XDF XILINX DEVELOPER FORUM

Future of FPGA Compilation and Acceleration

Presented By

Dr. Yuxin Wang
Senior R&D Engineer
Oct 16 2018

XILINX

Simplifying acceleration on Xilinx FPGAs since 2014

Founded in 2014 by Dr. Jason Cong, Director of the Center for Domain Specific Computing (CDSC) in UCLA

Products & Services

Merlin Compiler

ML optimized C/C++ to FPGA for any application

Kestrel Runtime

Dynamic application load balancing across multiple compute resources.

Industry Solutions

End-to-end solutions to address industry specific challenges

Expert Services

Trusted partners and advisors to leverage Falcon platform

Supported Xilinx Platforms

ON-PREM

Alibaba Cloud

amazon web services™

HUAWEI

intel Capital

Baidu 百度

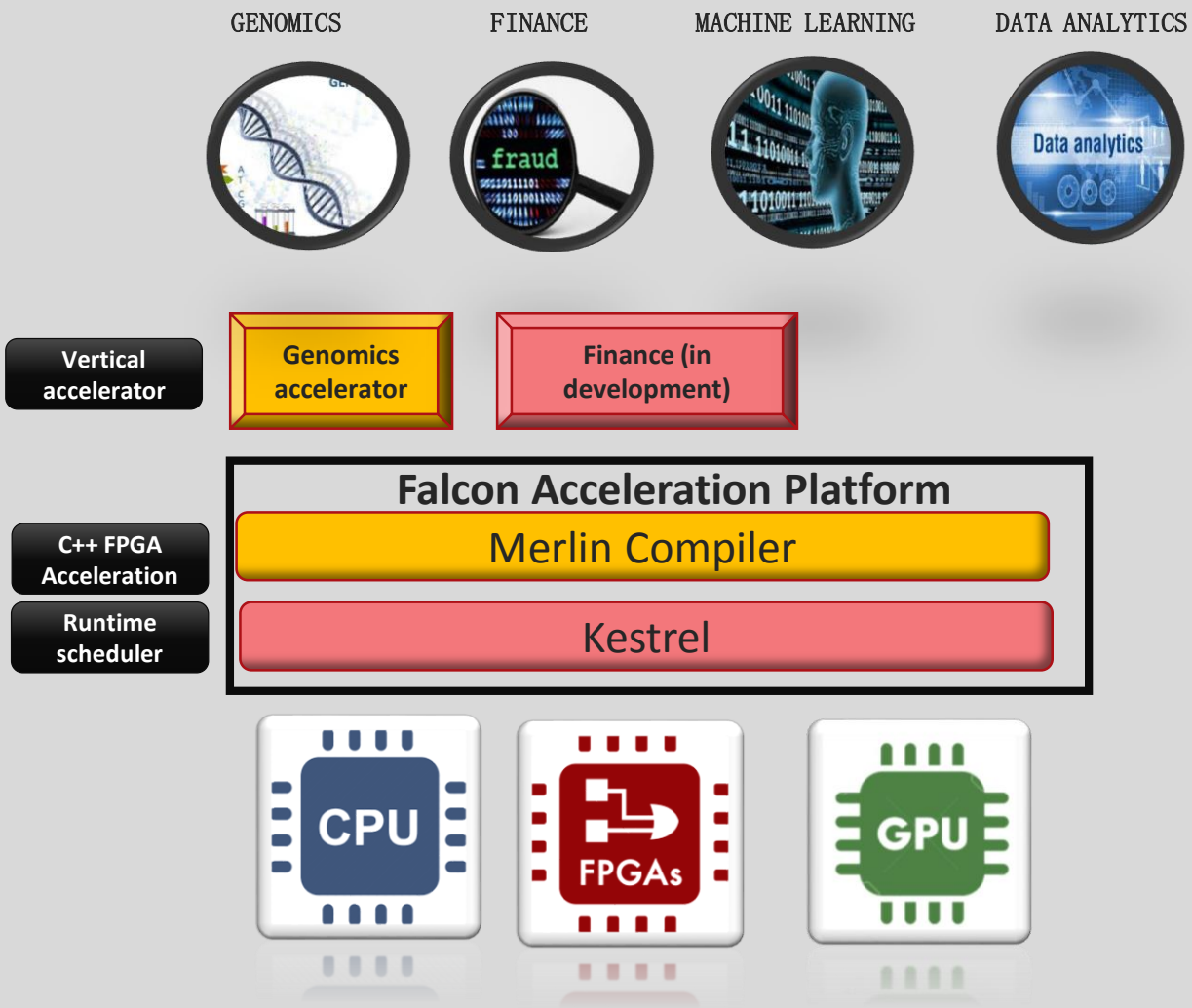
CLEAR

HUAWEI

SIERRA VENTURES

SV TECH VENTURES

Falcon Acceleration Platform



Seamless Acceleration Solutions

Vertical Specific Solutions, C/C++ FPGA Acceleration & Run-Time Environment

- ✓ Accelerated Genomics pipeline
- ✓ Fast C/C++ to FPGA accelerator development
- ✓ Accelerator management for heterogeneous compute clusters.
- ✓ Heterogeneous Platform Support across CPUs and FPGAs today and GPUs in the future

Available on public & private clouds or on-premise

AliCloud amazon web services Baidu Cloud HUAWEI

Acceleration Challenges

Software application developers

- New-gen applications need acceleration hardware
- App Developers need a familiar programming paradigm for these accelerators

GENOMICS

MACHINE LEARNING

FINANCE

DATA ANALYTICS

Hardware developers

- Teams that do have HW expertise, often lack the quantity needed to meet the scale & agility of new application development
- Need greater productivity to meet business needs

Designing Accelerators is Difficult

- C/C++ code first needs to be *converted* to OpenCL or HDL (Verilog/VHDL)
- Accelerators need to be *optimized* for performance

