# AUTOMATIC LICENSE PLATE RECOGNITION (ALPR)

## THE XMOS SOLUTION

WHITEPAPER

XMOS

# 1. ALPR – AN INTRODUCTION

Automatic License Plate Recognition (ALPR) is becoming increasingly common in and around modern cities. Pay-by-plate tolling, parking garages, and even speed cameras all use license plate numbers to collect fees. Many of these systems rely on cloud computing and/or humans in the loop to accurately identify plate IDs. This is an expensive and energy intensive operation that relies on transmitting large numbers of high-resolution images over the internet.

One of the most visible applications of ALPR systems is in overhead gantries for collecting highway tolls without the need for transponders or stopping at a tollbooth. ALPR is also making its way into car parks, primarily for enforcement of permits and payments.

In the future, more ubiquitous ALPR systems will enable new use-cases, for example car location and parking reservation.

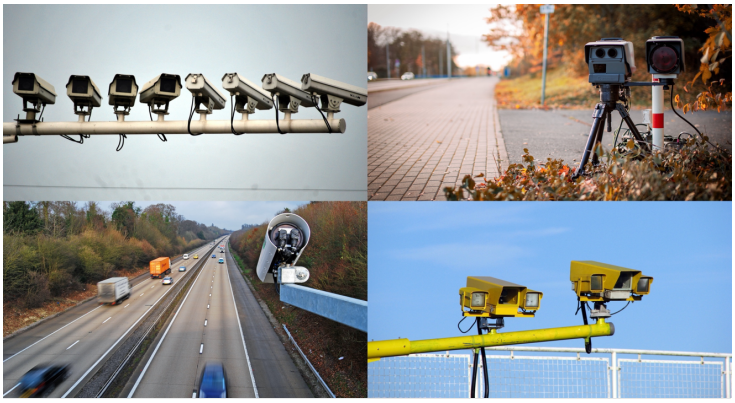
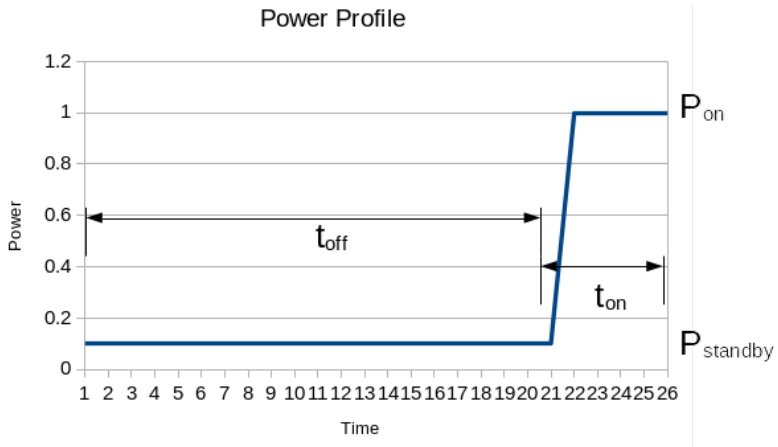


Figure 1: Examples of ALPR systems deployed today

To save on system cost and power, ALPR systems in the future will put the compute as close to the camera as possible and only send relevant data over lower-bandwidth communications links. This will save power on both compute and transmission of data, as well as expanding the possible options for backhaul technologies to include low-bandwidth and low-energy technologies. In many applications, it will also be possible for the compute/camera module to be kept asleep for most of the time. This means that average power will be further reduced by the duty cycle of that system. In a parking situation, for example, it might make sense to check for plates at a rate determined by the minimum billing interval, or an additional sensor with 1 bit of information could wake the system up when a car enters or leaves the space.



## 2. THE CUSTOMER PROBLEM

Due to the high vehicle speeds, distance to the target vehicle, and unpredictable lighting conditions, today's common application of ALPR on highway gantries requires very high-quality, high-resolution cameras that can process many plates per second. In addition to quality optics and camera hardware, these systems rely on large amounts of cloud computation and frequently necessitate human intervention. Cloud processing enables these systems to run smoothly, but they require full video frames to be sent over the network to run inferencing on server hardware. This, in turn, will run less energy efficient code - both from the overhead associated with running a server OS as well as the tendency to be less strict about model optimisations when memory and FLOPS are not constrained.

