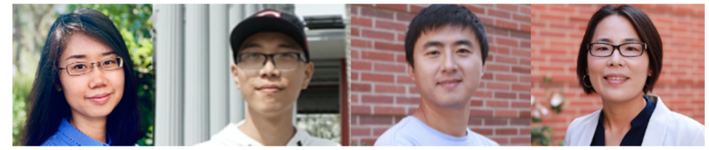


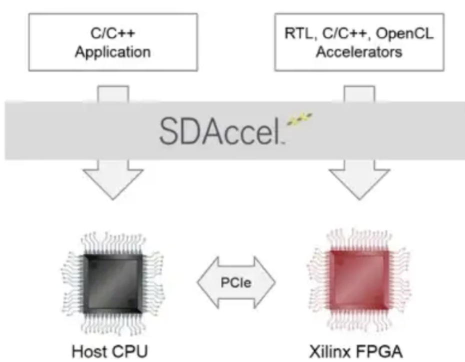
HeteroGen: Transpiling C to Heterogeneous HLS Code with Automated Test Generation and Program Repair

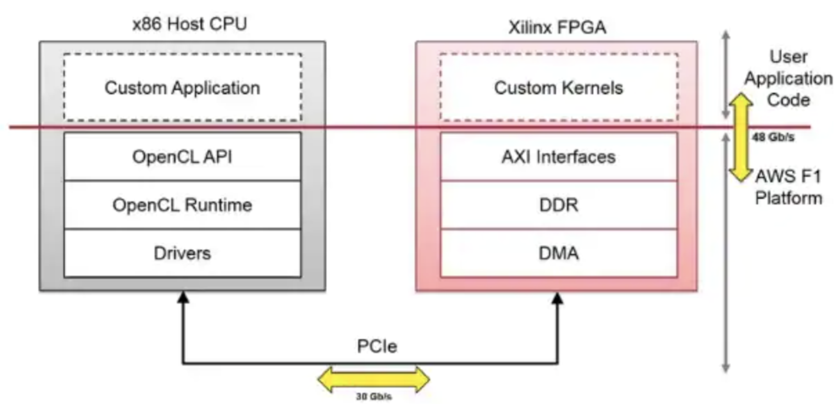
Qian Zhang, **Jiyuan Wang**, Harry Xu, Miryung Kim

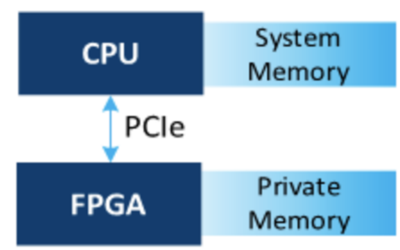
UCLA Samueli School of Engineering

ASPLOS ’22, Feb 28, Lausanne, Switzerland

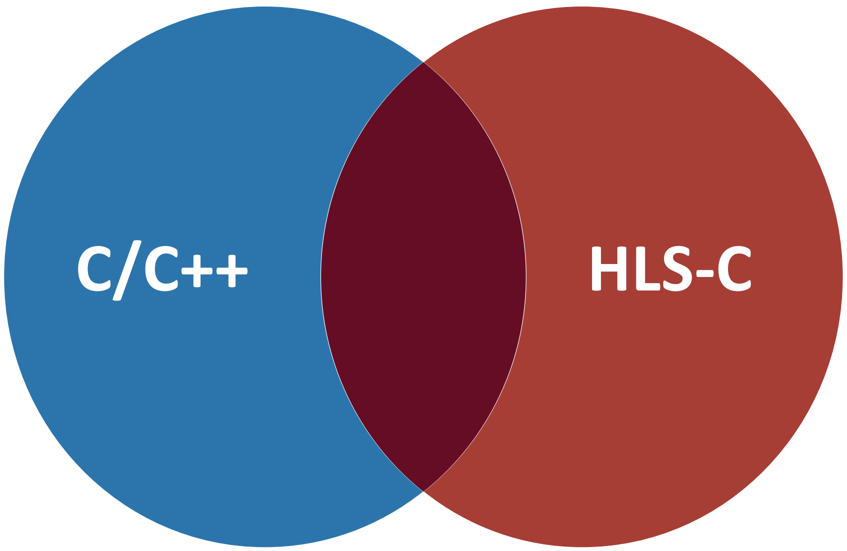
Heterogeneous computing is becoming popular







