Proposte di progettini d’esame

Prof. Andrea Marongiu, Dott. Paolo Burgio
({andrea.marongiu, paolo.burgio}@unimore.it)

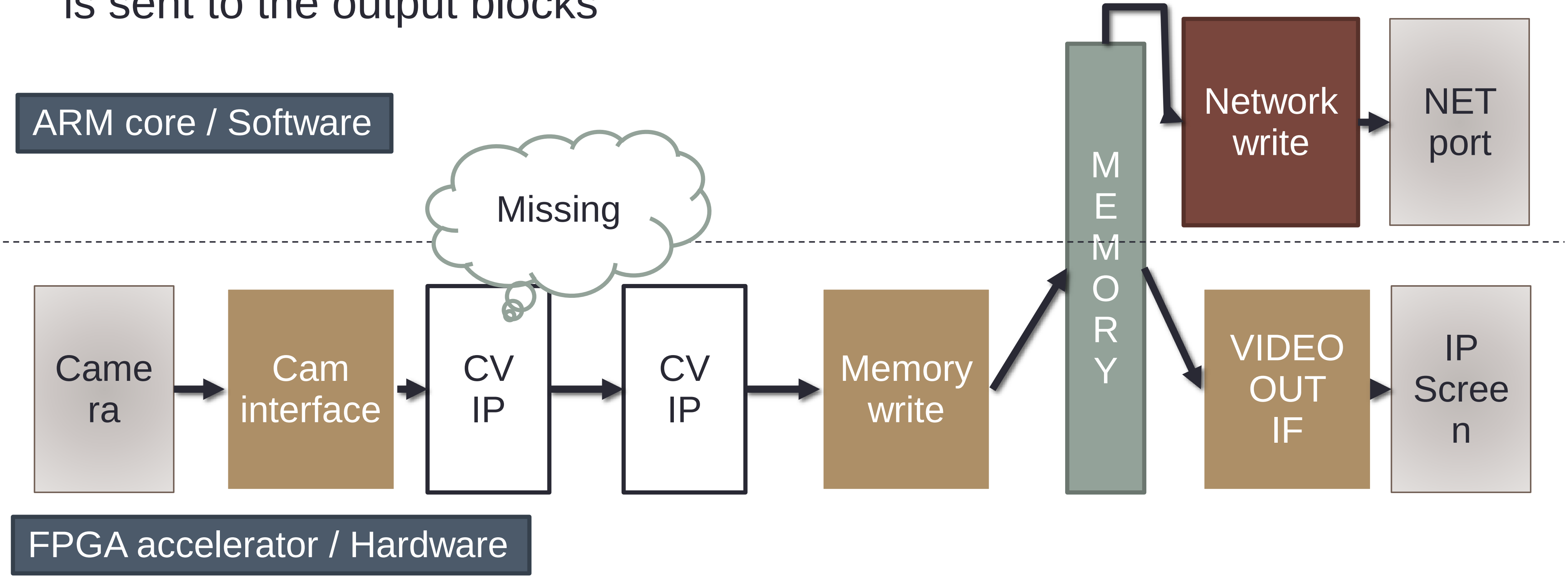
FPGA-accelerated CV pipeline

Step 1

- Porting on Xilinx Ultrascale+ platform of an existing camera/video interface implemented in programmable logic

Step 2

- Implementation of a simple CV block to process sampled data before it is sent to the output blocks



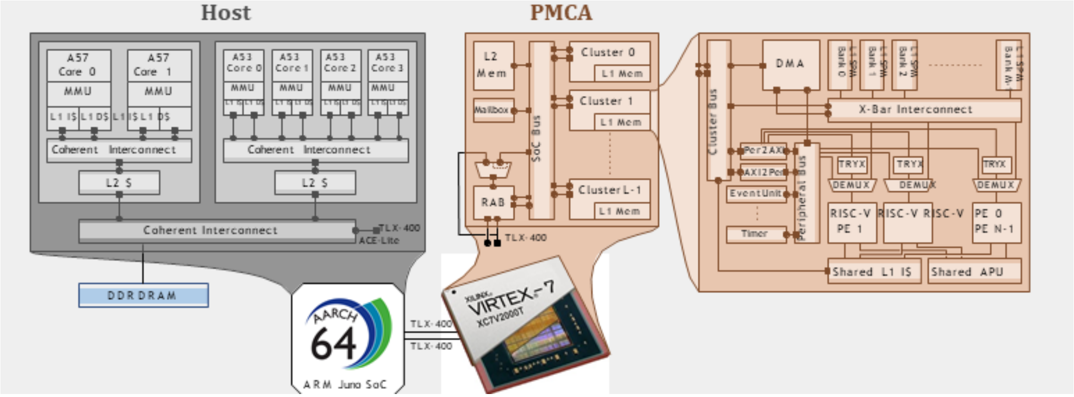
Visual odometry on FPGA platform autonomous robots

- Implementation of graphical kernel IP on FPGA
- Originally, it was designed for GPGPU
- Used for depth-map building, visual odometry….

Visual odometry is the process of determining the position and orientation of a robot by analyzing the associated camera images.

