Технологии QNX и КПДА в России

Москва, 25 апреля 2018

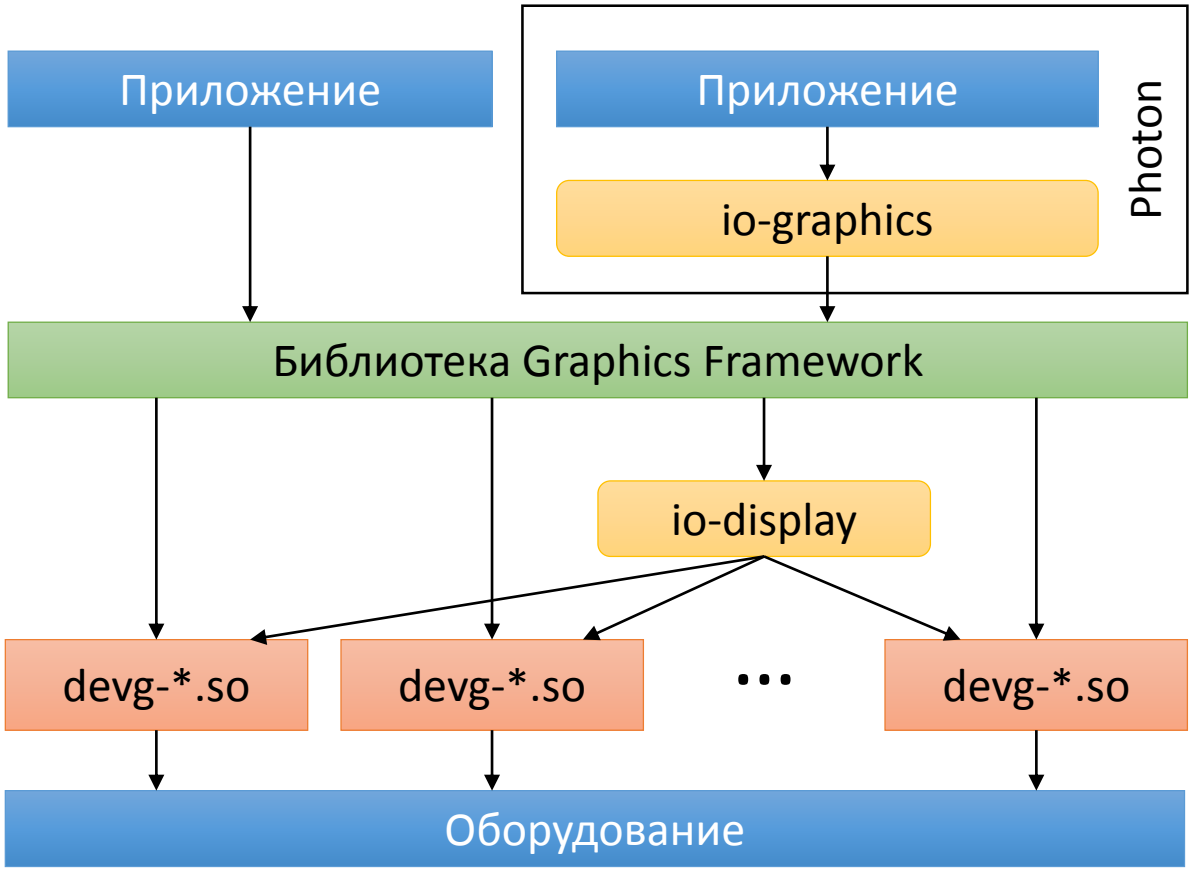
«Практические аспекты использования высоко-производительных графических процессоров в ЗОСРВ «Нейтрино»