HLS-C is not standard C/C++

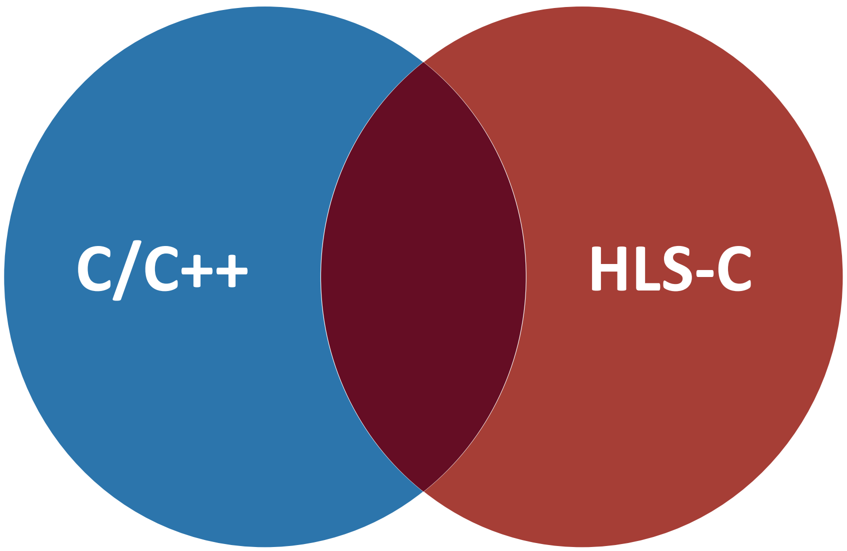


Manual rewriting for synthesizability and optimization

- Resource **finization**
- Hardware expertise and **pragmas** for optimization

UCLA Samueli School of Engineering

Developer tools are not available



- Resource **finization**
- Hardware expertise and **pragmas** for optimization

Manual rewriting for synthesizability and optimization

No developer tools for code translation

What is HLS Development Process Today?

```
int KNN()
   …

// Calculate distance
  for (i = 0 to number)
  {
  dist[i] = l2norm(data[i],
dim);
  }

//Top 1 nearest neighbor
   ...
}
```

7x speedup on FPGA.

1. Performance profiling
2. Kernel function identification
3. Manual **rewriting** for HLS compatibility

   6 minutes HLS compilation

4. **Differential testing** with input samples

   8 minutes CPU-FPGA simulation

   2.5 hours FPGA synthesis

5. Performance Optimization

UCLA Samueli School of Engineering

HLS-C requires specifying *bitwidth* for each type

```
float vecdot(
    float a[],
    float b[],
    int n) {
    for (int i = 0; i < n;
i++)
        sum += a[i] * b[i];
    return sum;
}
```

C Program

```
float vecdot(
    float a[],
    float b[],
    fpga_int<7> n) {
    for (fpga_int<7> i = 0;
i < n; i++)
        sum += a[i] * b[i];
    return sum;
}
```

HLS-C Program

UCLA Samueli School of Engineering

HLS-C uses a custom floating point type

```
float vecdot(
    float a[],
    float b[],
         fpga_int<7> n) {
    for (fpga_int<7> i = 0; i
< n; i++)
        sum += a[i] * b[i];
    return sum;
}
```

C Program

```
fpga_float<8,15> vecdot(
    fpga_float<8,15> a[],
    fpga_float<8,15> b[],
         fpga_int<7> n) {
    for (fpga_int<7> i = 0; i < n;
i++)
         sum += a[i] * b[i];
    return sum;
}
```

HLS-C Program

HLS-C requires *finitizing* resources

```
struct Node {
    Node *left, *right;
    int val; };
void init(Node **root) {
    *root = (Node
*)malloc(sizeof(Node)); }
void delete_tree(Node *root) {...
    free(root); }
void traverse(Node *curr) {
    if (curr == NULL) return;
    int ret = visit(curr->val);
    traverse(curr->left);
    traverse(curr->right);
}
```