OpenMP benchmark suite per HERO

- Porting of (a subset of) the **Rodinia** benchmarks on the HERO platform



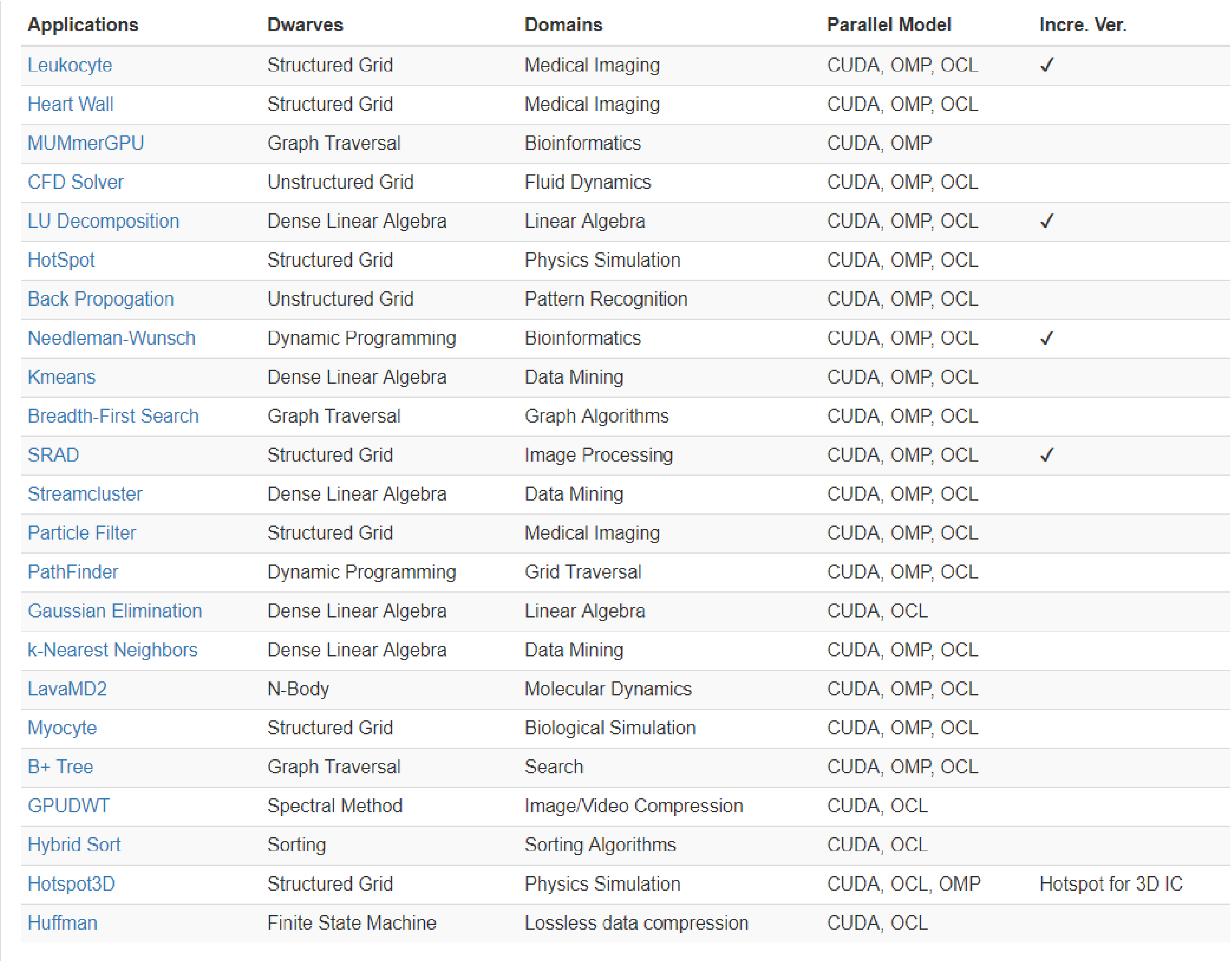
- OpenMP programming for heterogeneous systems

https://rodinia.cs.virginia.edu/doku.php

- Deal with (optimize for) specific HW features of the HERO platform
  - At the application level
  - At the system level

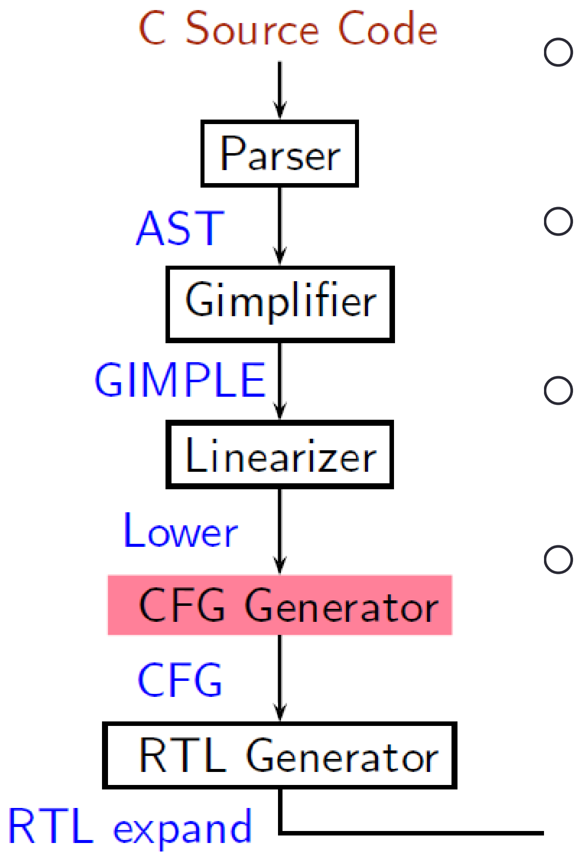
- Evaluate performance
  - Comparative to sequential, CUDA on NVIDIA Tegra