OpenCL C

Local buffers

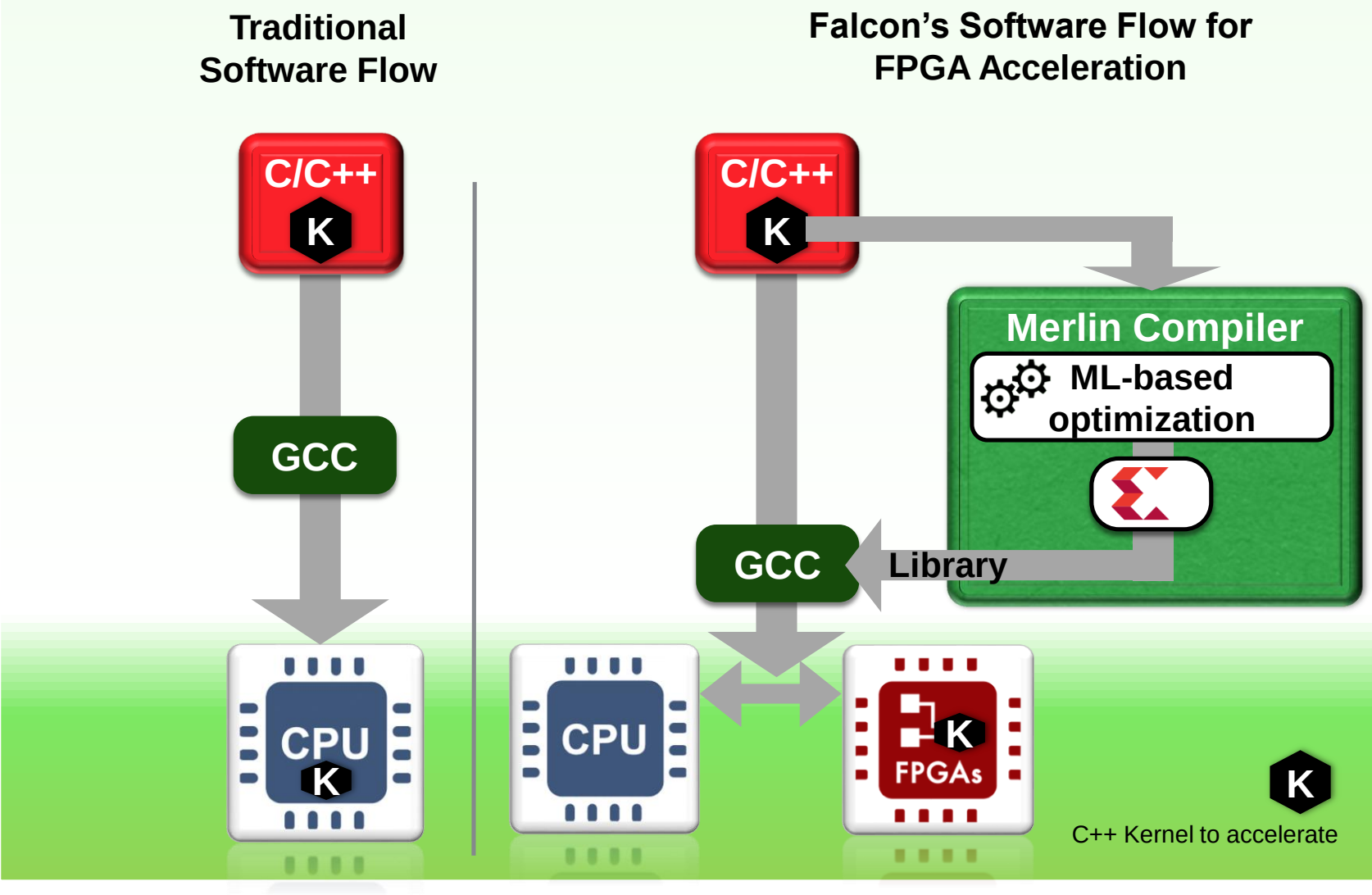
Tiling

Global memory access

```
__kernel
__attribute((reqd_work_group_size(BLOCK_SIZE,BLOCK_SIZE,1)))
__attribute((num_simd_work_items(SIMD_WORK_ITEMS)))
void matrixMult( // Input and output matrices
                 __global float *restrict C,
                 __global float *A,
                 __global float *B,
                 // Widths of matrices.
                 int A_width, int B_width)
{
    // Local storage for a block of input matrices A and B
    __local float A_local[BLOCK_SIZE][BLOCK_SIZE];
    __local float B_local[BLOCK_SIZE][BLOCK_SIZE];

    // Block index
    int block_x = get_group_id(0);
    int block_y = get_group_id(1);

    // Local ID index (offset within a block)
    int local_x = get_local_id(0);
    int local_y = get_local_id(1);

    // Compute loop bounds
    int a_start = A_width * BLOCK_SIZE * block_y;
    int a_end   = a_start + A_width - 1;
    int b_start = BLOCK_SIZE * block_x;

    float running_sum = 0.0f;

    for (int a = a_start, b = b_start; a <= a_end; a += BLOCK_SIZE, b += (BLOCK_SIZE * B_width))
    {
        A_local[local_y][local_x] = A[a + A_width * local_y + local_x];
        B_local[local_x][local_y] = B[b + B_width * local_y + local_x];

        // Wait for the entire block to be loaded.
        barrier(CLK_LOCAL_MEM_FENCE);

        #pragma unroll
        for (int k = 0; k < BLOCK_SIZE; ++k)
        {
            running_sum += A_local[local_y][k] * B_local[local_x][k];
        }

        // Wait for the block to be fully consumed before loading the next
        // block.
        barrier(CLK_LOCAL_MEM_FENCE);
    }

    // Store result in matrix C
    C[get_global_id(1) * get_global_size(0) + get_global_id(0)] = running_sum;
}
```

Matrix multiplication
4 lines of C grows to >50+ lines!!