Александр Молодцов, СВД Встраиваемые Системы

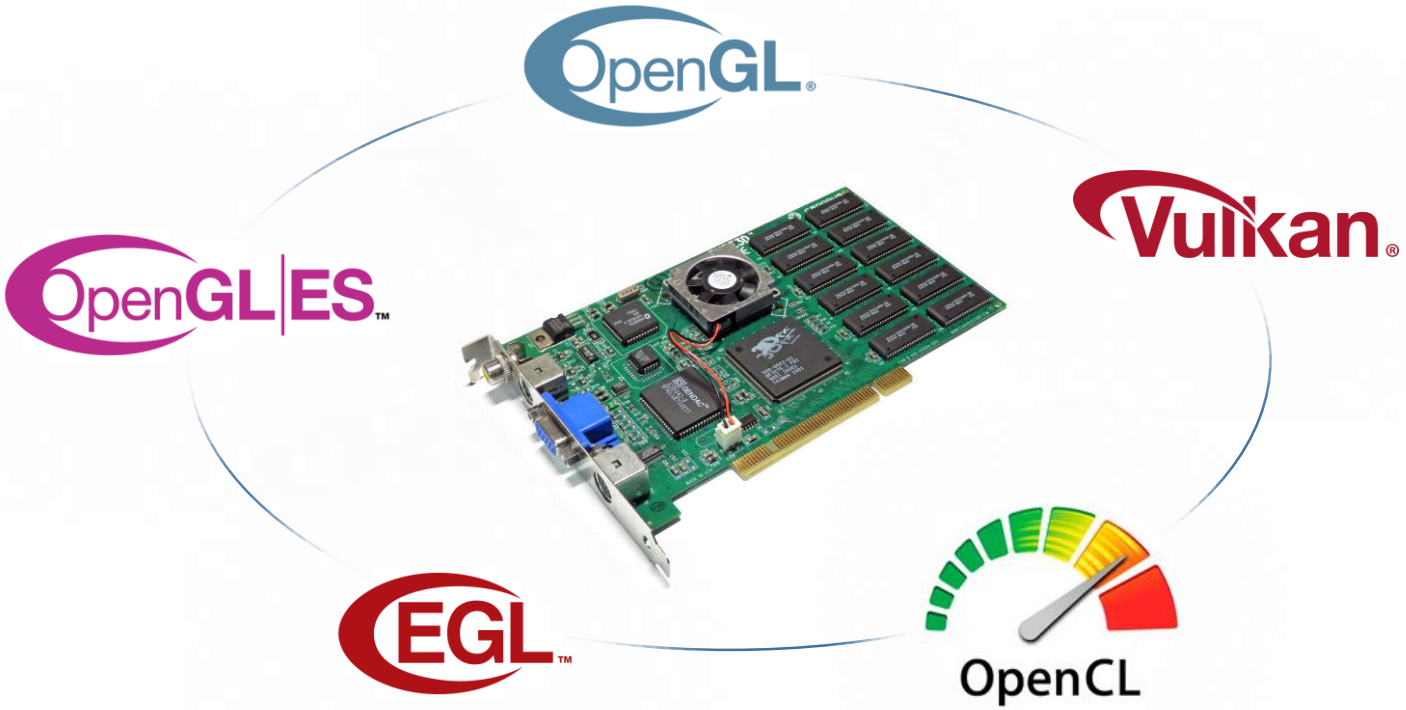
СВД Встраиваемые Системы

SWD Software

Graphics Framework



Интерфейсы для работы с GPU



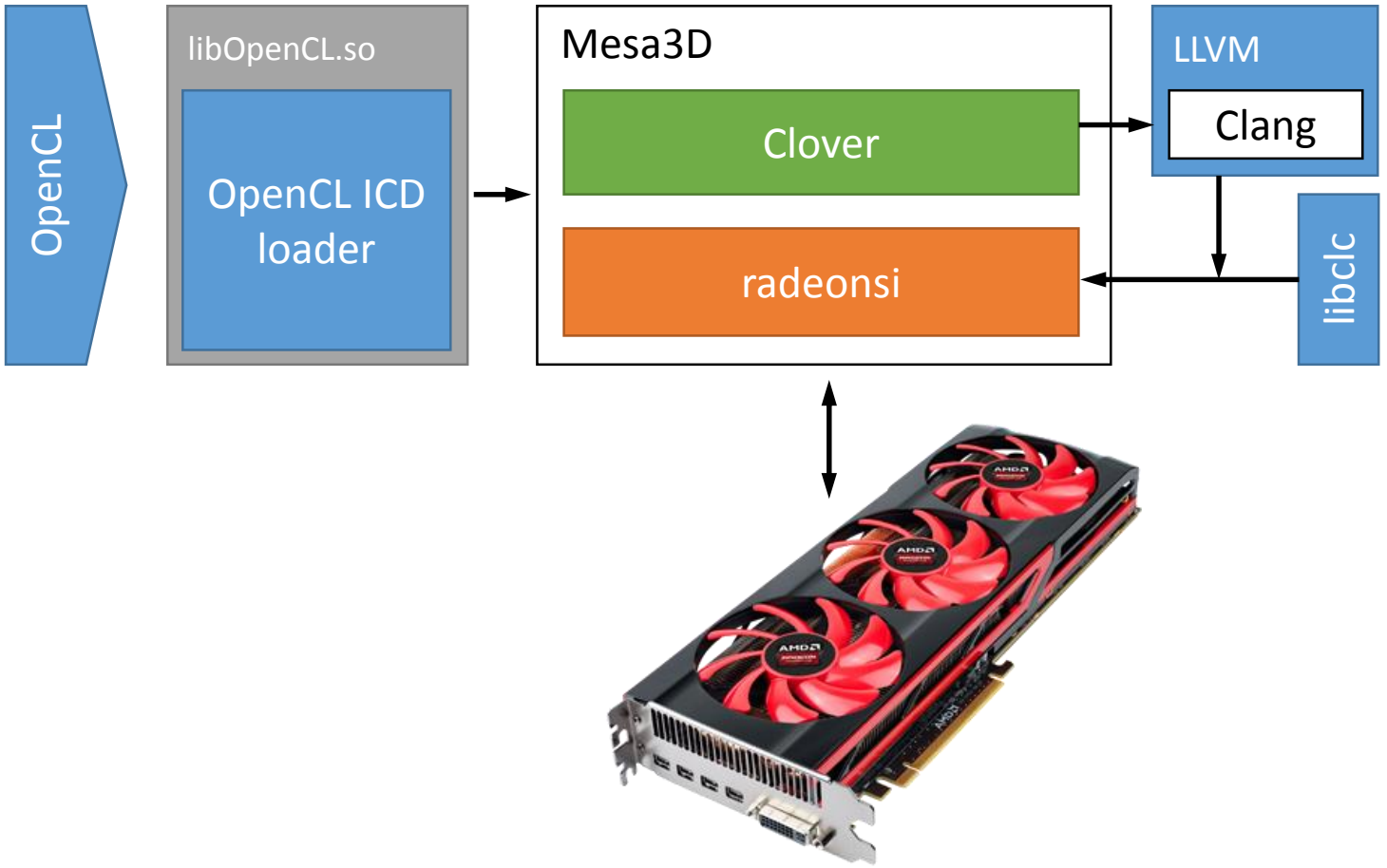
Mesa 3D

