

# PCB Design Challenges: Designing With DDR

Feature Interview The I-Connect007  
Editorial Team

Longtime signal integrity experts Rick Hartley and Barry Olney join the I-Connect007 Editorial Team for a discussion around DDR and the complications board designers inevitably face when they design for DDR. If, as Rick and Barry explain, the DDR design process is not that much more complicated than that of a typical high-speed board, why does DDR cause design engineers so much grief? Much of this comes down to following the process, running simulation, and not relying on reference designs.

**Andy Shaughnessy:** I thought maybe one of you could explain why DDR is used so often. What is the advantage of using DDR?

**Barry Olney:** It's the fastest technology we have for memory. With each new version there is a dramatic increase in device data rates and bandwidth; for instance, DDR4 has a maximum data rate of 3200 MT/s whereas DDR5 peaks at 6400 MT/s.

**Rick Hartley:** I have not used DDR5, but it's not just double data right now; it's quadruple data.

**Olney:** Yes, DDR5 has only recently come out. A few of the chip manufacturers in Taiwan have it, but I have not seen it myself yet, so I'm looking forward to it.

**Shaughnessy:** What does this look like? What does this entail? When someone says, "We've got a design for this DDR," what are we talking about? Is this a physical thing?

**Hartley:** Like any bus technology it has data, address, clocks, control lines, it has all the things that most bus technologies have. It's source-synchronous clocking, meaning that the clock is launched with signals so you're not launching a clock and then depending on the signals to arrive at the receiver within a given timeframe. You launch all of them together and if all the data, for example, is set up within a certain timeframe relative to the clock/strobe, everything functions as it should. And the same is true for the clocks of the address. It is source-synchronous, which makes it a much, I



Barry Olney

think, easier technology to use. What are your thoughts, Barry?

**Olney:** It is easier to use. Especially with the DDR3, DDR4, and DDR5, now you have the write leveling which synchronizes the clocks to the address, command, and control lines. You must realize that with DDR, there are two sets of clocks. There is the main clock of the two, and there are also the strobes that trigger the data capture. Basically, the clock and strobe must be at the longest delay of all signals because the address and data must settle before the data is captured.

**Hartley:** The key is to route the strobe to the longest so that it has the greatest delay and arrives last.

**Olney:** Exactly, and that's what a lot of people don't do. They have the strobe arriving first and then the data last, and you get all the reflections and noise. You don't know whether it's going to capture the correct data.

**Hartley:** The first time I did a DDR2 design was at L3 in 2012, and the next one we did was also DDR2 in 2014. One of our engineers did a timing analysis of the design in 2014. He came up with a number that nobody believed. Everybody in engineering said, "Oh, this can't be right, because all the app notes say that DDR timing is so critical." We came up with a number that said, for example, that the address line only needed to be  $\pm 125$  mils from some optimal length, which is basically  $\pm 3$  mm. Everybody said, "This can't be right." Yet shortly after that, I ran into a note that I found from Keysight Technology, written by an engineer named Chang Fei Yee, who wrote:

*"Maximum DDR2 skew of trace length to meet timing margin. In order to be compliant with the JEDEC specification, the maximum skew among all signals shall be less than  $\pm 2.5\%$  of the clock period drew in by the memory controller. All signals of the SDRAM are directly or indirectly referenced to the clock, for example in normal FR-4 material with a dielectric constant of approximately four, a differential clock-rate of 1.2GHz, the maximum skew shall be  $\pm 125$  mils, or  $\pm 3$  mm."*

This is precisely the number we came up with, even though half the engineers in our department said it couldn't be right. I intentionally mismatched the length of some of these lines just to keep them within that quarter of an inch distance from one another, and the thing worked perfectly. That was when I first realized how noncritical this really is.