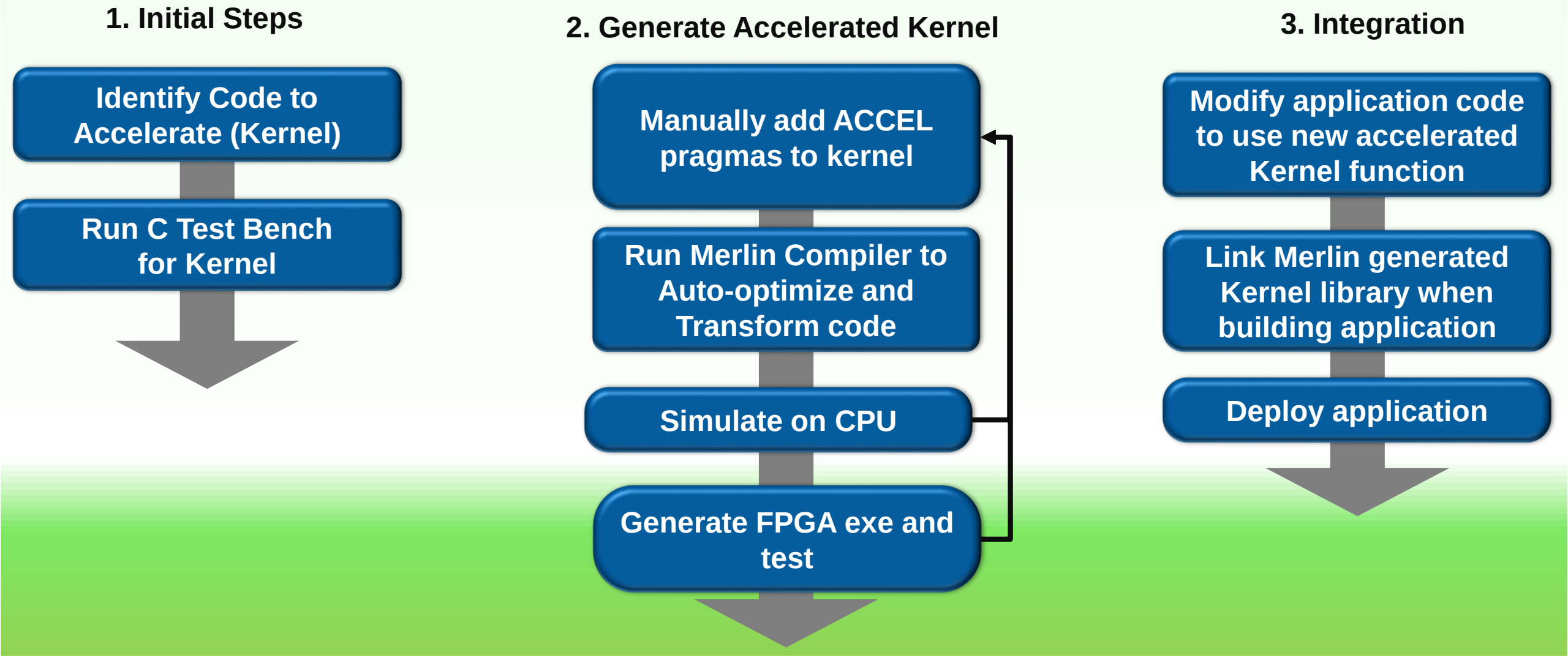
Merlin Compiler Overview



C/C++ FPGA Acceleration

- Pure C/C++ flow
- No FPGA expertise required
- OpenMP programming model with auto-generated 'ACCEL' pragmas
- Automatically performs source to source optimization for accelerated performance

Merlin Compiler C++ FPGA Accelerator Flow – 1 2 3



Software Application Developers can leverage FPGA acceleration with “As-is” code using Merlin

- 5-15X Performance Acceleration over CPUs
- Minimal code changes for acceleration benefits
- Single Acceleration platform across multiple vendor FPGAs & diverse apps such as Computer Vision, ML, Genomics