OpenMP benchmark suite per HERO



Applications	Dwarves	Domains	Parallel Model	Incre. Ver.
Leukocyte	Structured Grid	Medical Imaging	CUDA, OMP, OCL	✓
Heart Wall	Structured Grid	Medical Imaging	CUDA, OMP, OCL	
MUMmerGPU	Graph Traversal	Bioinformatics	CUDA, OMP	
CFD Solver	Unstructured Grid	Fluid Dynamics	CUDA, OMP, OCL	
LU Decomposition	Dense Linear Algebra	Linear Algebra	CUDA, OMP, OCL	✓
HotSpot	Structured Grid	Physics Simulation	CUDA, OMP, OCL	
Back Propogation	Unstructured Grid	Pattern Recognition	CUDA, OMP, OCL	
Needleman-Wunsch	Dynamic Programming	Bioinformatics	CUDA, OMP, OCL	✓
Kmeans	Dense Linear Algebra	Data Mining	CUDA, OMP, OCL	
Breadth-First Search	Graph Traversal	Graph Algorithms	CUDA, OMP, OCL	
SRAD	Structured Grid	Image Processing	CUDA, OMP, OCL	✓
Streamcluster	Dense Linear Algebra	Data Mining	CUDA, OMP, OCL	
Particle Filter	Structured Grid	Medical Imaging	CUDA, OMP, OCL	
PathFinder	Dynamic Programming	Grid Traversal	CUDA, OMP, OCL	
Gaussian Elimination	Dense Linear Algebra	Linear Algebra	CUDA, OCL	
k-Nearest Neighbors	Dense Linear Algebra	Data Mining	CUDA, OMP, OCL	
LavaMD2	N-Body	Molecular Dynamics	CUDA, OMP, OCL	
Myocyte	Structured Grid	Biological Simulation	CUDA, OMP, OCL	
B+ Tree	Graph Traversal	Search	CUDA, OMP, OCL	
GPUDWT	Spectral Method	Image/Video Compression	CUDA, OCL	
Hybrid Sort	Sorting	Sorting Algorithms	CUDA, OCL	
Hotspot3D	Structured Grid	Physics Simulation	CUDA, OCL, OMP	Hotspot for 3D IC
Huffman	Finite State Machine	Lossless data compression	CUDA, OCL	

Adding a log PASS to GCC



C Source Code

Parser

AST

Gimplifier

GIMPLE

Linearizer

Lower

CFG Generator

CFG

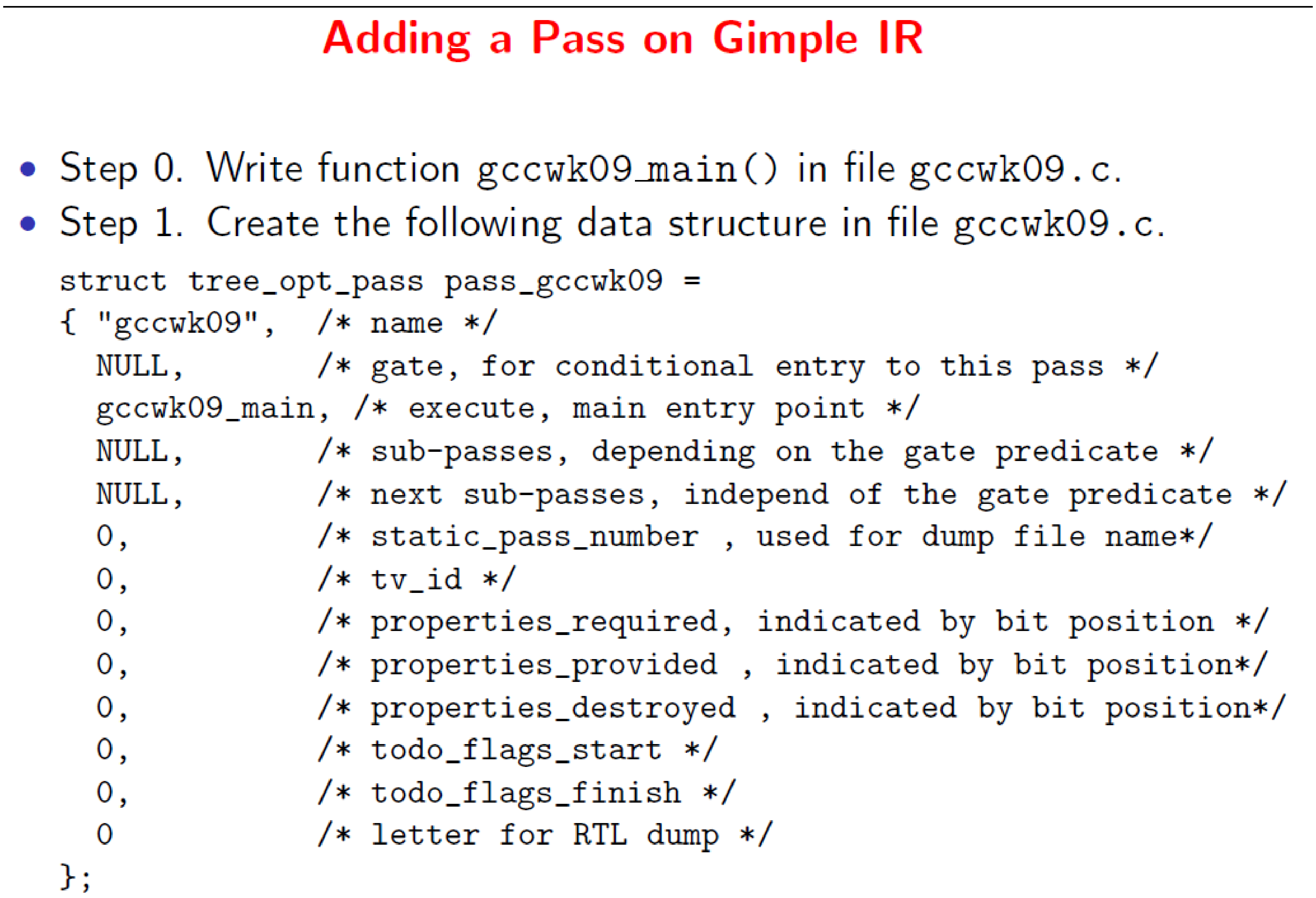
RTL Generator

RTL expand

Write an analysis pass to be added to the compilation pipeline of GCC that collects some information from a program by traversing the GIMPLE IR:

- *Count the number of copy statements in a program*
- *Count the number of variables declared “const” in the program*
- *Count the number of occurrences of arithmetic operators in the program*
- *Count the number of references to global variables in the program*

Adding a log PASS to GCC

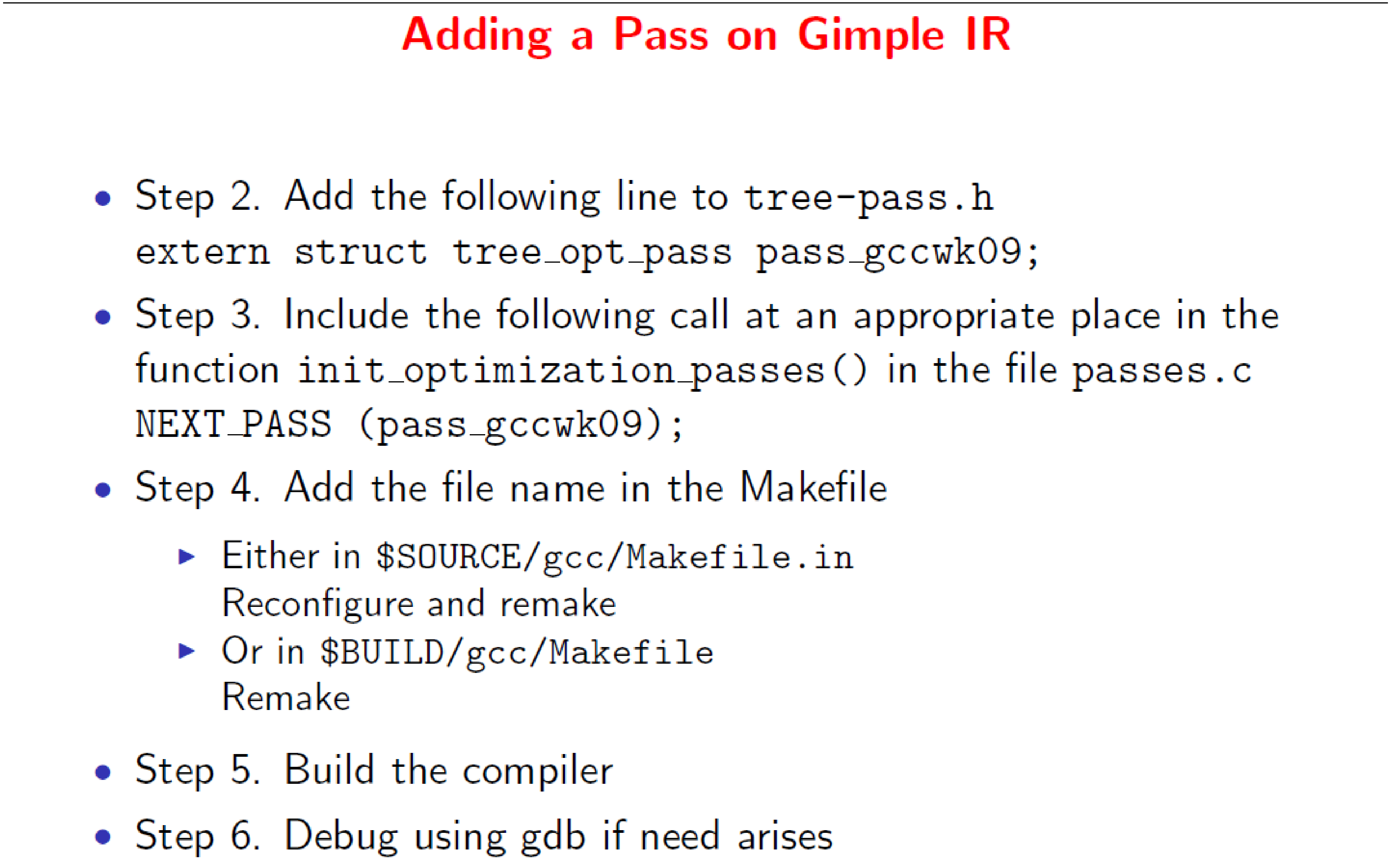


Adding a Pass on Gimple IR

- Step 0. Write function gccwk09_main() in file gccwk09.c.
- Step 1. Create the following data structure in file gccwk09.c.

```
struct tree_opt_pass pass_gccwk09 =
{ "gccwk09",   /* name */
  NULL,        /* gate, for conditional entry to this pass */
  gccwk09_main, /* execute, main entry point */
  NULL,        /* sub-passes, depending on the gate predicate */
  NULL,        /* next sub-passes, independ of the gate predicate */
  0,           /* static_pass_number , used for dump file name*/
  0,           /* tv_id */
  0,           /* properties_required, indicated by bit position */
  0,           /* properties_provided , indicated by bit position*/
  0,           /* properties_destroyed , indicated by bit position*/
  0,           /* todo_flags_start */
  0,           /* todo_flags_finish */
  0            /* letter for RTL dump */
};
```