**Olney:** I have a table on that, Rick. For 166 MHz, the interconnect margin is 155 picoseconds.

**Hartley:** Yes, which is huge!

**Olney:** That relates to about 75 mils.

**Hartley:** I know, it's crazy. And people route these things within  $\pm 5$  mils or  $\pm 0.1$  mm.

**Olney:** I guess I'm one of those crazy people because I'm a PCB designer, and the way I see it is everything has to be done properly. Whether it's a really fast DDR4 or a slow 200 MHz DDR, I route them all the same. I route them all to 10 picoseconds delay and it doesn't take any extra time to do that. People say, "Why should I have it so accurate?" Well, if you just do it out of habit every time then your design works perfectly. If it is 125 out, it might work on the test bench, but if you put it in the field and temperature cycle it, it will fall over. To make it reliable and performing efficiently, you need to have it spot on, and it takes little extra effort to do it that way.

**Hartley:** That is exactly what we did on the result. We intentionally misrouted a few of them in the prototype because I just wanted to see how it would function. We straightened it up in the final design, but we didn't go to these crazy lengths to make things ridiculously tight the way a lot of people do; we just routed them. We put serpentine in a couple of them to get them all within a reasonable skew margin and they worked perfectly. They worked perfectly in the field—and this is avionics, where it had to go through tremendous temperature cycling. There are many of these nav systems flying around in the world today, and the last I heard they were all working perfectly.

**Barry Matties:** Did you do any simulation on those, Rick?

**Hartley:** After the timing analysis we did simulation to verify that the timing analysis was right, and it said that it was right also. That was kind of a double blessing, and as Barry pointed out, DDR3 is even more forgiving, because it balances the lines. With DDR2, you had to do some amount of branch routing to get things to be timed correctly. With DD3, you don't have to do that. You simply route things in daisy-chain fashion, and it sends out a test pulse to



Rick Hartley

figure out what the line links are, and then calibrates everything to be at the right time. There's a more sophisticated way to say that, but you get the point.

**Olney:** Also a lot of people think, Rick, that you can't use fly-by routing with DDR2. That's not correct, I have done it before, it just needs you to hard code the clock delays to each chip from the strobe. So, the data is captured at exactly the right time as the clock passes on the fly-by. If you hard code them manually you'll be doing the same thing as your auto-leveling would do in DDR3c.

**Hartley:** Yes, DDR3 is automatic. That's interesting. I did know that was possible with DDR2.

**Olney:** Fly-by routing is a lot easier to route than T-topology because you have a lot more room to route around the outside of the chips instead of going straight through them point to point.

**Hartley:** I agree, no doubt about it. With the first DDR3 design we did, I realized we could do fly-by routing. What a blessing; that made life so much easier. And I've never done DDR4, so I don't know this for sure, but doesn't it signal the receiver that certain bytes upon receive have to be launched high?

---

**With the first DDR3 design we did, I realized we could do fly-by routing. What a blessing; that made life so much easier.**

---

**Olney:** I don't really get into that side of it, that's more the firmware of the chips, but I believe it does do that.

**Hartley:** The advantage is that you don't have a drain on the power bus. The instantaneous  $L(di/dt)$  drops in the power bus can lead to switching noise. If you're not switching as many lines at once from the driver, this won't happen.

**Olney:** Exactly, that was the idea of fly-by to start with so that all the signals weren't clocked at the same time. There's not as much simultaneous switching noise.

**Shaughnessy:** We get a lot of views on any article with DDR in the title. In our surveys with designers, they say they're having a hell of a time with DDR. You both have said in the past that DDR isn't really that difficult if you just follow the process, so why do so many designers and engineers have trouble with it?

**Olney:** I think it's uncertainty. They're not sure if it will work and they don't have simulation tools to confirm it, so it's up in the air. They hope it works, but they're not sure.

**Hartley:** That's a very good point. I think you hit the nail on the head. Lee Ritchey has a term called "design by fear." A lot of people aren't sure, so they take the worst possible case and figure, "I had better do this just to make sure it's going to work."