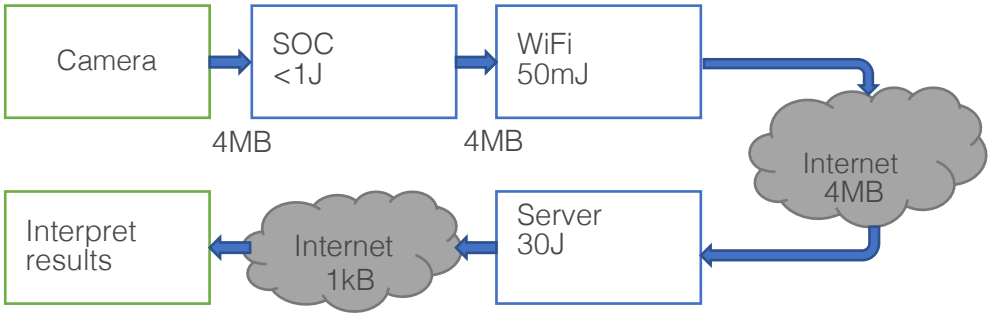
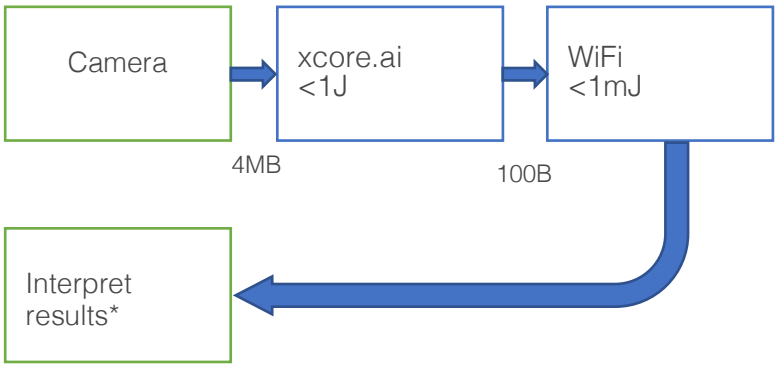



Figure 3: Energy and data requirements for a 100% cloud-based solution using nominal values and assuming a single frame of 1920x1080x16-bit image data.

The massive pipeline from camera to ALPR output might be warranted in this extremely challenging case but scaling it down for lower speed applications such as parking reservation, billing, and enforcement, will result in a lot of wasted power and bandwidth. The figure below shows where operational savings can be achieved by switching to local compute:




* Optionally store whole images for enforcement purposes

Figure 4: Diagram showing a device-based solution with much lower energy and data usage than the cloud-based solution in Figure 3

## 3. IMPLEMENTING A SOLUTION

Most published solutions to the ALPR problem involve one or more neural networks that perform three very distinct tasks:

1. Locate the plate in an image and generate a sub-image
2. Segment that characters in the plate
3. Perform optical character recognition (OCR)

Each inferencing task can also have a receiver operating characteristic (ROC) that can be tuned to reduce power consumption, for example: confirm that the license plate is the same character string in the same location before transmitting anything off-chip. Breaking up the problem in this way allows system designers to choose different tools for each part and trade off accuracy, speed, size, and ultimately cost. For example, the plate localisation algorithm will be highly dependent on the camera being used, which will be a big driver of system cost and model size. The character recognition model, on the other hand, will have a minimum number of pixels that a character must occupy. Working backwards from this requirement, with additional parameters such as plate size and expected distances, we can determine the lowest resolution camera that will still meet a given threshold of accuracy. For example, if our OCR network needs letters that are 10 pixels tall, and we need to be able to recognize a plate with 10cm tall letters, at a range of 6 meters with a FOV of 30 degrees:

Height at 6 meters: 2 x 6m * tan(30/2 degrees) = 3.2m

Height / letter height = 3.2m/0.1m = 32 letters high

With a minimum of 10 pixels per letter this sensor would need to be at least 320 pixels high.

