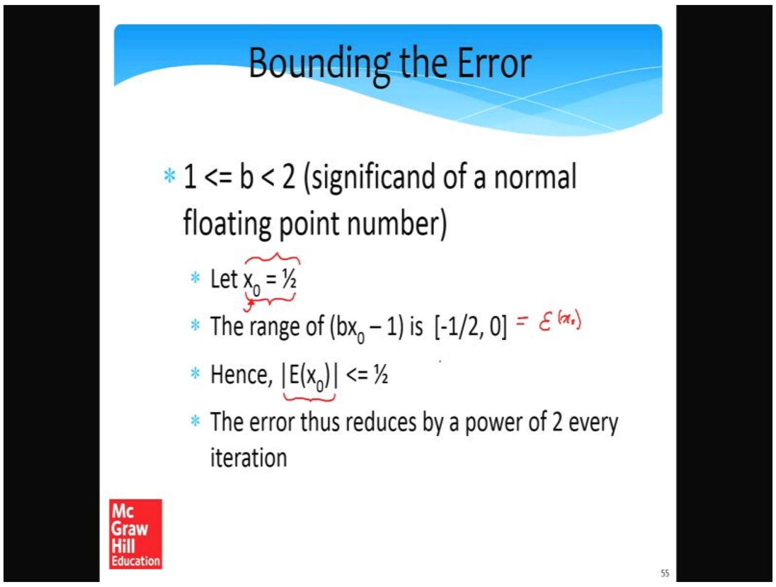
$$|\varepsilon(x_n)| = |\varepsilon(x_{n-1})|^2 = |\varepsilon(x_{n-2})|^4$$

Mc Graw Hill Education
54

So, what is the initial error? So, the initial error at point x at point sorry x 0 is b x 0 minus 1 that is the initial error. What is the error after we reach point x 1 is b x 1 minus 1 since this is what x 1 was computed to be if we go back we will say x 1 was computed to be this expression? So, b x 1 minus 1 is b times to x 0 minus b x 0 whole square minus 1 we do a little bit of algebraic manipulations till we come to this result. So, b x 0, the error is minus of b x 0 minus 1 square and b x 0 minus 1 is the error of x 0. So, this will become minus of error of x 0 square.

So, we can alternatively see that the error of at point x 1 it is absolute value is equal to the absolute value of the error is equal to the square of the absolute value of the error at x 0. Similarly, you know we can extend this argument we can say that the error sorry we can say that the error at any point x n is equal to the error at x of n minus 1 square, which is equal to a error of at x of n minus t to the power 4 and so on.

(Refer Slide Time: 45:52)



Bounding the Error

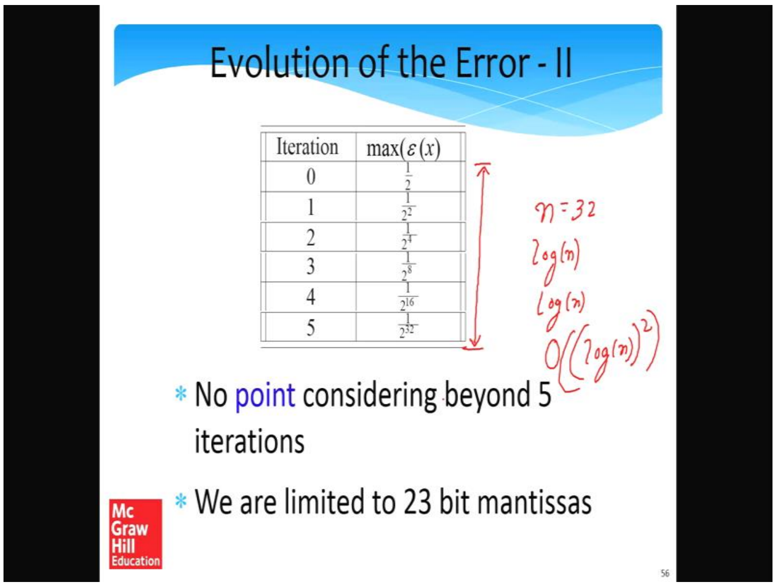
- 1 <= b < 2 (significand of a normal floating point number)
  - Let $x_0 = ½$
  - The range of $(bx_0 – 1)$ is [-1/2, 0] $= \mathcal{E}(x_0)$
  - Hence, $|E(x_0)| <= ½$
  - The error thus reduces by a power of 2 every iteration

So, we will use this result. The way we will use this result is like this. That let us now try to put a bound on the error. So, we know that since b is actually representing the significant of one of our floating point occurrence. It is between 1 and 2. So, one is less than equal to b is less than 2 is the significant of a normal floating point number. So, let us first guess x 0 right our first guess x 0, actually be half right. So, the range of b x 0 minus 1 b times x 0 minus 1 what will it be. So, what is the range of b, b is between 1 and 2. So, the range will actually b between minus half and 0 right.