**Olney:** The main issue that I see with DDR routing is that people don't choose the right impedance. DDR3 needs to be routed to 40 ohms single-ended and 80 ohms differential impedance. A lot of people still route to 50/100 ohms single-ended/differential. They just can't get in their heads that it's 40 ohms and 80 ohms. That's all to match the drivers. The drivers are normally 34 ohms. They have the wrong impedance to start with, which gives them more reflections. If you are routing to 50/100 ohms, you are going to get more reflections coming back. Data lanes must be routed on the same layer, generally the stripline layer. All the data lanes can be routed on one embedded stripline layer if the pinout of the FPGA is correct. The pin assignments should facilitate routing. You can order them so that the routing can just flow out and into the memory chips then the data with the strobe can be routed in each byte line out to the memory chips on one layer.

Then, the fly-by—the address—can go on that same layer because the address lines are on the other side of the memory chips away from the FPGA. You need to break out from the FPGA with two layers of the address and clock and take it around to where the fly-by traces come through; then they can branch down to the same layer as the data. You only need two layers to route it on.

**Matties:** This might be kind of a silly question, but isn't all this available on Google? I Googled DDR and the first thing that came up was that the clock needs to be longer. It needs to arrive after the data.

**Olney:** Google is smart but not always right.

**Hartley:** There's so much on Google that people don't know what to believe.

**Shaughnessy:** I read an article that you wrote six years ago, Barry, and you explained that DDR specs can be downloaded from JEDEC.org. If you want the real specs, go to JEDEC.org and get them. Why don't people just do that?

**Hartley:** DDR3 is the JEDEC-793B standard, so just download and read it and you'll know exactly what to do. Right, Barry?

**Olney:** That's right. But there is a lot of data to go through. I should also point out that Andy and I were talking about datasheets always being wrong the last time we spoke, and it's the same with DDR specs. If you go to Micron, they have one spec, one timing analysis; JEDEC has another, and Xilinx has another. They're all different. There's not one spec that's the same.

**Hartley:** Right. That's a good point. Here's a comedic story about that: The first time I ever did DDR2 (I won't say whose device we were using as the controller), in the app note section for DDR2 for that device, they actually said, "We had such a hard time getting this thing to work that if you don't design it exactly as we did, we won't guarantee it will work." They said, "We'll give you our Gerber files, and if you're a Cadence user, we give you the design file, and if you don't copy it exactly, we make no guarantees."

And I thought, "Who are these people?" The lead engineer called me up and said, "Rick, did you read this? This is terrifying; are you going to do that?" Knowing he was the nervous type, I said, "We will take care of it, don't worry about it. Everything's good." I didn't do anything they said in the app note, and it all worked perfectly.

**Olney:** Reference designs are always bad designs.

**Hartley:** Almost always. I don't think I've ever seen a reference design that was really good. I really don't.

**Olney:** I've seen DDR2 routed on four-layer boards in reference designs, and how they get it to work I'll never know.

**Matties:** It sounds like it's not that complicated if you do your homework.

**Olney:** If you follow a methodology, you can't get it wrong, really.

**Shaughnessy:** So, designing DDR involves a few slightly different steps than a typical high-speed board?

**Olney:** Each technology has different constraints, so you want to keep the technologies apart. If you have a 5-volt or 3-volt technology next to a memory that's 1.5, then you're going to get more crosstalk because it's a higher voltage; there's separation. You have technology rules, and you have the rules that separate the technologies.

---

**You have technology rules, and you have the rules that separate the technologies.**

---

**Hartley:** I've told people for years that whenever I would do a circuit board with multiple powers driving different technologies—let's say 1.5, 2.5, 3.3, and 5 volts—I make sure that I place the parts, as much as possible, so that they each have their own power distribution sections, for several reasons. One, you have fewer power planes because you can divide the plane up into multiple pieces; second, you're not routing 1.5-volt or 1.8-volt lines next to

3.3-volt lines, which will severely increase crosstalk into the lower voltage line.