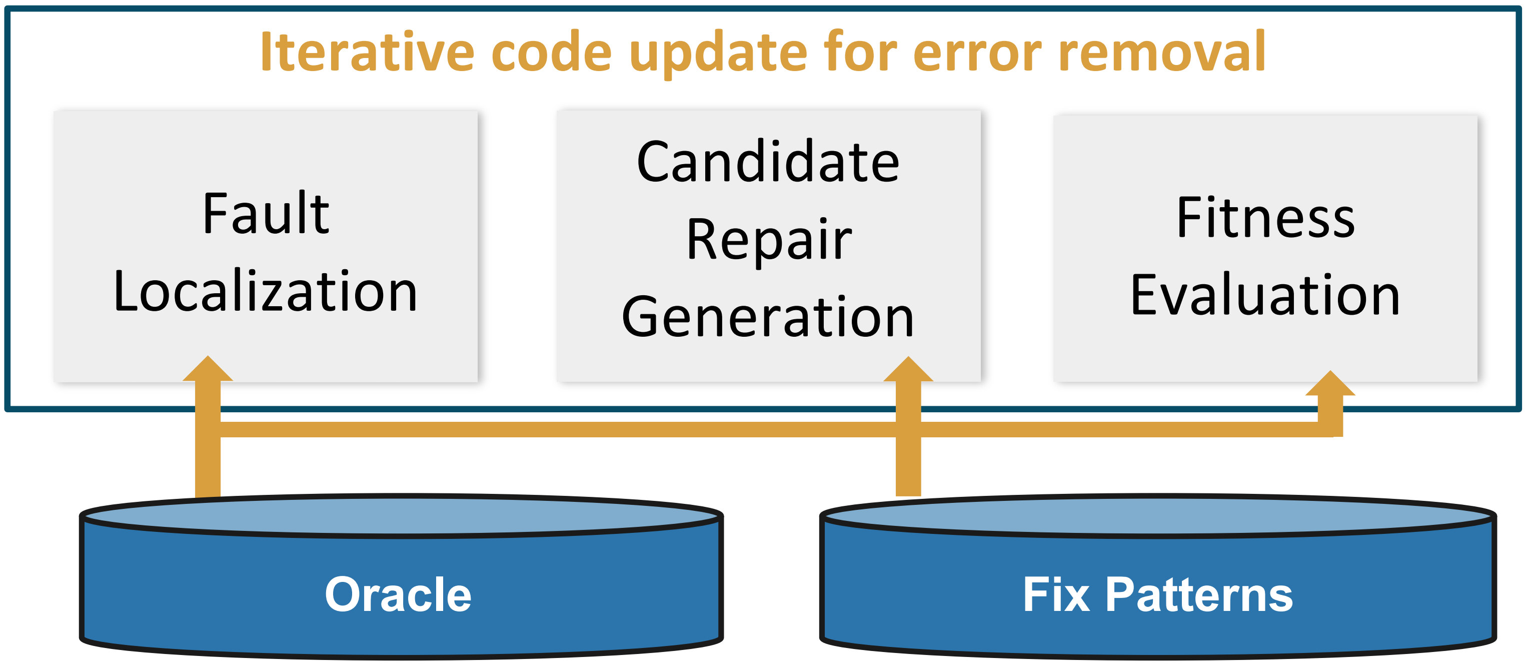
4 errors

C Program

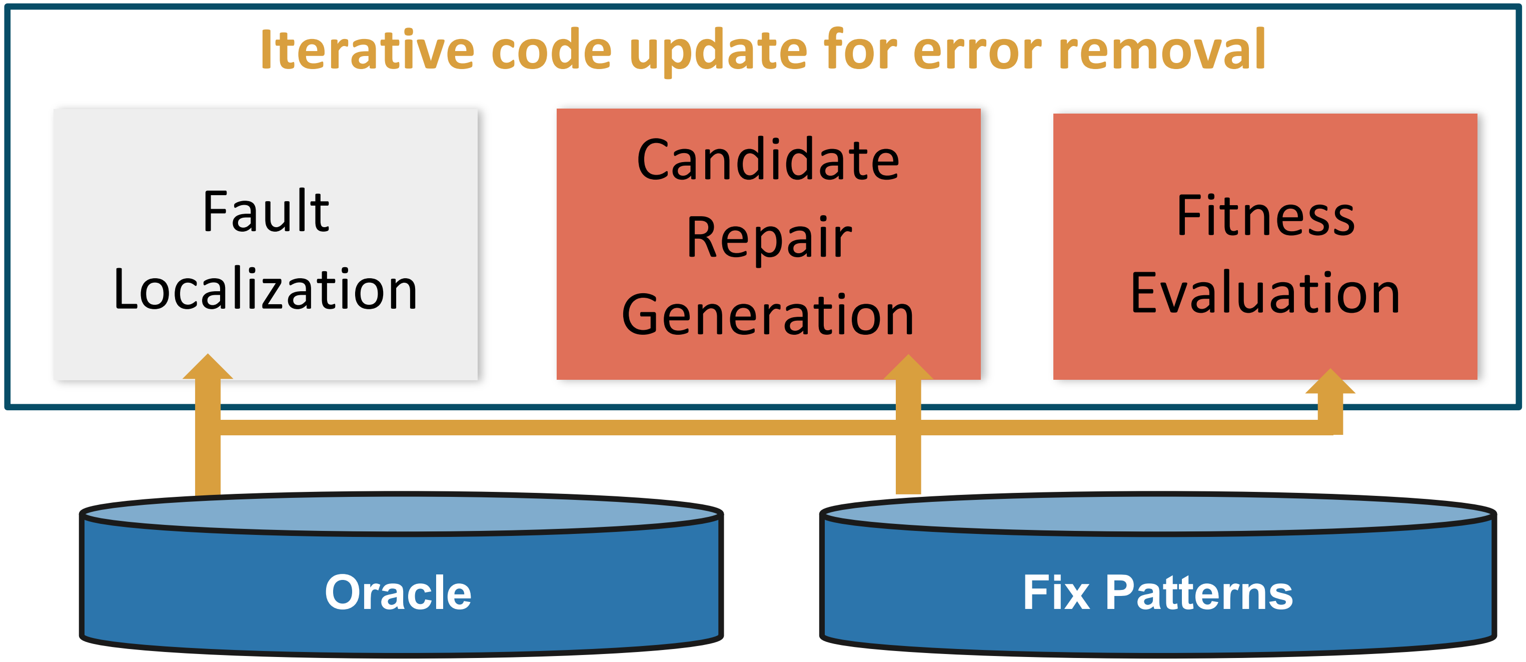
```
Node Node_arr[NODE_ARR_SIZE];
struct Node {
    Node *left, *right;
    int val; };
void init(Node_ptr *root) {
    *root =
(Node_ptr)node_malloc(sizeof(Node)
); }
void delete_tree(Node_ptr root)
{...
    node_free(root); }
void traverse_converted(Node_ptr
curr) {
        stack<context>
s(STACK_SIZE);
        while (!s.empty()) {
    ...}}
```

HLS-C Program

Automated Program Repair (2008~current)