Out-of-the-box Performance

Example Designs	CPU (ms)	Merlin FPGA (ms)	Merlin Speedup
Black Scholes Asian	477370	34430	**13.9X**
Black Scholes European	116310	8590	**13.5X**
Heston European	341650	34430	**10X**
Heston European Barrier	38630	17220	**2.2X**

Hardware Developers can increase their productivity with Merlin

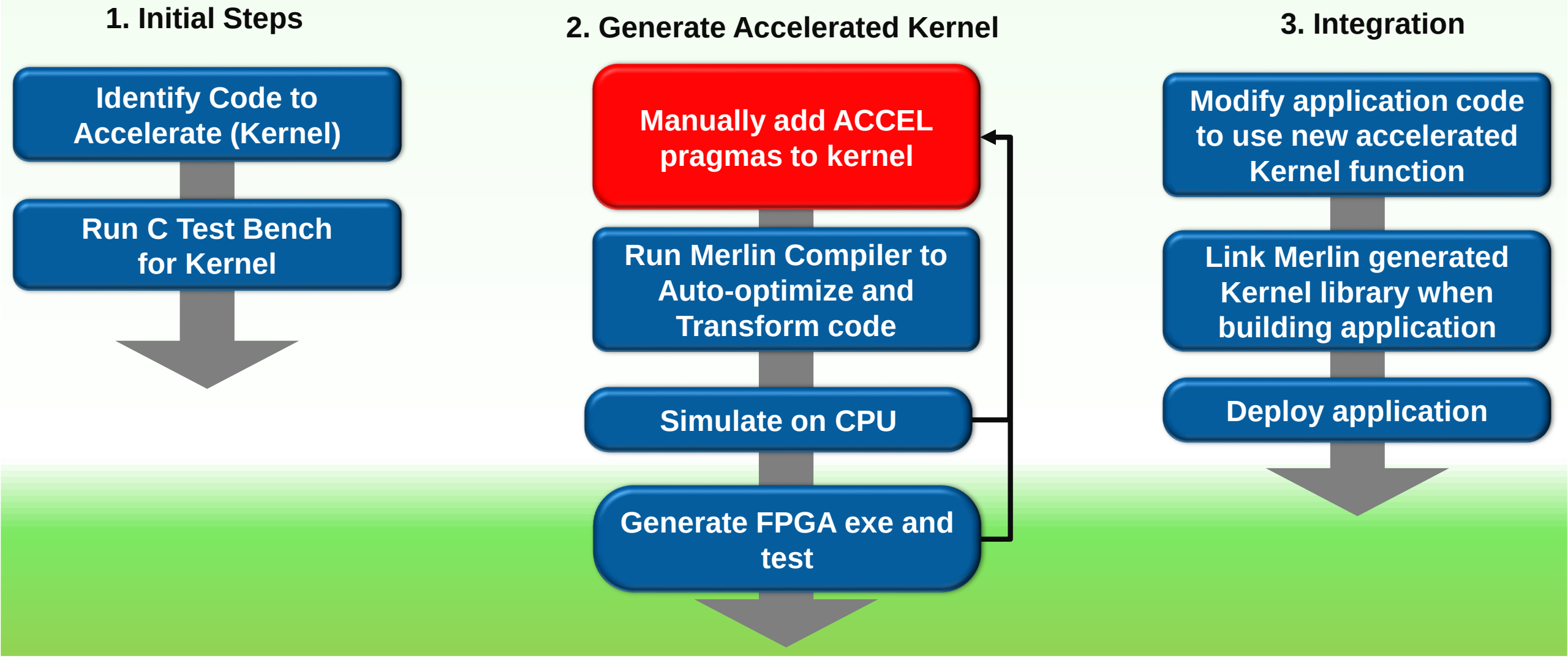
- 6-10X gain in productivity with equivalent performance
- Machine-learning based auto-generated or manually inserted pragmas
- Quality of results comparable to manually optimized OpenCL implementation