**Olney:** The last design I did had 35 different power supplies.

**Hartley:** Now that becomes tougher.

**Olney:** How do you put 35 different copper pours in? The main processor BGA has four or five different supplies that must connect to other circuitry; that blocks your routing channels and makes it really challenging: “We’ll fit them in here or there on plane or signal layers.” But when you start routing it, pushing traces around and trying to get around these copper pours, it’s very difficult.

**Hartley:** It is very challenging, I agree.

**Olney:** One of the nice things about DDR is that it’s source-synchronous, and so you can route it to a 4-mil trace with 4-mil spacing. You don’t have to worry about crosstalk because the data and address arrive at the memory chips and it has time to settle before the clock and strobes come along and capture that data. Providing the clock is a little bit longer, it all works without crosstalk and reflections. You just don’t want to get the data and address mixed up.

**Shaughnessy:** Barry, in your columns you talk about the benefits of using a router for DDR3 and that designers typically don’t like auto-routers, but you said they can be very useful here and you can drive the router with the schematic.

**Olney:** Absolutely. If you set up the constraints properly in the schematic then you can drive the router from the schematic, which I do. First, when you select a chip—maybe from the main processor or a part of that, mainly just the memory bus—you fan out from that, generally from the top layer to internal layers so you don’t have to route on the external microstrip

layer, which can radiate a lot of EM; it’s also faster speed so you don’t have to worry about the difference in propagation delay, between different layers. Stripline traces also have less crosstalk which means you can route in closer proximity. I fan out first, and then I’ll select the first data bus going to the first memory; I will route that with the router, so I route it bit by bit and as I do it, I clean it up. If it doesn’t do it quite perfectly, I can push and shove it around and get the signals almost right.

Once you’ve basically routed it and you’ve made sure you’ve got a little bit of room for serpentine, then you can apply the fine tuning, the auto-tuning of the routes. Cadence and Altium can route to delay, so you don’t route to length anymore. Length and delay are two totally different things, depending on which layer you’re on, so if you’re routing to delay, you’re getting the exact numbers, the exact flight time, to every chip in comparative flight time.

**Shaughnessy:** Do designers run into trouble with the other serial links like PCI Express? Or is that just a different animal?

**Olney:** Generally, PCI Express goes to a connector, so they’re spread out to start with and you just follow the pattern and basically route them in order. You can have them tightly together, and you must in most cases.

**Hartley:** It is a different animal, and it’s like ethernet or any of those things; they generally go to a connector.

**Shaughnessy:** But DDR can be employed anywhere. That’s the beauty of it, I guess.

**Olney:** It can also go to a connector. You may have onboard memory, or if you could have SODIMMs, for instance. When you’re routing to SODIMMs, it’s quite a different strategy than routing to onboard memory. In onboard memory, you generally don’t have series terminators on your data. With plugin memory,

then you need the series terminators to prevent reflections on the long data lines.

**Matties:** Over the years, expert designers have told us that with high-speed design, there really is no substitute for experience. How does a young design engineer become an expert at DDR?

**Hartley:** They make a lot of mistakes. I certainly did early on.

**Olney:** My wife says to me when I'm writing my articles, "Why are you giving all of your secrets away?" I say, "Because nobody can remember them. I can't remember them." (Laughs) You've got to try to recall it all.

**Hartley:** My wife asked me that about the classes I do, "Why are you telling everybody all the stuff you know?" I told her, "Well, I'm not going to live forever, for one thing, and I want the world to get as much of this as we can give them."

**Matties:** Is there a role for EDA tools that helps in this regard? When does the tool come in and really help take the heavy load?