- OpenGL 4.5
- OpenGL ES 1.1/2.0
- Vulkan
- GF3D
- EGL 1.4
- OpenCL 1.1

СВД Встраиваемые Системы

SWD Software

OpenCL

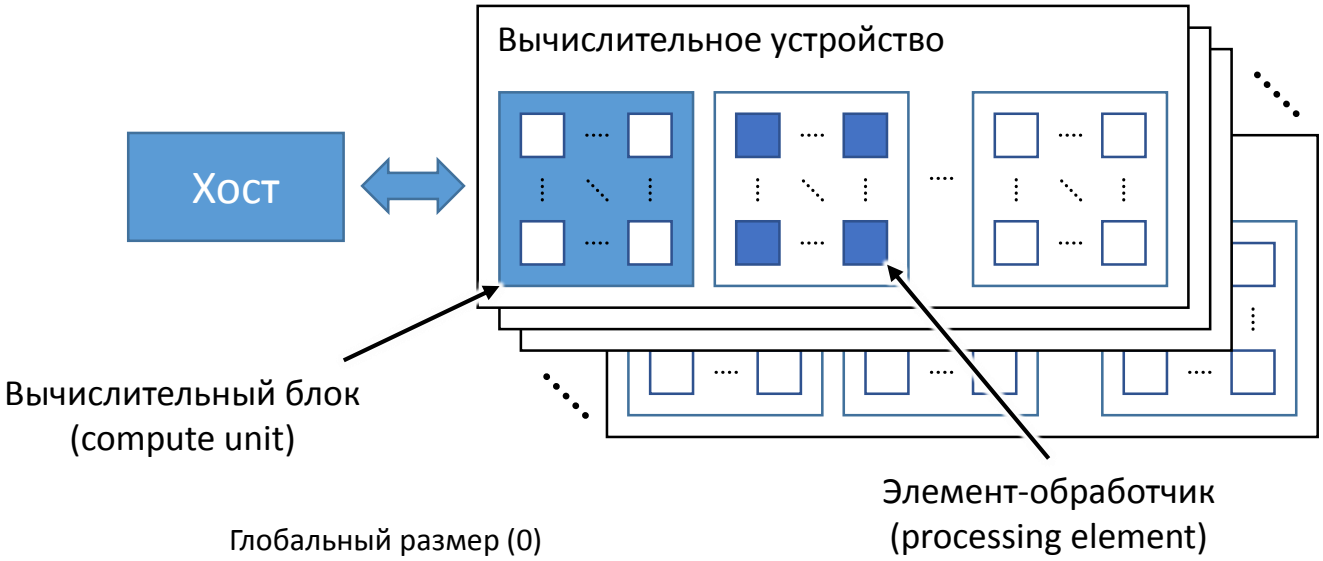

OpenCL
libOpenCL.so
OpenCL ICD loader
Mesa3D
Clover
radeonsi