Challenges: Long Compilation Time

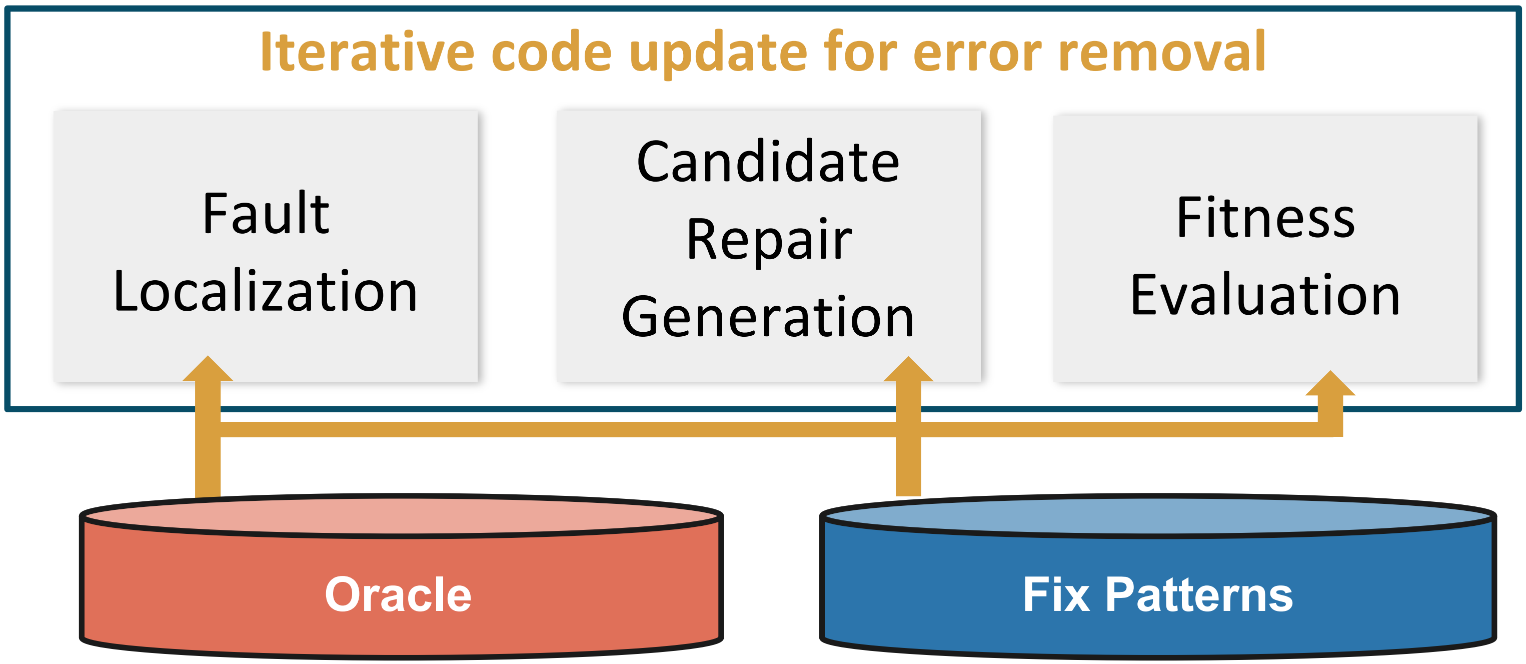


0.2s in GenProg[1] *vs.* 14 minutes in HLS

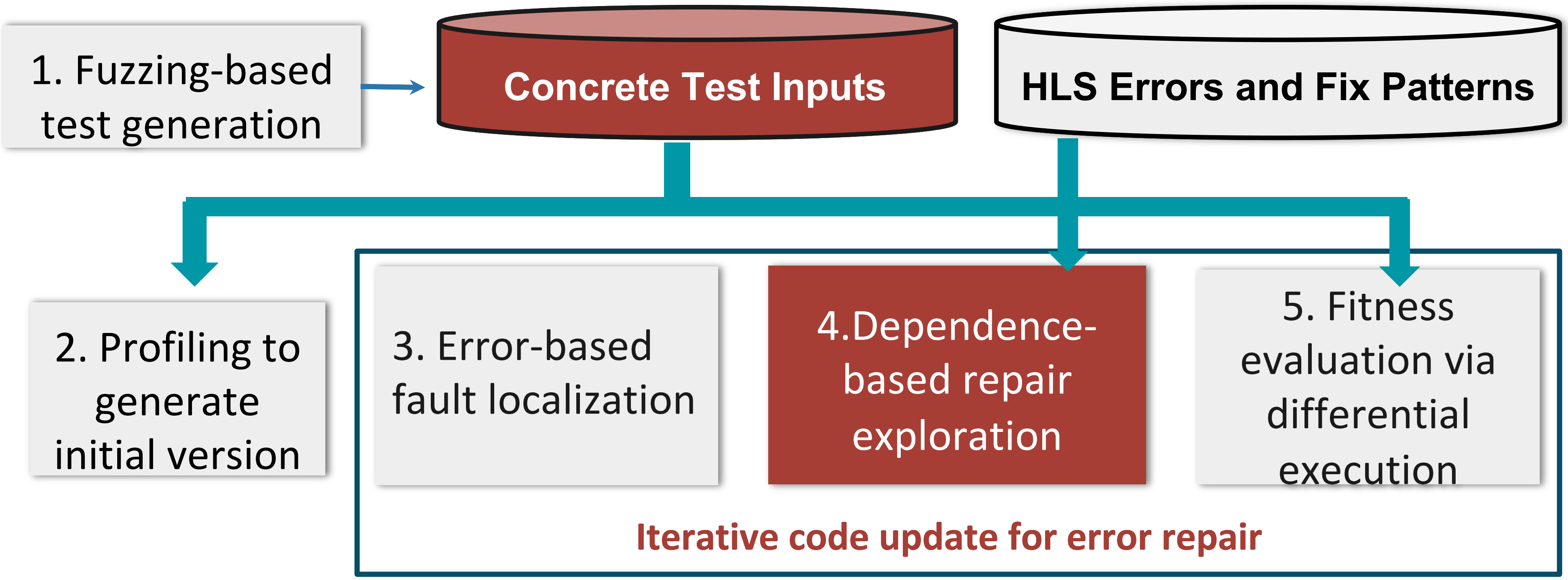
UCLA Samueli School of Engineering

1. Le Goues, Claire, et al. Genprog: A generic method for automatic software repair.

Challenges: Check Behavior Equivalence



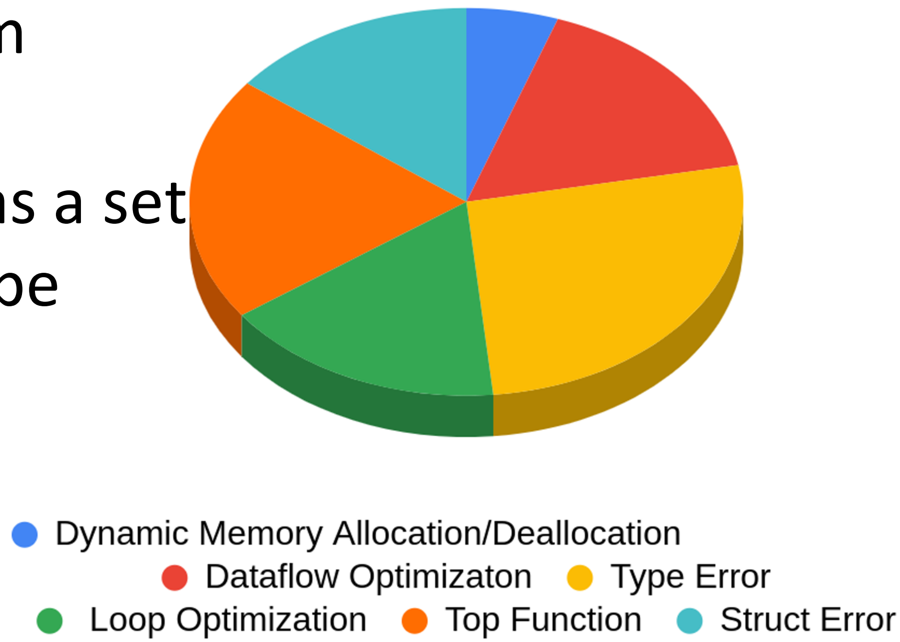
HeteroGen Overview



Observation 1: HLS repairs have patterns.

- HLS compatibility rewrite patterns from real-world user posts from Xilinx.
- We represent the repair edits as a set of **parameterized AST** edits to be concretized to a given context.

static_stack($a1:val)



An error study based on 1000 posts.

UCLA Samueli School of Engineering

Observation 2: HLS repairs require dependent edits.

```
struct Node {
    Node *left, *right;
    int val; };
void init(Node **root) {
    *root = (Node
*)malloc(sizeof(Node)); }
void delete_tree(Node *root) {...
    free(root); }
void traverse(Node *curr) {
    if (curr == NULL) return;