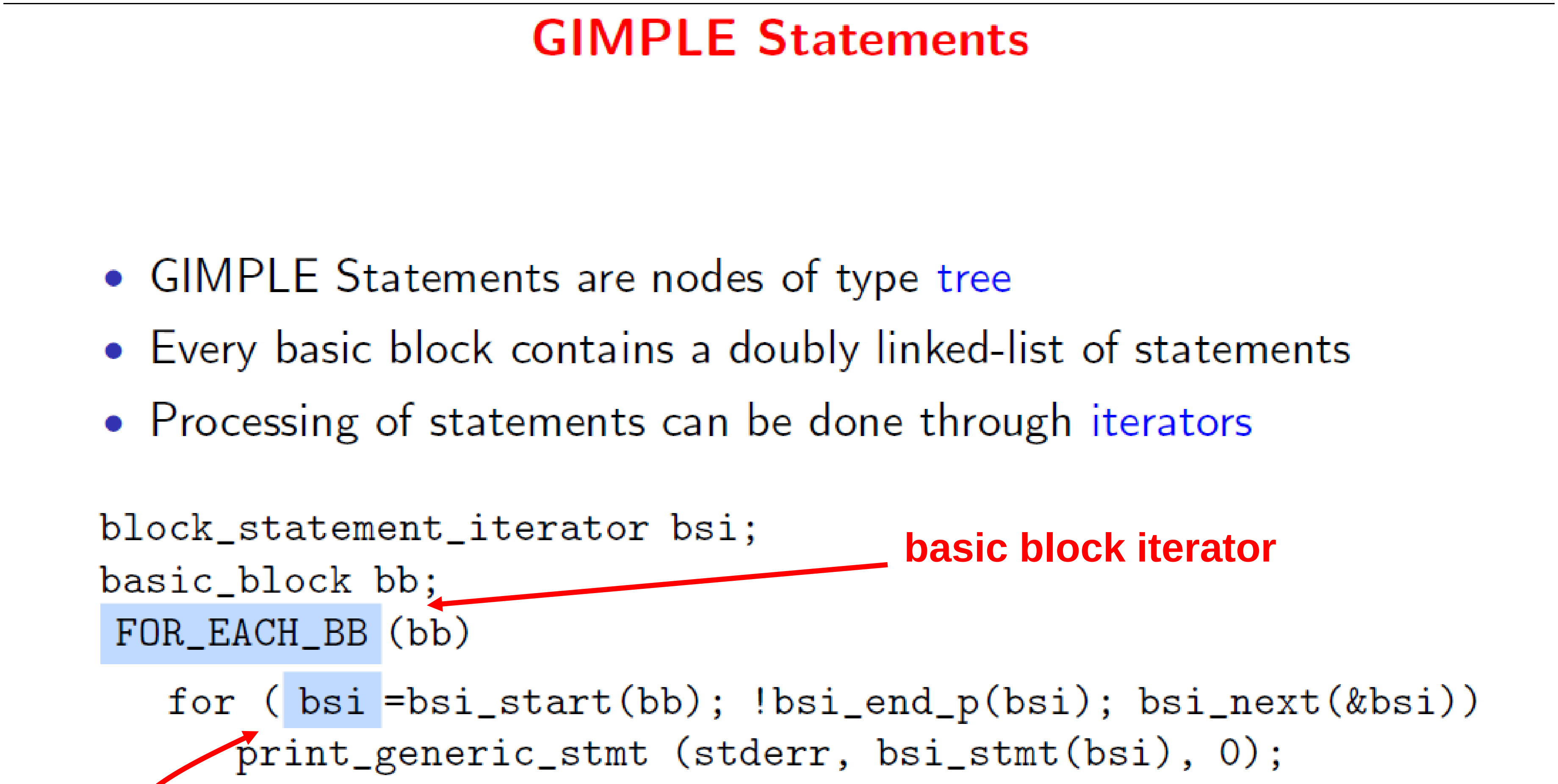
Adding a log PASS to GCC



Adding a Pass on Gimple IR

- Step 2. Add the following line to `tree-pass.h`
  `extern struct tree_opt_pass pass_gccwk09;`
- Step 3. Include the following call at an appropriate place in the function `init_optimization_passes()` in the file `passes.c`
  `NEXT_PASS (pass_gccwk09);`
- Step 4. Add the file name in the Makefile
  - Either in `$SOURCE/gcc/Makefile.in`
    Reconfigure and remake
  - Or in `$BUILD/gcc/Makefile`
    Remake
- Step 5. Build the compiler
- Step 6. Debug using gdb if need arises