LLVM
Clang
libclc

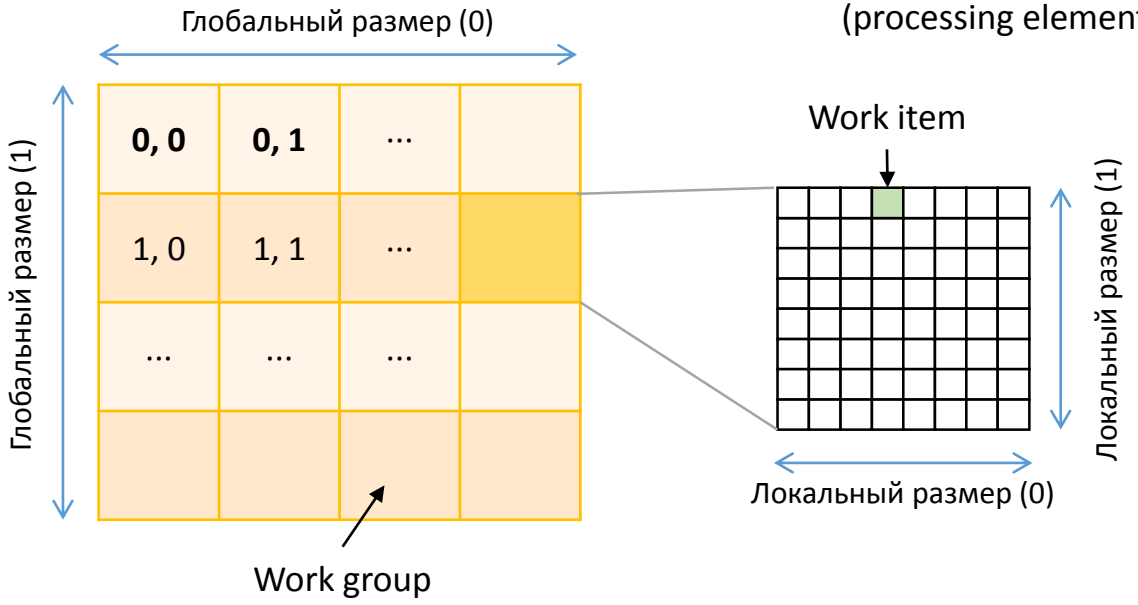
OpenCL kernel

- Kernel – маленькая функция, написанная на OpenCL C, являющимся подмножеством C (C99).
- Kernel является точкой входа для исполняемого устройством кода.