    int ret = visit(curr->val);
    traverse(curr->left);
    traverse(curr->right);
}
```

unsupported recursion

C Program

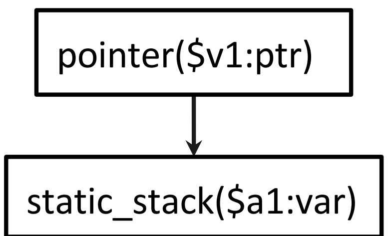
static_stack($a1:var)

Observation 2: HLS repairs require dependent edits.

```
struct Node {
    Node *left, *right;
    int val; };
void init(Node **root) {
    *root = (Node
*)malloc(sizeof(Node)); }
void delete_tree(Node *root) {...
    free(root); }
void traverse(Node *curr) {
    if (curr == NULL) return;
    int ret = visit(curr->val);
    traverse(curr->left);
    traverse(curr->right);
}
```

prerequisite edit

C Program

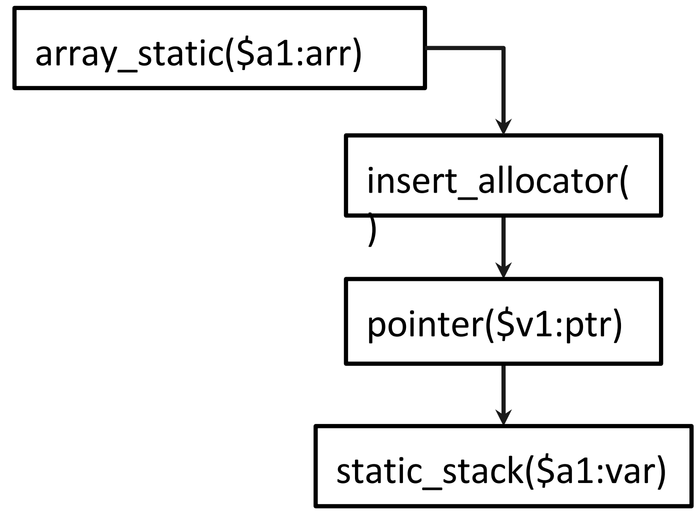


Observation 2: HLS repairs require dependent edits.

```
struct Node {
    Node *left, *right;
    int val; };
void init(Node **root) {
    *root = (Node
*)malloc(sizeof(Node)); }
void delete_tree(Node *root) {...
    free(root); }
void traverse(Node *curr) {
    if (curr == NULL) return;
    int ret = visit(curr->val);
    traverse(curr->left);
    traverse(curr->right);
}
```