So, that will be the range of b x 0 minus 1, which is the error at point x 0. So, hence E x 0 the error at point x 0 is always less than equal to half. Since the error at any point is less than equal to half and we reduce the error by a power of 2, you know in every iteration ultimately the error will approach 0. Because if we come back to this equation then initially if the error you know is half then at this point the error will be one-forth and you know this will become one-eighth and it will sort of keep going down and down and down till it ultimately reaches 0.

So, because you know in any number that we if you multiply by itself the number is less than 1, it will ultimately reach 0. So, the error will ultimately converge at 0. So, this at least tells us there is an initial estimate if we set x 0 as half it is a good estimate because we can not only bound the error, but we can in repeated steps the error will ultimately converge to 0.

(Refer Slide Time: 47:49)



Evolution of the Error - II

Iteration	$\max(\varepsilon(x))$
0	$\frac{1}{2}$
1	$\frac{1}{2^2}$
2	$\frac{1}{2^4}$
3	$\frac{1}{2^8}$
4	$\frac{1}{2^{16}}$
5	$\frac{1}{2^{32}}$

$n = 32$

$\log(n)$

$\log(n)$

$O((\log(n))^2)$

* No point considering beyond 5 iterations
* We are limited to 23 bit mantissas

56

So, let us look at the maximum error that is possible in iteration 0 in iteration 0 it is half in iteration 1 it is 1 by 2 square 2 it is 1 by 2 to the power 4 3 it is 1 by 2 to the power 8, 4 it is 1 by 2 raised to the power 16 and ultimately 5 is 1 by 2 raised to the power 32. So, one thing that needs to be kept in mind that the error after 5 iterations is becomes 1 by 2 to the power 32 which is a fairly small number, but there is an there is a very important point to make here that the error function that we actually consider which is let me go back the error function that we actually consider here, I had at that point said it is defined arbitrarily.

Now let me take some liberty to say that it was not that arbitrary it has a certain physical connotation and this physical connotation we will require right now. So, why do not you remember this expression and carry on with me till this slide. So, we had said that look the error function E x is defined as b x minus 1, which is essentially b times x minus 1 by b. So, x minus 1 by B is pretty much the distance between our current estimate of what the route is and the theoretical route right. So, this is our current estimate that is x minus 1 by b.

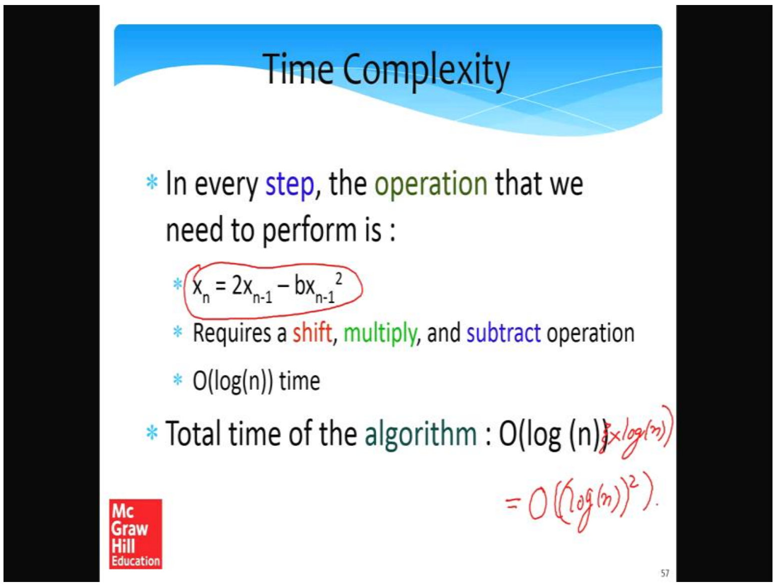
So, that is essentially the difference between what we want to compute 1 by b and we currently have. So, basically you can say that this is b times delta. So, delta is the difference. So, if b times delta is let us say equal to 2 to the power minus 32. And b itself is a number. So, what was the bound 1 b again the bound 1 b was between 1 and 2. So, b itself if it if it is between 1 and 2. It means that delta is definitely you know the bound on delta. So, let us say if this is 1 this remains if this is 2 then delta will be a number between 2 to the power minus 32 and 2 to the power minus33 this is anyway much smaller than what our mantissas can represent because our mantissas cannot go beyond 23 bits. So, 5 iterations is more than enough to drive the delta. So, delta here is refined as the difference between what we have and what is the ideal value right. So, this delta over here 5 iterations is sufficient to drive the delta.