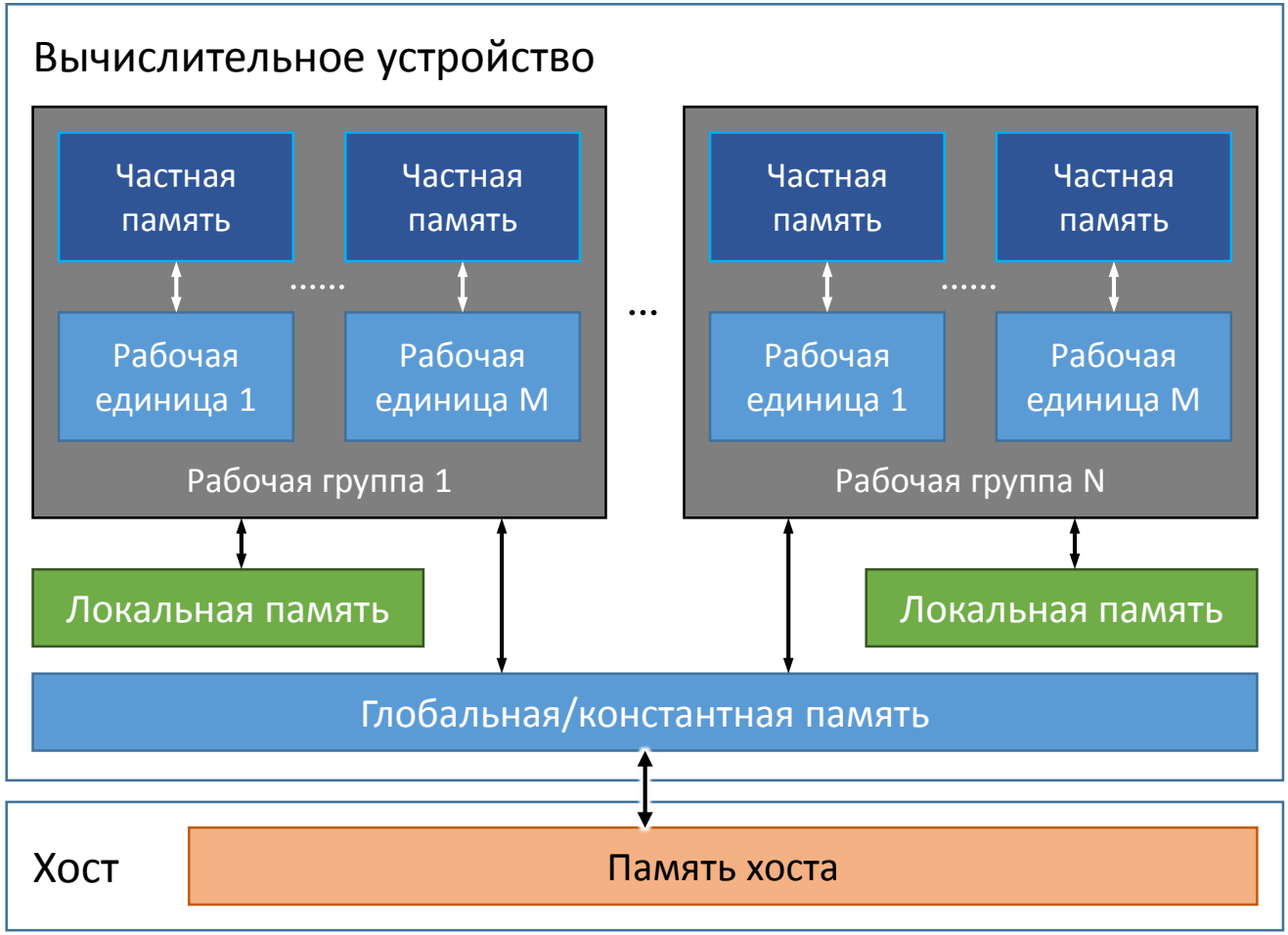
- Отличия от C:
  - Нет указателей на функции
  - Нет рекурсии
  - Имеет векторные типы данных
  - Имеет типы данных для изображений
  - Структуры доступны, но они плохо влияют на производительность и взаимодействие с хостом может оказаться нетривиальным
  - Нет механизма взаимодействия между разными kernel'ами

Модель платформы





Модель памяти



Программная модель

- Параллелизм заданий (task parallel)
- Параллелизм данных (data parallel)