prerequisite edit

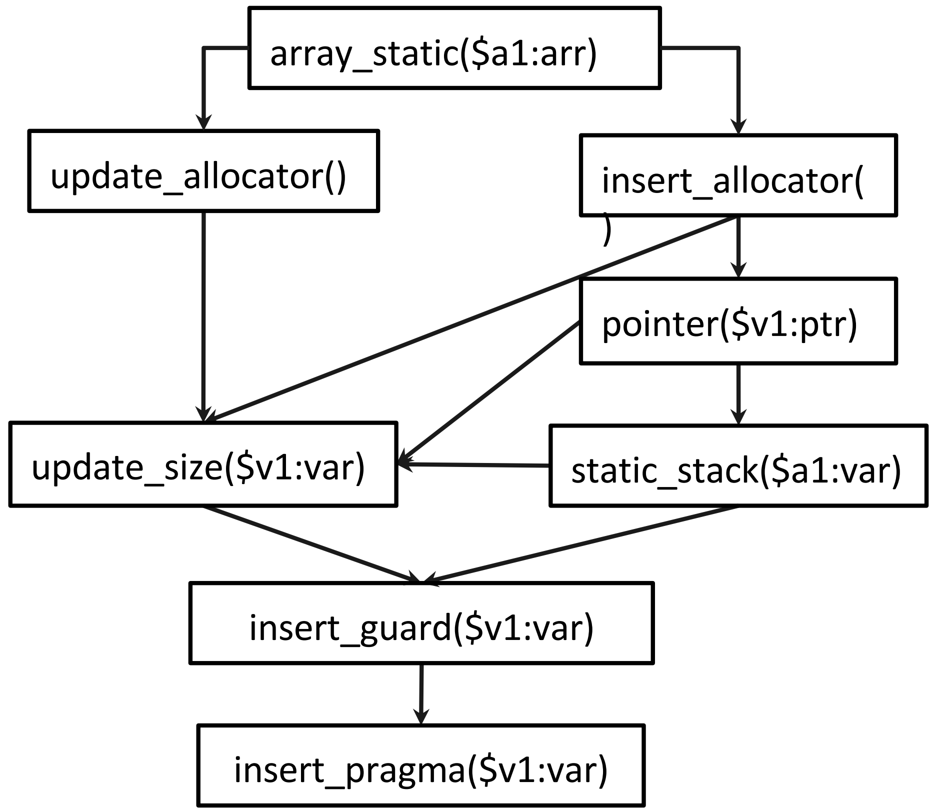
C Program



Observation 2: HLS repairs require dependent edits.

```
struct Node {
    Node *left, *right;
    int val; };
void init(Node **root) {
    *root = (Node
*)malloc(sizeof(Node)); }
void delete_tree(Node *root) {...
    free(root); }
void traverse(Node *curr) {
    if (curr == NULL) return;
    int ret = visit(curr->val);
    traverse(curr->left);
    traverse(curr->right);
}
```

C Program



UCLA Samueli School of Engineering

Optimization 1: Early Repair Candidate Rejection

- **Early Candidate Rejection**
  - If a repair does not conform to HLS coding styles, it does not need to be compiled

14 mins full synthesis and simulation

vs.

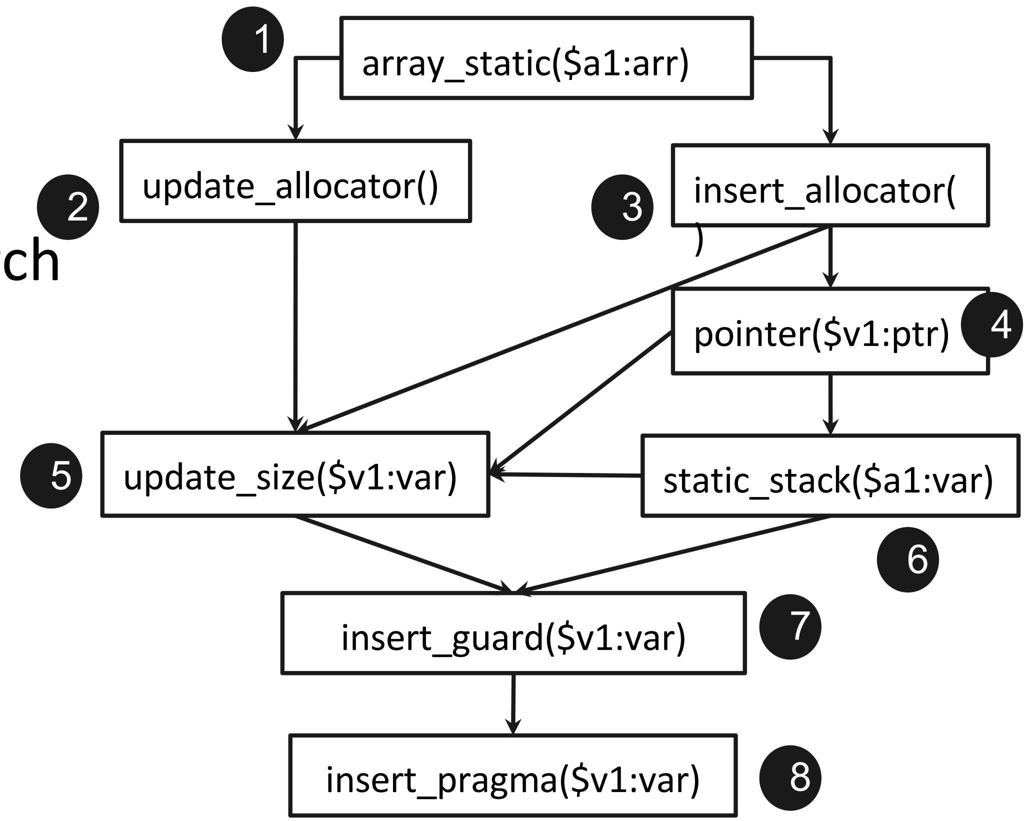
1 second conformance checking

```
void foo (...) {
  int8 array1[M];
  int12 array2[N];
  ...
  #pragma HLS unroll
skip_exit_check factor=4
  loop_2: for(i=0;i<M;i++) {
    array1[i] = ...;
    array2[i] = ...;
    ...
  }
  ...
}
```