Adding a log PASS to GCC



GIMPLE Statements

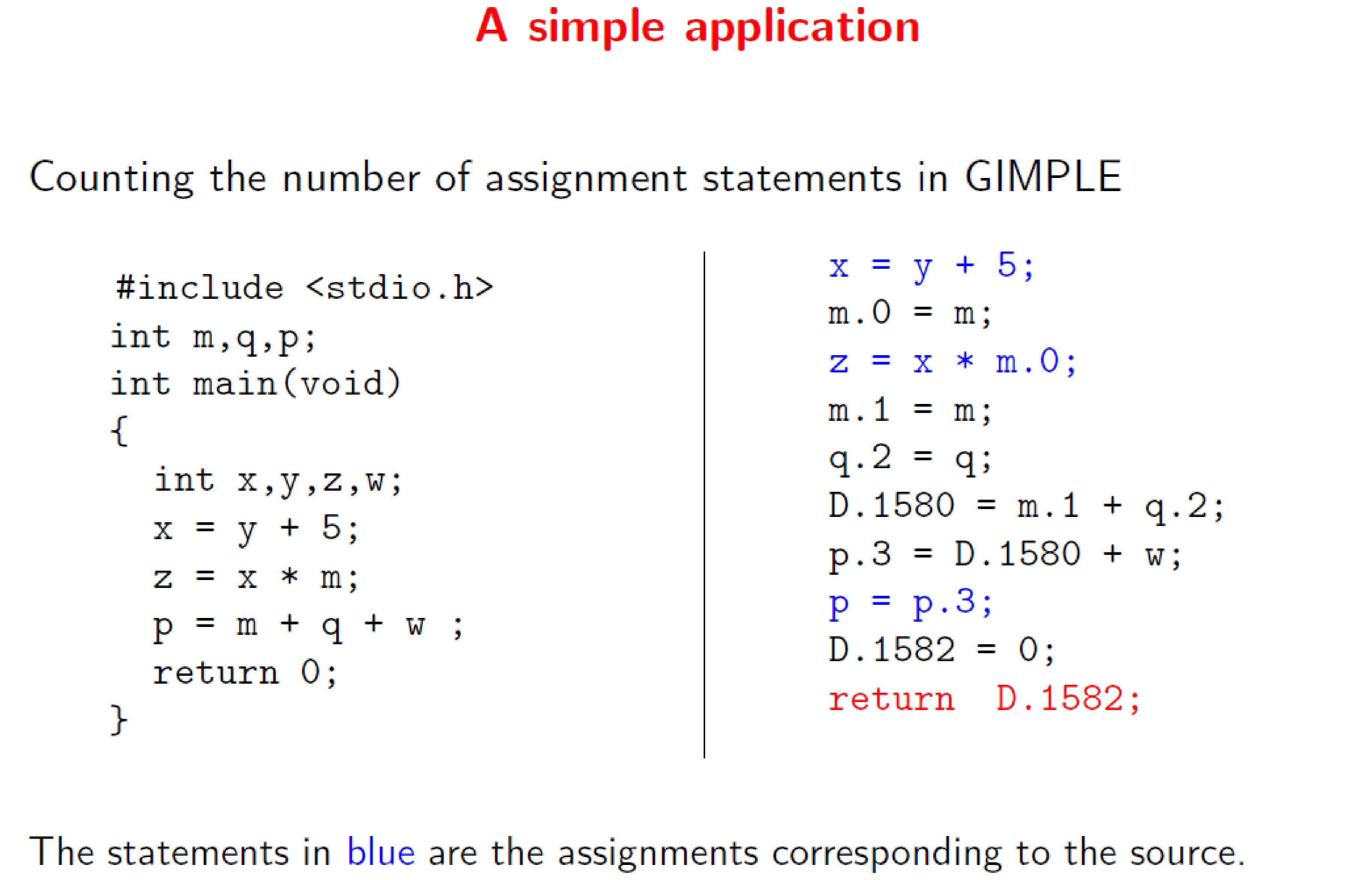
- GIMPLE Statements are nodes of type tree
- Every basic block contains a doubly linked-list of statements
- Processing of statements can be done through iterators

```
block_statement_iterator bsi;
basic_block bb;
FOR_EACH_BB (bb)
   for ( bsi =bsi_start(bb); !bsi_end_p(bsi); bsi_next(&bsi))
      print_generic_stmt (stderr, bsi_stmt(bsi), 0);
```

basic block iterator

statement iterator

Adding a log PASS to GCC



A simple application

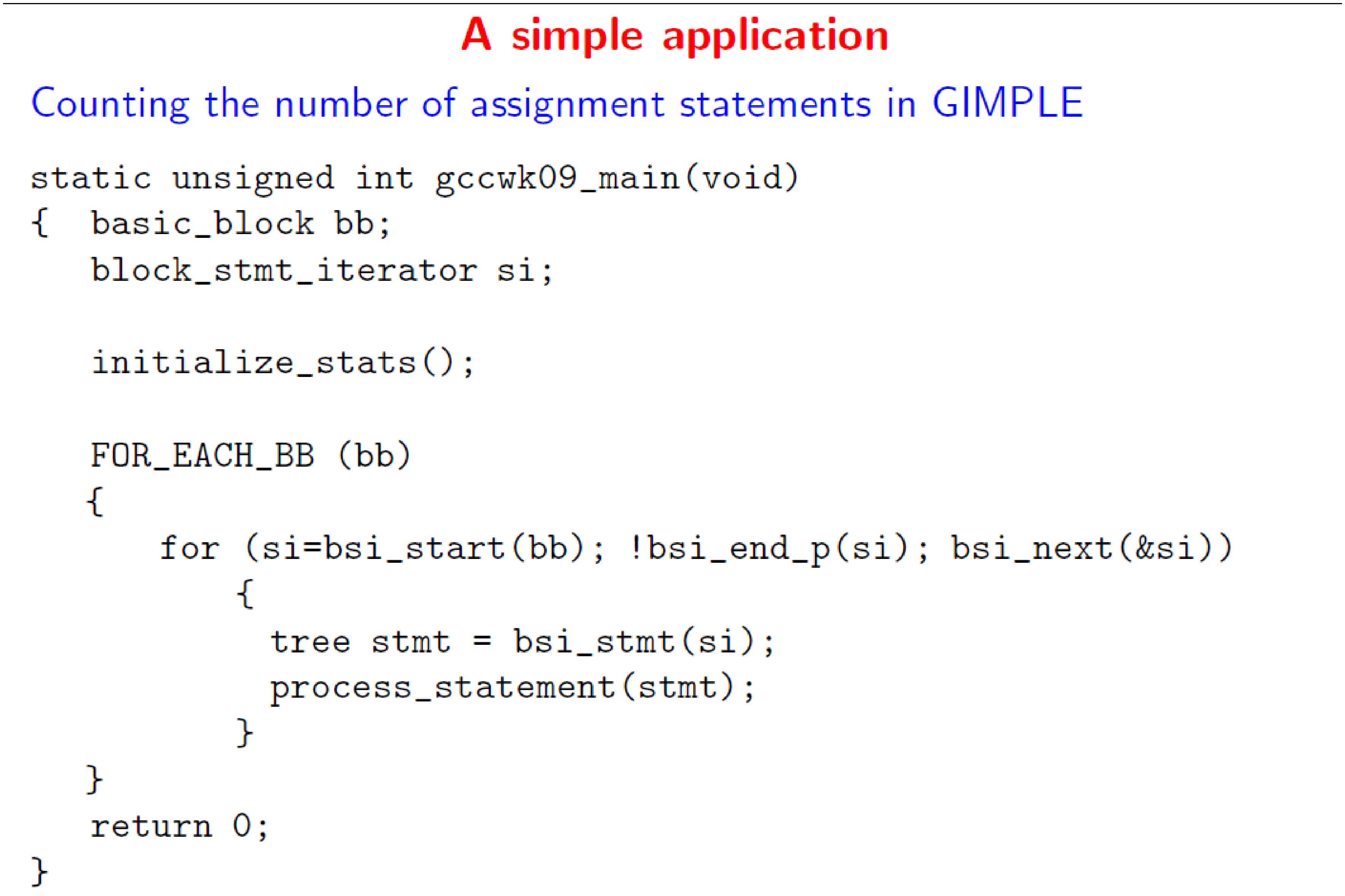
Counting the number of assignment statements in GIMPLE

```
#include <stdio.h>
int m,q,p;
int main(void)
{
  int x,y,z,w;
  x = y + 5;
  z = x * m;
  p = m + q + w ;
  return 0;
}
```

```
x = y + 5;
m.0 = m;
z = x * m.0;
m.1 = m;
q.2 = q;
D.1580 = m.1 + q.2;
p.3 = D.1580 + w;
p = p.3;
D.1582 = 0;
return D.1582;
```