Традиционная модель с циклом:

```
void
mul(const int n,
    const float *a,
    const float *b,
          float *c)
{
   int i;
   for (i = 0; i < n; i++)
      c[i] = a[i] * b[i];
}
```

Модель с параллелизмом данных:

```
__kernel void
mul(
    __global const float *a,
    __global const float *b,
    __global       float *c)
{
   int id;
   id = get_global_id(0);
   c[id] = a[id] * b[id];
}
```

Запуск вычислений - хост

- Хост программа осуществляет:
  - Настройку окружения для OpenCL программ
  - Создание и управление kernel’ами

- Основные этапы работы простой хост-программы:
  1. Подготовка платформы – устройства, контексты, очереди
  2. Создание и постройка программы - динамической библиотеки для kernel’ов
  3. Настройка объектов памяти
  4. Задание аргументов для kernel функций
  5. Отправка команд – перенос данных объектов памяти и запуск kernel’ов

Подготовка платформы

- Получить первую доступную платформу:

```
err = clGetPlatformIDs(1, &firstPlatformId,
      &numPlatforms);
```

- Использовать первое доступное GPU устройство предоставленное платформой:

```
err = clGetDeviceIDs(firstPlatformId,
      CL_DEVICE_TYPE_GPU, 1, &device_id, NULL);
```

- Создать простой контекст для единственного устройства:

```
context = clCreateContext(firstPlatformId, 1,
      &device_id, NULL, NULL, &err);
```

- Создать простую очередь команд для выполнения устройством:

```
commands = clCreateCommandQueue(context, device_id,
      0, &err);
```

Подготовка программы

- Задать исходный код для kernel-программы в строковой константе или считать из файла.

- Создать объект программы:

```
program = clCreateProgramWithSource(context, 1,
      (const char**) &KernelSource, NULL, &err);
```

- Скомпилировать и склинковать программу:

```
err = clBuildProgram(program, 0, NULL,NULL,NULL,NULL);
```

Работа с памятью

- Определить объекты памяти OpenCL:

```
cl_mem d_a = clCreateBuffer(context, CL_MEM_READ_ONLY,
     sizeof(float)*count, NULL, NULL);

cl_mem d_b = clCreateBuffer(context,
     CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
     sizeof(float)*count, h_b, NULL);

cl_mem d_c = clCreateBuffer(context, CL_MEM_WRITE_ONLY,
     sizeof(float)*count, NULL, NULL);
```

Определение kernel

- Создать kernel объект из kernel функции “mul”:

```
kernel = clCreateKernel(program, “mul”, &err);
```

- Задать аргументы kernel функции “mul” соединив их с объектами памяти:

```
err  = clSetKernelArg(kernel, 0, sizeof(cl_mem), &d_a);
err |= clSetKernelArg(kernel, 1, sizeof(cl_mem), &d_b);
err |= clSetKernelArg(kernel, 2, sizeof(cl_mem), &d_c);
```

Работа с очередью

- Записать данные буферов из хоста в глобальную память:

```
err = clEnqueueWriteBuffer(commands, d_a, CL_FALSE, 0,
      sizeof(float)*count, h_a, 0, NULL, NULL);
```

- Запланировать выполнение kernel:

```
err = clEnqueueNDRangeKernel(commands, kernel, 1, NULL,
      &global, &local, 0, NULL, NULL);
```

- Запланировать копирование данные буфера обратно в память хоста по адресу “h_c”:

```
clEnqueueReadBuffer(queue, d_c, CL_TRUE,
      sizeof(float)*count, h_c, NULL, NULL, NULL);
```

- Дождаться выполнения команд:

```
clFinish(commands);
```

Документация

- Спецификация OpenCL 1.1:

https://www.khronos.org/registry/OpenCL/specs/opencl-1.1.pdf

- «Шпаргалка» по OpenCL API 1.1:

https://www.khronos.org/files/opencl-1-1-quick-reference-card.pdf

Спасибо за внимание

Александр Молодцов
Инженер-программист

(812) 346-8956
a.molodtsov@kpda.ru

www.kpda.ru
www.swd.ru

СВД Встраиваемые Системы

SWD Software