UCLA Samueli School of Engineering

Optimization 2: Expedite Search using Dependence

- **Dependence-guided search** helps construct both valid edits and also prune the search space of potential repairs
  - 1
  - 1 -> 2
  - 1 -> 3
  - 1 -> 2 -> 5

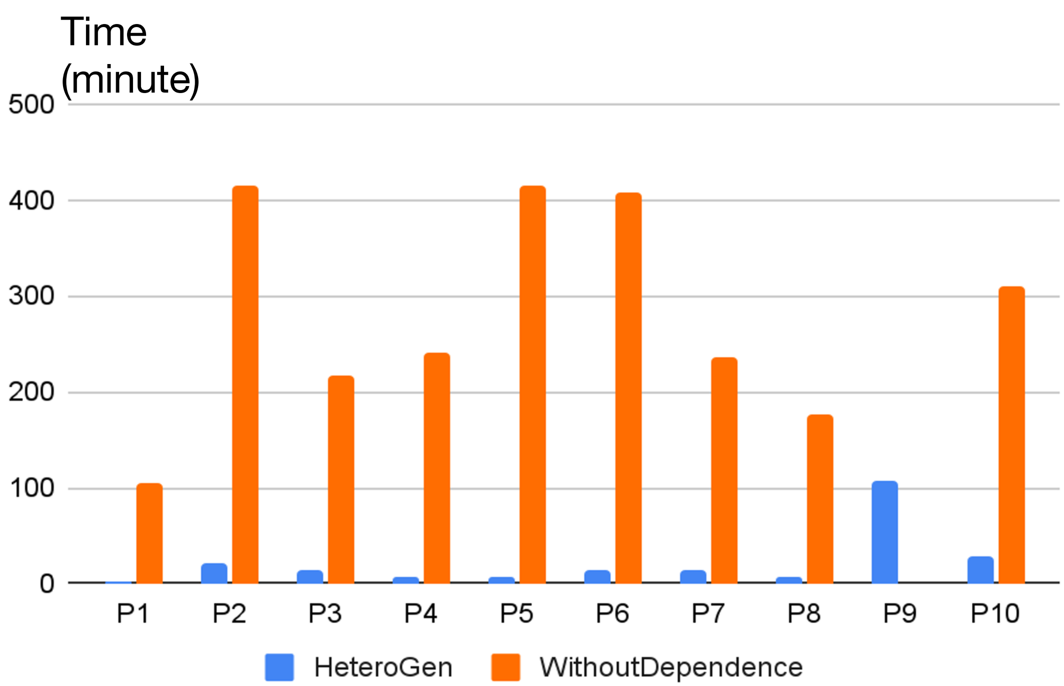


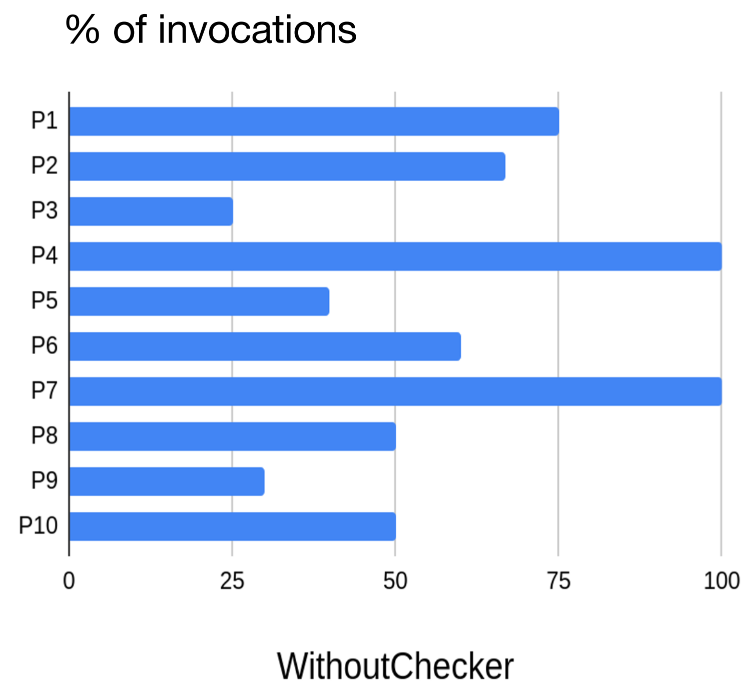
Evaluation: Code Edit

Subject	Edits (LOC)	
	Manual	HeteroGen
Signal Transmission	78	69
Arithmetic Computation	8	9
Merge Sort	276	356
Image Processing	136	32
Graph Traversal	144	438
Matrix Multiplication	25	16
Bubble Sort	45	25
Linked List	156	298
Face Detection	3272	144
Digit Recognition	61	35

HeteroGen preserves test behavior for all subjects. It automates up to 438 line edits, reducing HLS rewriting effort.