Low frame rate imaging applications are ideally suited for XMOS' xcore.ai platform. Local inferencing on xcore.ai is less resource intensive than idling and forwarding data from a larger chip running an embedded operating system. xcore.ai performs the ALPR processing very close to the camera which greatly reduces the number of devices that need to be able to pass along high-resolution images, reducing the overall system BOM. Some example applications are: visual wake word, feature detection, and ALPR. In addition to the AI acceleration offered by the 256-bit wide vector processing unit (VPU) the xcore.ai chip has the flexibility to define, in software, the following parameters of an ALPR solution:

- SPI / MIPI / other camera interface
- I2C or USB command and control
- Backhaul via USB / I2C / UART / WiFi / etc.
- 1 or more machine learning models
- TensorFlow lite for microcontrollers or custom VPU accelerated code
- OpenCV-like pre/post processing library functions
- Software image signal processing (ISP)

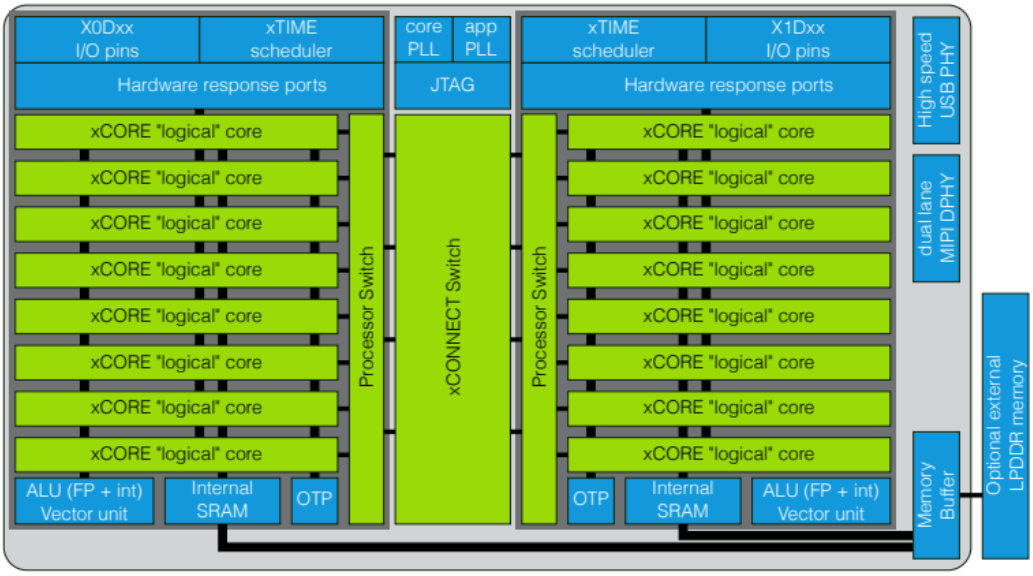



Figure 5: xcore.ai block diagram

In addition to being a low-power and cost-effective platform, xcore.ai allows system designers to use the same chip and core machine learning algorithms in multiple devices and configurations.

Most models that can be translated into a TensorFlow lite int8 quantized representation can be automatically transformed to run on xcore.ai using our xcore-opt tool. The table below shows the trade-offs between internal RAM only and external RAM based designs:

| FEATURE | SRAM + FLASH | EXTERNAL RAM |
|---|---|---|
| Models with up to 3 million weights | yes | yes |
| Working memory requirements > 350kB* | no | yes |
| Least expensive package | yes | no |
| Lowest power | yes | no |

* Working memory may be traded off for execution speed in xcore-opt

The XMOS solution is ideal for parking applications that require higher resolution cameras or several video frames to compensate for lighting conditions, vehicle position, or motion effects. We can rapidly capture video over MIPI and store several frames in off-chip RAM (LPDDR is natively supported with our FB265 package and the xcore architecture makes integrating other RAM technologies easy with smaller packages). The xcore can then perform pre-processing and inference on these frames over the course of several seconds, after the vehicle has parked/departed.

