What is claimed is:

1. A method comprising:

determining, before run time, whether a code for a programmable culling unit has bounds that can be determined; and

if so, for each variable, tracking the bounds at compile time through the code.

2. The method of claim 1 including using information about interval instructions to generate scalar instructions.

3. The method of claim 2 wherein using data includes using tracked interval bounds.

4. The method of claim 1 wherein if the bounds cannot be determined at compile time, making an assumption at compile time of the interval bounds.

5. The method of claim 1 including simplifying one or more bounds to single sided bounds.

6. The method of claim 4 including reassessing the assumption at run time.

7. The method of claim 6 wherein if the assumption is determined to be valid, running the code.

8. The method of claim 6 including if the assumption is determined not to be valid, making a new assumption and compiling the code with the new assumption.

9. The method of claim 8 wherein if the new assumption does not work, compiling the code without using interval bounds.

10. The method of claim 1 wherein tracking the bounds including evaluating instructions using bounded arithmetic.

11. The method of claim 10 wherein evaluating instructions using bounded arithmetic includes using one of interval arithmetic, affine arithmetic, or Taylor arithmetic.

12. A computer readable medium storing instructions executed by a computer to:

determine during compilation whether code for a programmable culling unit has bounds that can be calculated; and

if so, for each variable, tracking the bounds through the code during compilation.

13. The medium of claim 12 further storing instructions to use information

about interval instructions to generate scalar instructions.

14. The medium of claim 13 further storing instructions to use tracked interval bounds as said data.

15. The medium of claim 12 further storing instructions to make an assumption at compile time of the interval bounds if the interval bounds cannot be calculated at compile time.

16. The medium of claim 12 further storing instructions to simplify one or more double sided interval bounds to single sided interval bounds.

17. The medium of claim 15 further storing instructions to reassess the assumption at run time.

18. The medium of claim 17 further storing instructions to run the code if the assumption is determined to be valid.

19. The medium of claim 17 further storing instructions to make a new assumption if the assumption is determined not to be valid and compile the code with the new assumption.

20. The medium of claim 19 further storing instructions to compile the code without using interval bounds if the new assumption is not valid.

21. An apparatus comprising:

a programmable culling unit; and

a compiler coupled to said programmable culling unit, said compiler to determine whether code for the programmable culling unit has bounds that can be calculated and, if so, for each variable, tracking the bounds through the code during compilation.

22. The apparatus of claim 21 wherein said apparatus is a display adapter.

23. The apparatus of claim 21, said compiler to make an assumption at compile time of the interval bounds if the interval bounds cannot be calculated.

24. The apparatus of claim 23 including a controller.

25. The apparatus of claim 24 wherein said controller to reassess the assumption at run time.

26. The apparatus of claim 25, said controller to run the code if the assumption is determined to be valid.

27. The apparatus of claim 25, said compiler to make a new assumption if the assumption is determined not to be valid and to compile a code with the new assumption.

28. The apparatus of claim 27, said compiler to compile the code without using bounds if the new assumption is not valid.

29. The apparatus of claim 21, said compiler to use information about interval instructions to generate scalar instructions.

30. The apparatus of claim 29, said compiler to use interval bounds to generate scalar instructions.

## CLAIMS

1. A method comprising:

determining, before run time, whether a code for a programmable culling unit has bounds that can be determined; and

if so, for each variable, tracking the bounds at compile time through the code.

2. The method of claim 1 including using information about interval instructions to generate scalar instructions.

3. The method of claim 2 wherein using information includes using tracked interval bounds.

4. The method of claim 1 wherein if the bounds cannot be determined at compile time, making an assumption at compile time of the interval bounds.

5. The method of claim 1 including simplifying one or more bounds to single, sided bounds.

6. The method of claim 4 including reassessing the assumption at run time.

7. The method of claim 6 wherein if the assumption is determined to be valid, running the code.

8. The method of claim 6 including if the assumption is determined not to be valid, making a new assumption and compiling the code with the new assumption.

9. The method of claim 8 wherein if the new assumption does not work, compiling the code without using interval bounds.

10. The method of claim 1 wherein tracking the bounds including evaluating instructions using bounded arithmetic.

11. The method of claim 10 wherein evaluating instructions using bounded arithmetic includes using one of interval arithmetic, affine arithmetic, or Taylor arithmetic.

12. A computer readable medium storing instructions executed by a computer to:

determine during compilation whether code for a programmable culling unit has bounds that can be calculated; and

if so, for each variable, tracking the bounds through the code during compilation.

13. The medium of claim 12 further storing instructions to use information about interval instructions to generate scalar instructions.

14. The medium of claim 13 further storing instructions to use tracked interval bounds as said information.

15. The medium of claim 12 further storing instructions to make an assumption at compile time of the interval bounds if the interval bounds cannot be calculated at compile time.

16. The medium of claim 12 further storing instructions to simplify one or more double sided interval bounds to single sided interval bounds.

17. The medium of claim 15 further storing instructions to reassess the assumption at run time.

18. The medium of claim 17 further storing instructions to run the code if the assumption is determined to be valid.

19. The medium of claim 17 further storing instructions to make a new assumption if the assumption is determined not to be valid and compile the code with the new assumption.

20. The medium of claim 19 further storing instructions to compile the code without using interval bounds if the new assumption is not valid.

21. An apparatus comprising:

a programmable culling unit; and

a compiler coupled to said programmable culling unit, said compiler to determine whether code for the programmable culling unit has bounds that can be calculated and,

if so, for each variable, tracking the bounds through the code during compilation.

22. The apparatus of claim 21 wherein said apparatus is a display adapter.

23. The apparatus of claim 21, said compiler to make an assumption at compile time of the interval bounds if the interval bounds cannot be calculated.

24. The apparatus of claim 23 including a controller.

25. The apparatus of claim 24 wherein said controller to reassess the assumption at run time.

26. The apparatus of claim 25, said controller to run the code if the assumption is determined to be valid.

27. The apparatus of claim 25, said compiler to make a new assumption if the assumption is determined not to be valid and to compile a code with the new assumption.

28. The apparatus of claim 27, said compiler to compile the code without using bounds if the new assumption is not valid.

29. The apparatus of claim 21, said compiler to use information about interval instructions to generate scalar instructions.

30. The apparatus of claim 29, said compiler to use interval bounds to generate scalar instructions.