rSmart – The Reconfigurable (Real) Smartphone

Smartphones today are really not that smart. They are designed to satisfy a large population's varying requirements in general and are not tuned for individual customer needs. Several works have shown that this is not really ideal and the industry keeps ignoring these results and customer calls. They are also ignoring the revenue gains when a single smartphone can do the best video processing, graphics and do decoding/encoding at the fastest speed enhancing response times and user experience.

What is missing here? The current "smartphone" has a lot of hard-IPs which are individually designed to do a specific activity but really have no use (other than dissipating power) if the running app has no use for them. For example, if a particular user watches a lot of Youtube, the camera IP has no use for him. Another usecase is when someone is travelling and uses GPS, which consumes power and the rest of the active platform only dissipates power without any real use

rSmart

rSmart is different from conventional "smartphones" in that it is built on FPGA. Reconfigurable FPGAs offer a lot of flexibility to meet functionality requirements since they can be reprogrammed in the *field* – after the user has the smartphone. FPGAs have gained quite a lot of ground in terms of matching ASIC performance and hence are not a big compromise in terms of performance. By reconfiguring parts we save on power and gain a lot on performance which is every engineer's dream. Further from the validation perspective, a lot of post-silicon time is saved as any hardware issues can be resolved though a patch. In fact, rSmart can bring in a whole new set of apps targeting the underlying hardware and these offering different algorithms for implementing the functionality.

Two ways to build rSmart

1. Allow the user to specify the IPs of interest. These will the specific IPs the user plans to use typically and these cannot be reconfigured. The rest of the IPs can be reconfigured

2. Auto-detect based on the specific app that the user is running and reconfigure the apps accordingly

We will discuss both approaches and infrastructure requirements for both

User-guided reconfiguration:

Typically users have well set pattern of usage for their phones – when they use and what they use their phones for. So they can guide rSmart into deciding which IPs should be active and which need not be. For example, a user doing a lot of browsing can check the CPU, display, memory and WiFi to make sure these are not reconfigured while the other IPs can be flexibly reconfigured by the underlying infrastructure. This method simplifies a lot of the monitoring infrastructure that the self-reconfiguring logic needs

Self-reconfiguration:

In self-reconfiguration, rSmart includes additional logic to enable the reconfiguration. This includes the *ActivityMonitor* track the activity of the current app the user is running. Typically this includes monitoring activity on the individual IPs. Depending on how frequent the IP is being used, the monitoring logic would identify the IP as a candidate for reconfiguring

Additional logic required

1. IP stubs: Each IP includes a stub which is just an interface to the minimal features that the IP provides so that the application would not crash. Typically, this should not include any functionality but just make sure the accesses go through to access internal registers. IP stubs help provide information if a stub is being accessed too often.

2. Reconfiguration manager: This maintains the set of IPs that can be reconfigured (input from user or ActivityMonitor and IP stubs) and decides the IP that is required based on this information

Software Impact

Given that the underlying hardware changes, we need the software to be "smart" as well. Sometimes this could be transparent and the hardware takes care but it is more efficient if the software is also knowledgeable. The rSmart driver would interface with the reconfiguration manage and understand the underlying changes. For example, if we have a gaming app (AngryBirds), the CPU and graphics IP would be heavily involved. Assuming the camera IP is getting reconfigured, the rSmart driver would identify independent threads that the graphics IP handles and load balances between across the 2 IPs (Refer Figure 1).

Summary: Pros and Cons of rSmart

+ rSmart allows smartphones to be tuned to user needs, thereby providing great performance and at lower power. With increasing requirements to meet individual user needs, this is a great step forward towards meeting these

- rSmart involves FPGAs and hence performance might not be the best in class but this is not a fundamental obstacle and once the hardware engineers focus on this, the performance gap should disappear.

- rSmart requires additional space for storing the programming logic. Given the integration levels we are looking at in future, this is again not a fundamental obstacle

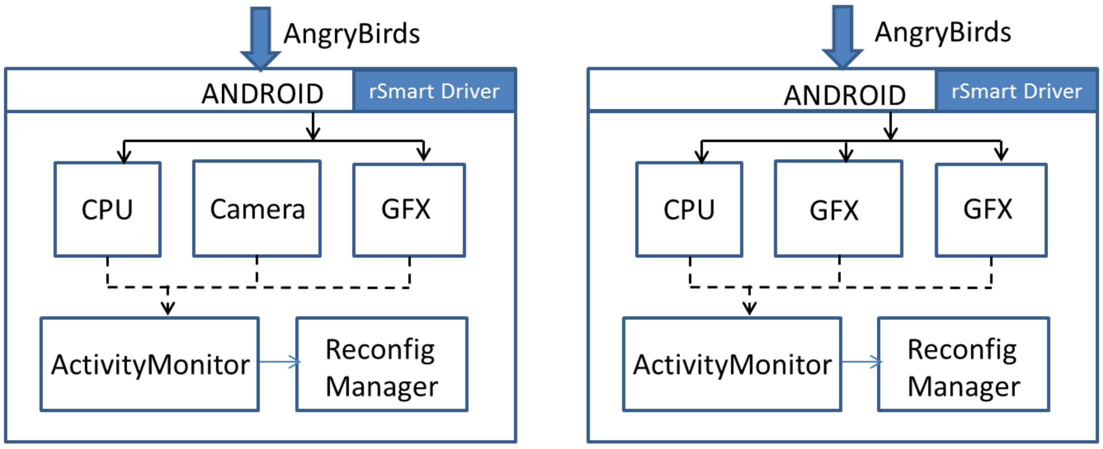

Figure 1: AngryBirds on rSmart