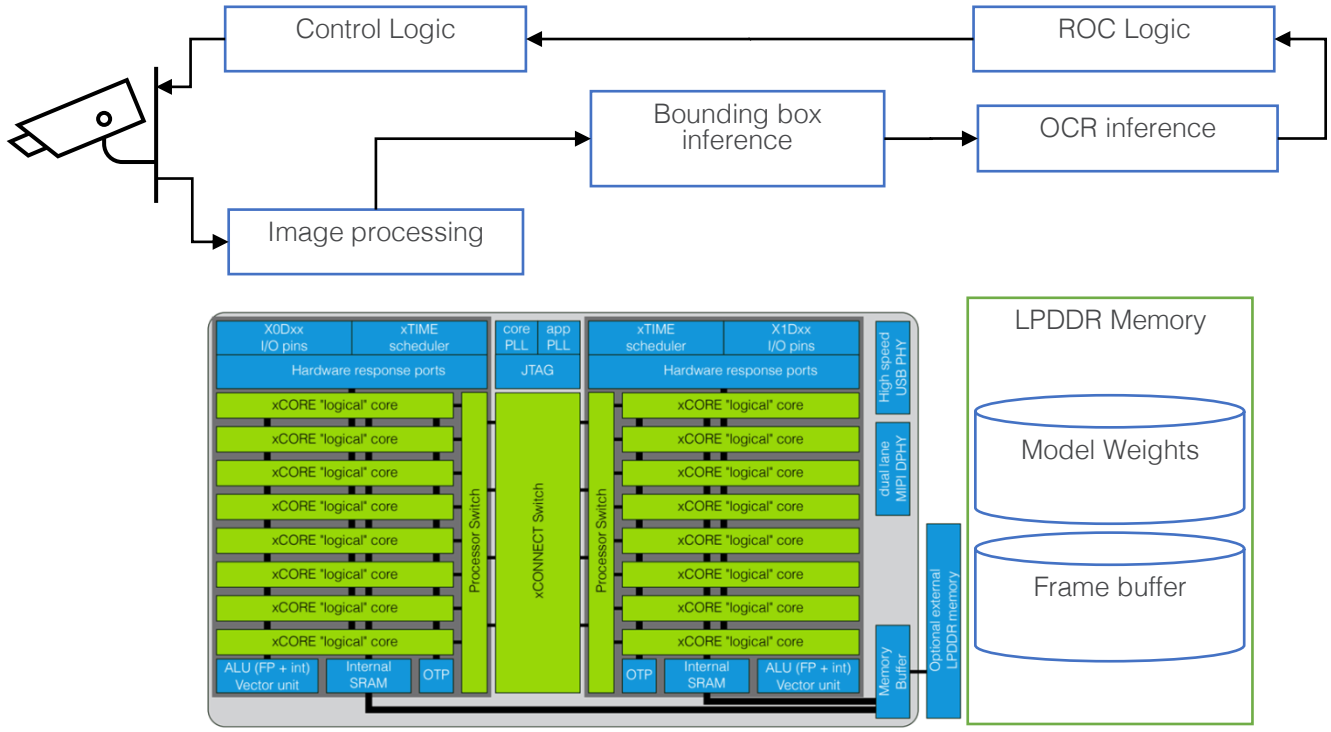



Figure 6: Block diagram showing a possible architecture for a high resolution (~1920x1080) ALPR solution on xcore.ai.

This approach highlights the flexibility of the xcore as part of an overall system. In this case, the xcore is woken up by an external sensor (radar/lidar/etc.) when it is time to buffer video frames. Once the frames are captured, the processing begins, with an emphasis on power savings and latencies that are acceptable for the application. In the case of parking, the acceptable latency could be as high as a minute, provided the camera is able to capture new frames when receiving an interrupt. This scheme allows for the removal of an expensive, power hungry SoC – unsuitable for an outdoor solar + battery application.

## 4. REAL WORLD RESULTS

Our partner, Cloudtop, has adapted a solution that was originally a cloud-based model to fit entirely on an xcore.ai chip, without the need for external LPDDR. The savings in model size came from reducing the problem scope to a single, stationary vehicle and a lower resolution camera. XMOS tools also enable model weights to be stored in Flash memory, reducing the total working memory requirements for inferencing on large models.