Optimized Performance

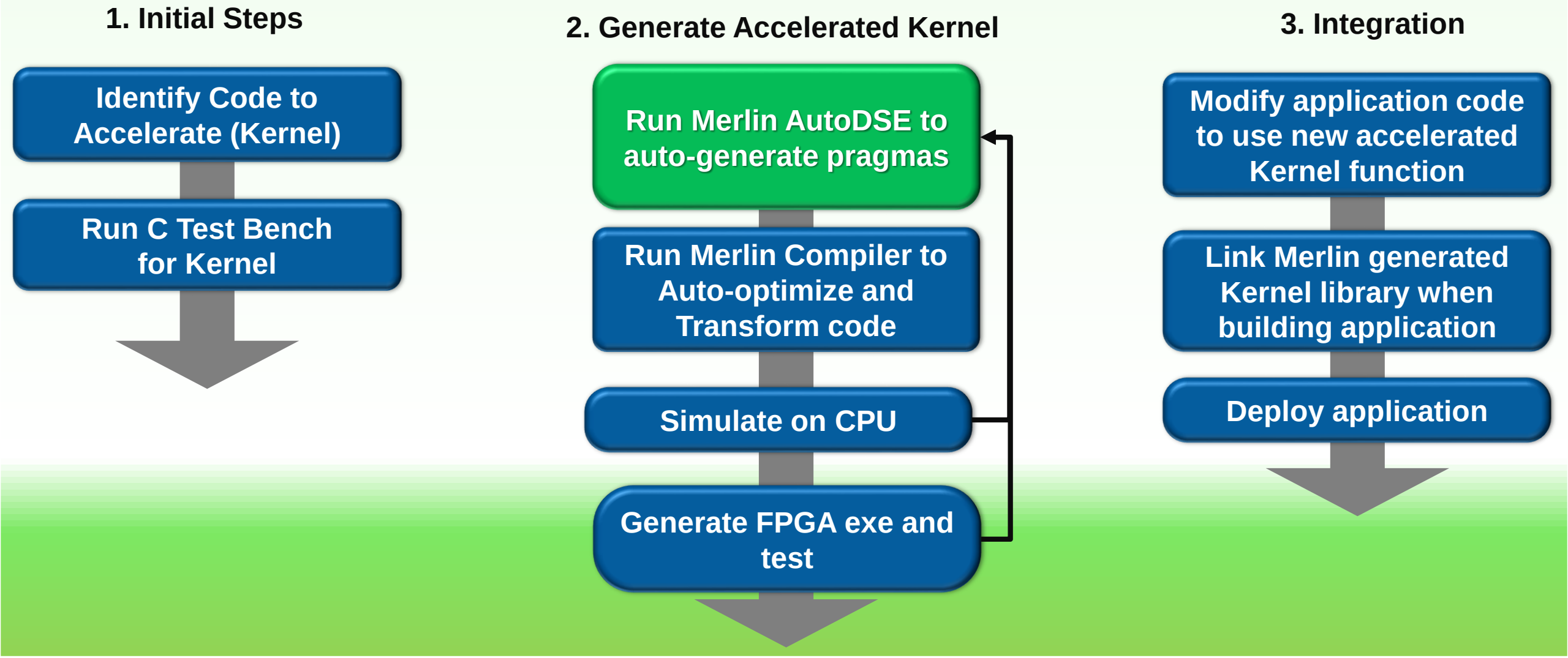
Example Designs	CPU (ms)	Merlin FPGA (ms)	Manual OpenCL (ms)	Merlin Speedup	Merlin Productivity Saving*
Black Scholes Asian	477370	920	830	**519X**	**7.8X**
Black Scholes European	116310	230	230	**506X**	**6.8X**
Heston European	341650	1470	1530	**232X**	**7.1X**
Heston European Barrier	38630	690	750	**56X**	**6.4X**

* Productivity measured in time saving over manual rewrite of C++ to hand optimized OpenCL Kernel and Host CPU Code
Platform: AWS F1 Xilinx vu9p