The statements in blue are the assignments corresponding to the source.

Adding a log PASS to GCC



A simple application

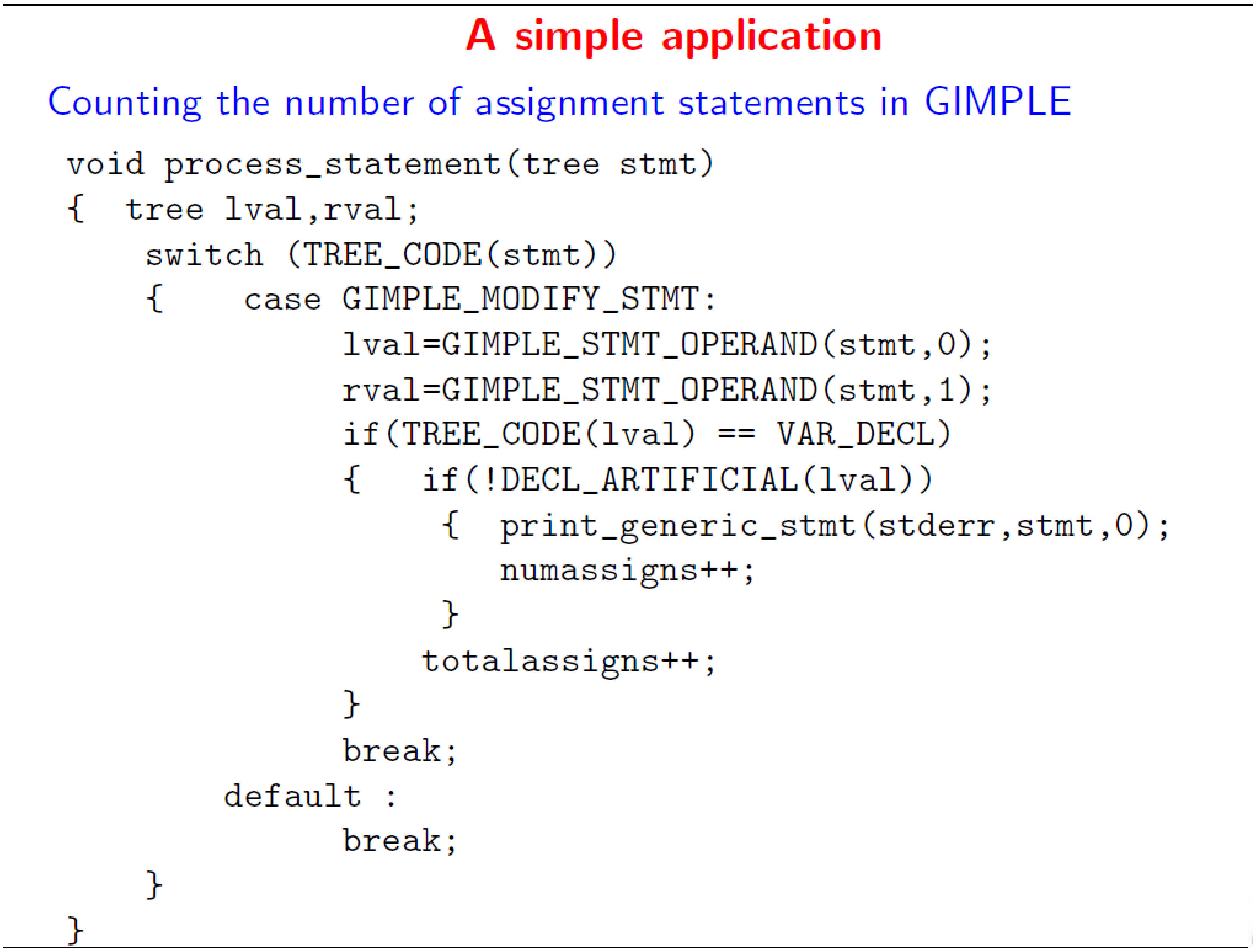
Counting the number of assignment statements in GIMPLE

```
static unsigned int gccwk09_main(void)
{  basic_block bb;
   block_stmt_iterator si;

   initialize_stats();

   FOR_EACH_BB (bb)
   {
       for (si=bsi_start(bb); !bsi_end_p(si); bsi_next(&si))
           {
             tree stmt = bsi_stmt(si);
             process_statement(stmt);
           }
   }
   return 0;
}
```