---

**You can't just hand someone a simulation tool and expect them to simulate a board. They have to know how it all works and understand it.**

---

**Olney:** First, you need to understand the technology behind it. You can't just hand someone a simulation tool and expect them to simulate a board. They have to know how it all works and understand it. It is the same with routing,

placement, and positioning power supplies—they have to understand why they're doing certain things. You have to teach them, but it's also experiences—you have to gain the knowledge over time. It doesn't matter how good your tool is if you don't know how to drive it.

**Hartley:** To make matters worse, as we all know, there are not more than two universities on the planet that teach circuit board design. Young people coming out of school have no understanding of this stuff, and young engineers are being pushed into board design. In our day, you could choose a path. Today, almost all the young engineers are doing the circuit design and the board design, and they're thrown into it with no knowledge of the things Barry is talking about; they've got to figure it out quickly.

**Shaughnessy:** Could you guys teach a new EE how to do DDR design in one month? Or six months?

**Hartley:** I think it is possible within one design cycle. What do you think, Barry?

**Olney:** Yes, I've trained many PCB designers. I have taught a lot of people how to design high-speed boards and they usually become excellent after a year; I check their work and they make no mistakes; they just follow the process. They probably need at least two design cycles to be a proficient independent designer, as each board is different.

**Hartley:** If you want to learn the whole high-speed design philosophy, it takes about a year. But if you just want to learn how to design DDR, you will have that down in one or two design cycles.

**Olney:** If you want to simulate it, that's a different story altogether as there are a lot of issues. Models are the biggest issue, and I've actually developed software that can help me, in par-

ticular, to extract sub-models from models because, quite often, models won't work. It's pretty straightforward: You extract the topology, the transmission line into the simulation tool from the board layout, and then you attach the IBIS models to each end, but when they don't work, what do you do? You've got nothing. Now, I have software that extracts the sub-models from a non-working model, and it creates a new model that I can use.

The other issue is a lot of signals like PCI Express and Ethernet go off the board, so what then? You have a connector on the end of the board that all your signals go to. The trick there is not to use the connector model but to use the driver model in reverse and use the driver model so you can simulate the actual board. You can't control what happens off the board, you can only make the board perfect and then someone attaches a mile of cable to it.

**Hartley:** That's the other thing—you basically have to simulate that cable at its impedance to determine how to design the board to drive that cable properly.

**Olney:** Not really, because the impedance of the receiver, if you stick it in place of the connector, you will have the right impedance.

**Hartley:** The receiver will, yes, but what if it's a long cable? Long enough that you care about its impedance?

**Olney:** That's what I'm saying: You can't control what happens off the board; you can only deal with what's on the board.

**Shaughnessy:** What final advice would you give to someone who is working with DDR and is having a hell of a time with it? Any tips or tricks? Anything?

**Olney:** Basically, you need to read a lot and educate yourself. With DDR3, 4, and 5, it is important to use the correct impedance of

40/80 ohms, and check the timing and channel isolation. Simulate the clocks/strobes and the two worst nets of each group as you don't have time to evaluate each individual signal. Then compare the flight times of each signal to the clocks/strobes, which is a lot quicker.

---

## Understand at what point a transmission line becomes distributed vs. lumped and how you treat it. You must control its impedance.