Merlin Compiler C++ FPGA Accelerator Flow – 1 2 3



Merlin Compiler Automated Design Space Exploration

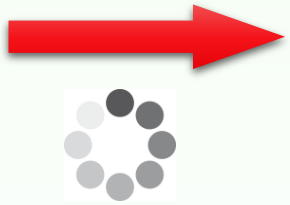


Merlin Compiler Automated Design Space Exploration

```
#pragma ACCEL kernel
void aes256_encrypt_ecb_kernel(
    uint8_t key[32],
    uint8_t data[16*BATCH]) {

 for ( int i = 0; i < BATCH; i++) {
   aes256_encrypt_ecb( key, data + i*16);
}
```

AES kernel (CPU runtime 90.9s)
UltraScale on Amazon F1:
8.4s (**10.8x over CPU**)

Machine learning algorithms
Gradient-based algorithms



```
Starting design space exploration ...
06-05 13:45 [INFO] Maximum DSE time is set to 240 mins
06-05 13:45 [INFO] HLS time for each design point is limited to 15 mins
06-05 13:46 [INFO] Insert 18 design space pragmas
06-05 13:46 [INFO] Create 16 design space factors
06-05 13:46 [INFO] Design space includes 6.42E+10 points
06-05 13:46 [INFO] Profiling the design space using HLS
06-05 13:50 [INFO] Partition design space to 3 partitions
06-05 13:50 [INFO] Exploring 3 design space partitions using 8 threads
06-05 13:50 [INFO] Target platform: xilinx:aws-vu9p-f1:4ddr-xpr-2pr:4.0
06-05 13:50 [INFO] Starting partition 0 DSE for maximum 240 mins
06-05 13:50 [INFO] Starting partition 1 DSE for maximum 239 mins
06-05 13:50 [INFO] Starting partition 2 DSE for maximum 239 mins
06-05 13:51 [INFO] Finished partition 2 DSE in 0 mins
06-05 13:52 [INFO] Finished 1/3 partitions, best: inf
06-05 13:56 [INFO] Finished 1/3 partitions, best: 151171139 cycles from partition 1
06-05 14:06 [INFO] Finished 1/3 partitions, best: 67207242 cycles from partition 1
06-05 14:18 [INFO] Finished 1/3 partitions, best: 25346115 cycles from partition 1
06-05 14:24 [INFO] Finished 1/3 partitions, best: 17010840 cycles from partition 1
06-05 14:41 [INFO] Finished partition 0 DSE in 50 mins
06-05 14:42 [INFO] Finished 2/3 partitions, best: 4261020 cycles from partition 1
...
```

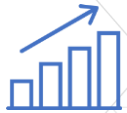
```
#pragma ACCEL kernel
void aes256_encrypt_ecb_kernel(
    uint8_t key[32],
    uint8_t data[16*BATCH]) {
#pragma ACCEL interface variable=data bus_bitwidth=512

#pragma ACCEL tiling factor=4096 parallel factor=4
 for ( int i = 0; i < BATCH; i++) {
#pragma ACCEL pipeline flatten
  aes256_encrypt_ecb( key, data + i*16);
}
```

AES kernel with auto-generated pragmas
UltraScale on Amazon F1:
0.07s (**5347x over CPU**)

“S2FA: An Accelerator Automation Framework for Heterogeneous Computing in Datacenters ”, DAC’18

https://www.falconcomputing.com/

Acceleration simplified across multiple applications regardless of hardware experience

Application	User	CPU to FPGA acceleration	Benefits
	SDAccel Developer		 **Up to 35x** Performance Gain  **Up to 10x** Productivity Gain **Zero FPGA experience required**
	Software Dev / Data Scientist		
	Bioinformatician		**Whole genome analysis in < 6hrs**

Merlin Compiler | Expert Services | Kestrel Runtime | Genomics

Adaptable.
Intelligent.

XDF XILINX DEVELOPER FORUM

XILINX