So, low that it is beyond our limit of precision. So, thus we should not be concerned which further means that if we do the newton Raphson iteration for 5 iterations. Whatever value of you know x that we have seen initially we have, x 0 then we will have x 1 and finally, we will have x 5 the value of x 5 is very close to the value 1 by v which is what we wanted to compute. So, there is still a small difference and this difference is delta, but the difference is smaller than our smallest precision. So, and since this floating point representation is approximate we can use this fact to in the sense ignore that minimal amount of error because there is nothing that we can do about it. And go forward and why are we ignoring because our mantissas are fixed size which is 23 bits right, so that is the reason there is no choice.

Now given the fact that we have put a limit on a bounce on the error, what we will what we do see from this slide, is that 5 iterations were suffice and why 5 because you know we reached 2 to the power 32 which is beyond our you know beyond our limit of 23 bits. So, how many iterations do we take well if we have a 32-bit number system in that sense n is equal to 32, and we proceed in the powers of 2. So, we proceed in log steps. So, we require log n iterations and in each iteration what do we do we compute a new value of x. So, what is the new value of x the way we compute it is x 1 is equal to 2 x 0 minus b x 0 square or alternatively any x n is 2 times x n minus 1 minus b x n minus 1 square. So, there are only multiplications and subtractions here. So, each step also takes log n time. So, the total time taken by all the steps is log n times log n which is log n whole square order of log n whole square.

So, this is the takes the exactly the same amount of time as the goldsmith algorithm. It is just the different way of doing things it has certain graphical connotation and one more interesting thing that this can do is that of course, we used it to compute the reciprocal we can use, but we can use it to compute other mathematical functions also, but this I will mention after you know a few slides.

(Refer Slide Time: 53:19)



Time Complexity

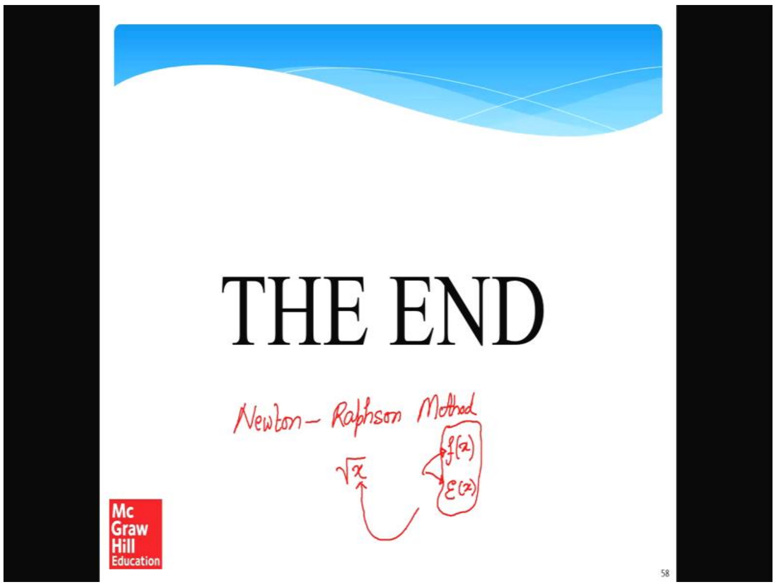
- In every step, the operation that we need to perform is :
  - $x_n = 2x_{n-1} - bx_{n-1}^2$
  - Requires a shift, multiply, and subtract operation
  - O(log(n)) time
- Total time of the algorithm : $O(\log (n) \times \log(n))$

$= O((\log(n))^2)$.

57

So, the time complexity is like this again a small error in this slide I will fix it. In every step of the operation we need to compute this thing. Now this requires some shifts in multiplies and subtracts sp this requires order of log n time. So, the total time of the algorithm sorry a small mistake here is order of log n for each step multiplied with log n steps right which is order of log of n square, log n whole square all right. So, this is the time seen same as the goldsmith algorithm in that sense.

(Refer Slide Time: 54:01)



THE END

Newton – Raphson Method

$\sqrt{x}$

$f(x)$

$\mathcal{E}(x)$

So, we reached the end of the chapter one computer arithmetic the fairy large chapter. So, I leave you with a problem to do. Can you use the newton Raphson method to design a piece of hardware a hardware circuit to compute square root of x given a number, we want to given a positive number we want to compute it is square root? So, for this you need to find 2 estimates. One estimate for f x such that it is root is equal to square root of x, and another error function such that you know it has some physical connotation with actual error and also the error function reduces significantly in each iteration, such that the number of iterations are bounded and ultimately the error approaches 0. So, you need to guess these 2 functions to a little bit of math the rest remains the same and at the end you need to give us the fast hardware algorithm to computes square root of x.

So, this ends chapter 7. I will see you in chapter 8, where we will discuss processor design.