---

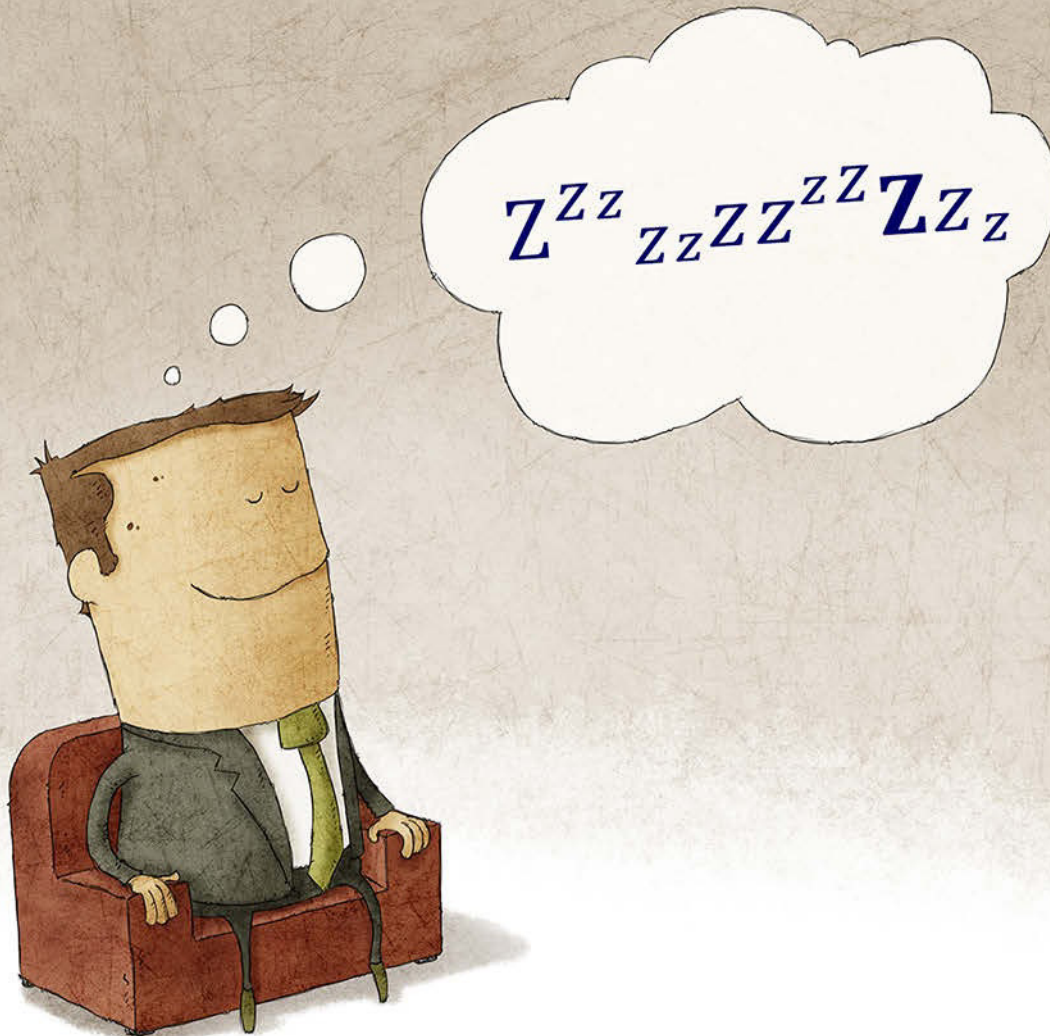
**Hartley:** I would say read a lot. If you have access to good simulation tools, use them. Understand at what point a transmission line becomes distributed vs. lumped and how you treat it. You must control its impedance. Usually with DDR, of course, you must control impedance; that's true of any high-speed design. If it's short enough you don't have to worry about it, but if it's a distributed link, of course you do. I treat all lines as if they are distributed, assuming they may route long enough to be distributed. If they're not distributed, then we simply don't terminate.

**Matties:** This is great, guys. We appreciate so much your time, thoughts, and insights. Thank you both.

**Olney:** Thank you. Great speaking with you guys, and Rick, nice to chat with you.

**Hartley:** Thank you, Barry. I read your column every month. I'm glad we had the chance to talk. **DESIGN007**

# We **DREAM** Impedance!



Did you know that two seemingly unrelated concepts are the foundation of a product's performance and reliability?

- Transmission line impedance and
- Power Distribution Network impedance

**DISCOVER MORE**

iCD software quickly and accurately analyzes impedance so you can sleep at night.

**iCD Design Integrity: Intuitive software for high-speed PCB design.**

*"iCD Design Integrity software features a myriad of functionality specifically developed for PCB designers."*

– Barry Olney