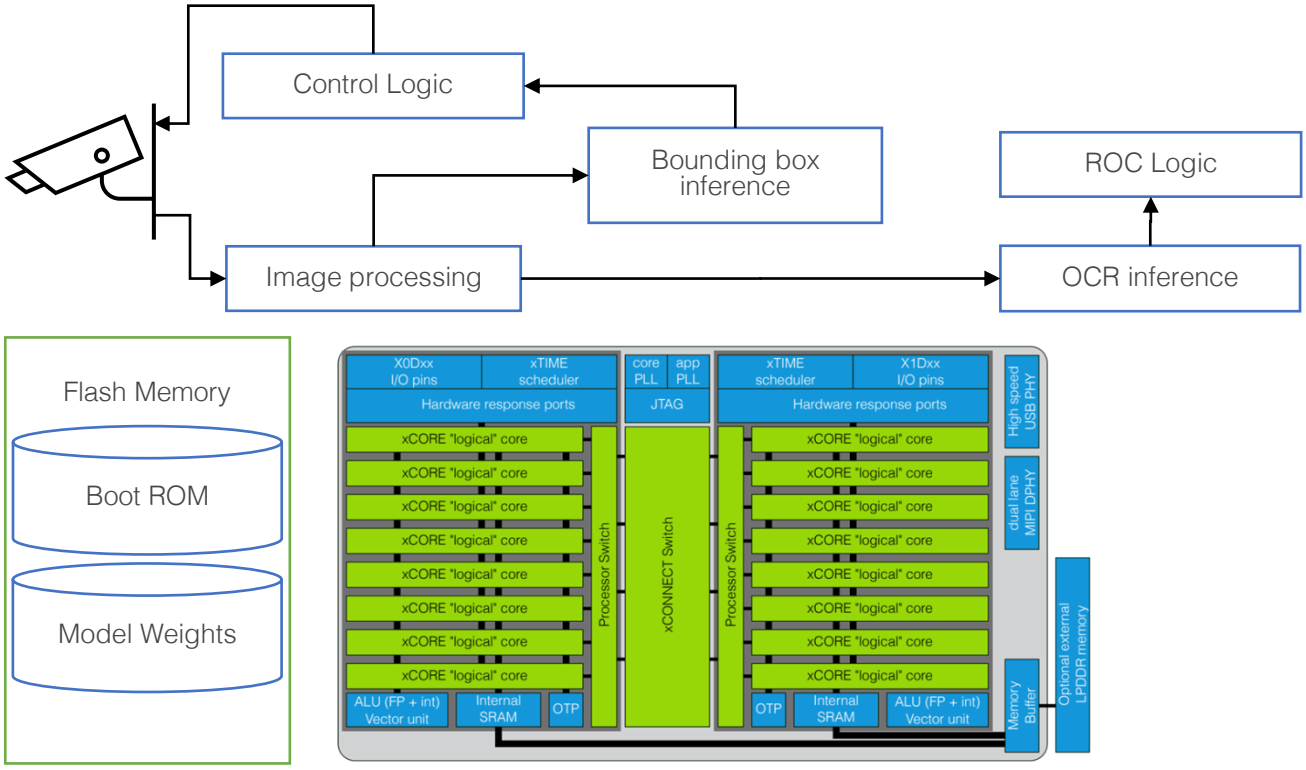



Figure 7: Block diagram showing a possible architecture for a low resolution (~640x480) ALPR solution on xcore.ai.

In this case a 1600x1200 camera sensor was chosen to avoid the increase in mechanical costs associated with using a narrow FOV lens. The image is cropped in real-time so that a small region of interest (~640x480) can be stored in on-chip SRAM for processing. The single shot detector (SSD) network looks at this image to determine the location of the license plate. Once that location is determined, the xcore.ai requests another frame from the camera and only stores the region where the plate was detected. This smaller image is then fed into a second network that performs OCR to get the character string that describes the license plate. Since this is for a parking application, the plate can be assumed to be stationary between these two captures and we use this knowledge to further reduce our need for memory.

This approach lowers the cost significantly, by removing the need for off-chip RAM. Depending on the camera interface requirements, the cost of the xcore.ai can be further reduced by selecting other packages from the xcore.ai range. The low latency of a solution that fits entirely in internal memory also saves power by reducing the amount of time the chip needs to be awake. The dual network, modular approach means that heuristics can be derived from the first stage to determine if we need to take a second picture and do OCR, for instance determining if the plate bounding box is moving.

This approach may be adapted to fast moving vehicles with a narrow FOV camera and combined detection/recognition network. This is for further study.

# 5. CONCLUSION

ALPR challenges can vary significantly depending on the expected lighting conditions, view angles, number of plates are being recognised by each camera, and the speed of the vehicles. An xcore.ai-based design can be configured to trade off speed, cost, external memory requirements, and accuracy by relocating image buffers, re-ordering processing, and being able to execute neural networks from SRAM, Flash, or external LPDDR/SDRAM.

The xcore.ai chip bridges the gap between expensive, heavy weight SOCs and traditional microcontrollers. Unlike AI accelerators, the flexibility of xcore enables us to combine all the classes of compute required to build complete systems, including control, DSP, AI, and I/O for a multitude of intelligent IoT applications. This enables the most cost-effective systems to be built rapidly, using software alone.

If you would like to discuss your ALPR requirements and find out more about the XMOS solution, please contact at sales@xmos.com.