Adding a log PASS to GCC

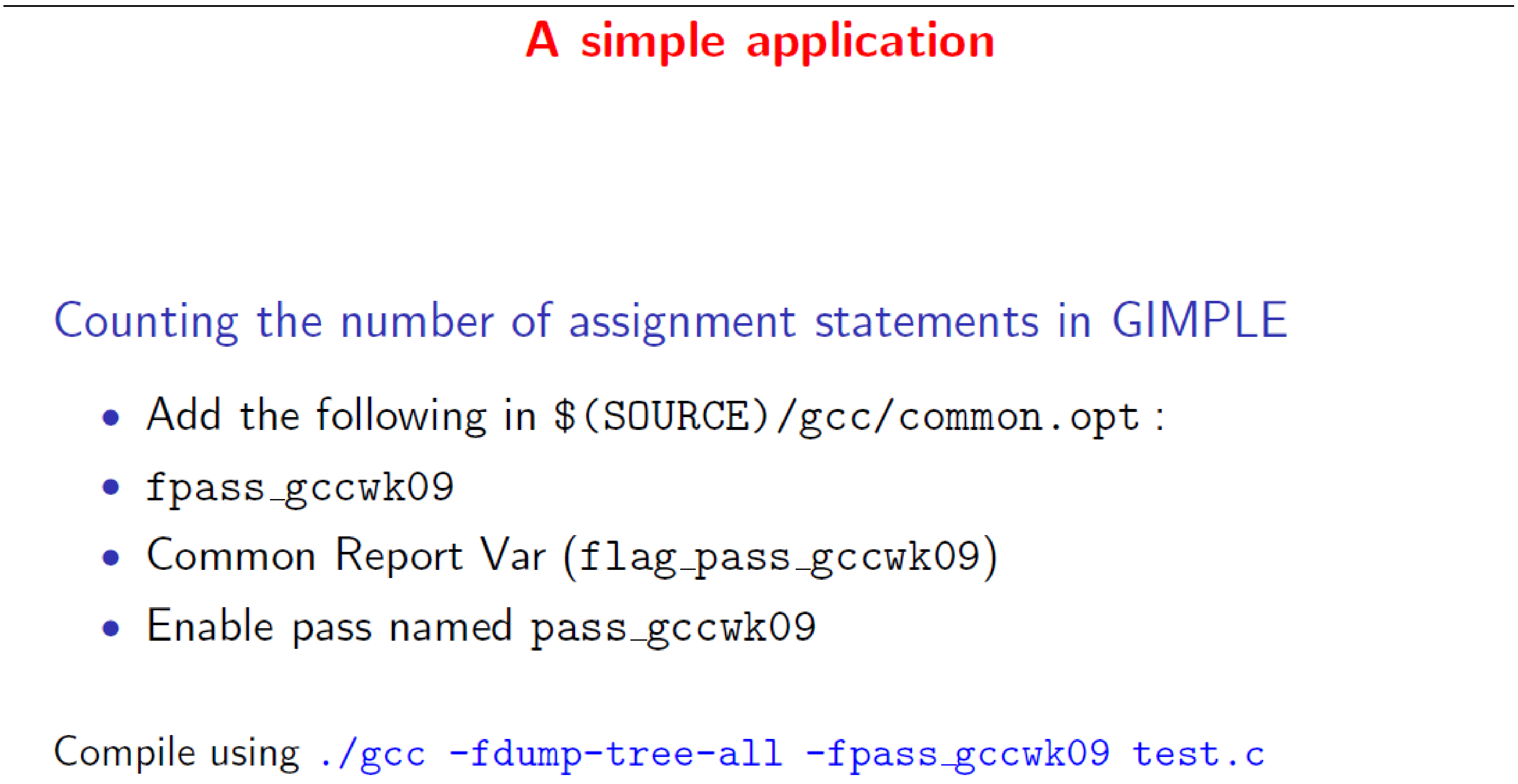


A simple application

Counting the number of assignment statements in GIMPLE

```
void process_statement(tree stmt)
{  tree lval,rval;
    switch (TREE_CODE(stmt))
    {    case GIMPLE_MODIFY_STMT:
              lval=GIMPLE_STMT_OPERAND(stmt,0);
              rval=GIMPLE_STMT_OPERAND(stmt,1);
              if(TREE_CODE(lval) == VAR_DECL)
              {   if(!DECL_ARTIFICIAL(lval))
                   {  print_generic_stmt(stderr,stmt,0);
                      numassigns++;
                   }
                  totalassigns++;
              }
              break;
        default :
              break;
    }
}
```

Adding a log PASS to GCC



A simple application

Counting the number of assignment statements in GIMPLE

- Add the following in $(SOURCE)/gcc/common.opt :
- fpass_gccwk09
- Common Report Var (flag_pass_gccwk09)
- Enable pass named pass_gccwk09

Compile using `./gcc -fdump-tree-all -fpass_gccwk09 test.c`

API Reference

- http://gcc.gnu.org/onlinedocs/gccint.pdf Pg- 233-235
- Refere the same document for some detailed documentation