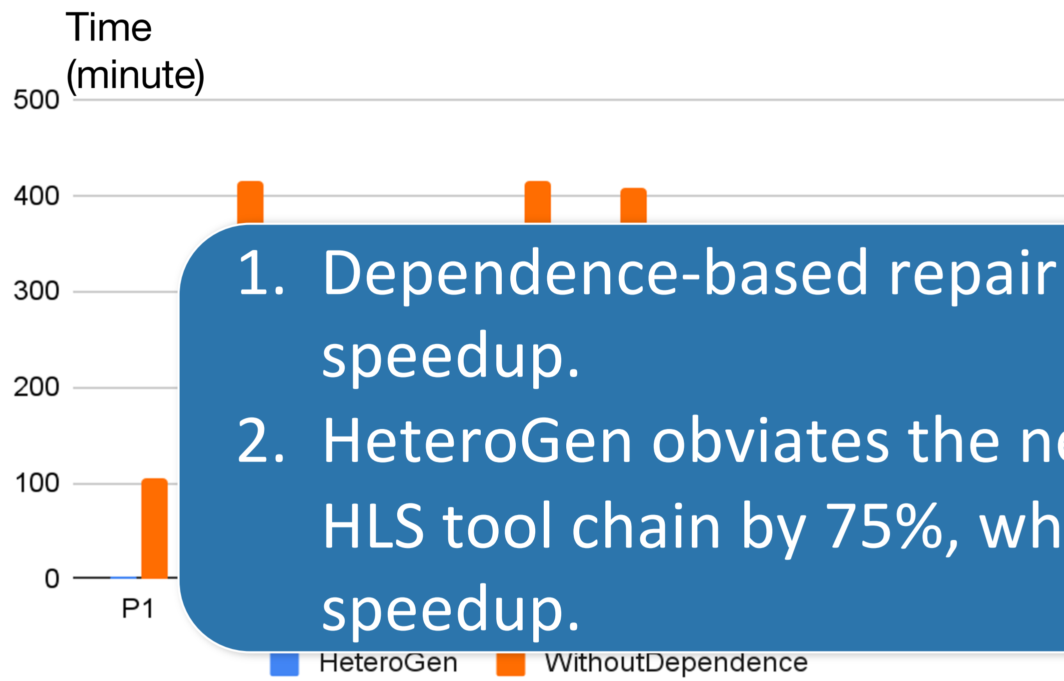
Evaluation: Speedup

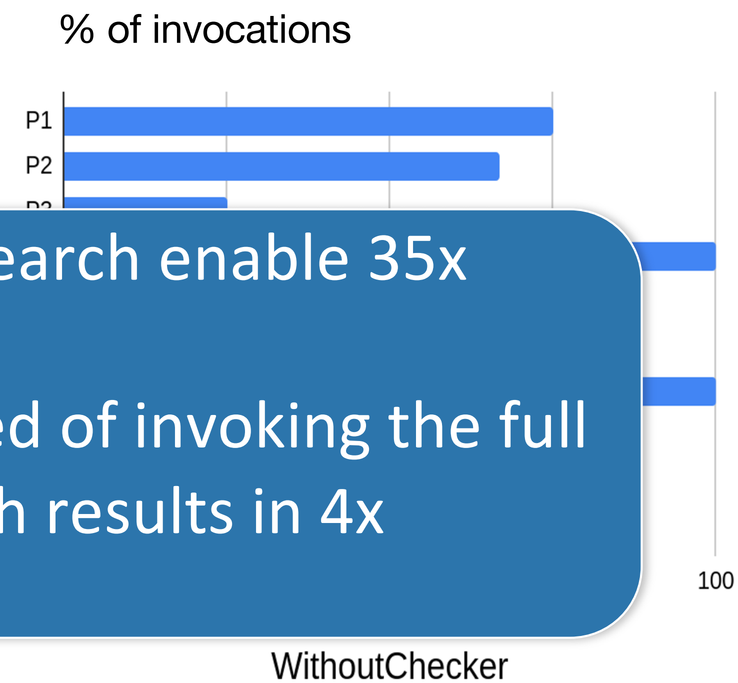




UCLA Samueli School of Engineering

Evaluation: Speedup





1. Dependence-based repair search enable 35x speedup.
2. HeteroGen obviates the need of invoking the full HLS tool chain by 75%, which results in 4x speedup.

UCLA Samueli School of Engineering

Evaluation: Summary

90%

Effectiveness

HeteroGen produces an HLS-compatible version for 9 out of 10.

97%

Coverage

Auto-generated inputs cover 97%, while pre-existing tests reach 36% coverage.

35X

Speed-up

Dependence-based search contributes to 35X speedup than the one without.

~438 lines

Automation

It automates upto 438 lines.

1.64X

Latency

It produces a HLS version 1.63X faster than the original C

UCLA Samueli School of Engineering

Summary

- HeteroGen automatically translates C/C++ to HLS code by search-based code repair
- HeteroGen speeds up the repair process with two optimization:
  - Apply code edits with dependence to reduce search space
  - Code style check to avoid unnecessary compilation process
- HeteroGen Ensure the correctness of translated code by automated testing
- HeteroGen on Github: https://github.com/UCLA-SEAL/HeteroGen

UCLA Samueli School of Engineering

Developer Tools - Test, Refactor, and Repair

HeteroRefactor (ICSE 2020) → Automated refactoring, achieves performance improving and resource saving for FPGA program

HeteroFuzz (ESEC/FSE 2021) → 754X faster in exposing the same set of distinct platform portability errors than naive fuzzing

HeteroGen (ASPLOS 2022)

Developer Tools for Heterogeneous Computing: https://github.com/UCLA-SEAL

Thanks for listening!

Q&A

UCLA Samueli School of Engineering